1. Home . . . 9
1.1 JIRA 101 . . . 10
1.2 JIRA User's Guide . . . 18
1.2.1 JIRA Concepts . . . 19
1.2.1.1 What is an Issue? . . . 19
1.2.1.2 What is a Project? . . . 21
1.2.1.3 What is Workflow? . . . 23
1.2.2 Getting Started . . . 23
1.2.2.1 Logging into JIRA . . . 24
1.2.2.2 Exploring the JIRA Workspace . . . 24
1.2.2.3 Using Keyboard Shortcuts . . . 25
1.2.3 Working with Issues . . . 26
1.2.3.1 Creating an Issue . . . 26
1.2.3.2 Setting Security on an Issue . . . 27
1.2.3.3 Attaching a Screenshot . . . 28
1.2.3.4 Editing Rich-Text Fields . . . 31
1.2.3.5 Scheduling an Issue . . . 34
1.2.3.6 Logging Work on an Issue . . . 35
1.2.3.7 Creating a Sub-Task . . . 38
1.2.3.8 Watching and Voting on an Issue . . . 41
1.2.3.9 Commenting on an Issue . . . 42
1.2.3.10 Linking Issues . . . 44
1.2.3.11 Cloning an Issue . . . 47
1.2.3.12 Moving an Issue . . . 48
1.2.3.13 Viewing an Issue's FishEye Changesets . . . 50
1.2.3.14 Viewing the Bamboo Builds related to an Issue . . . 51
1.2.3.15 Modifying Multiple ('Bulk') Issues . . . 52
1.2.4 Searching for Issues . . . 58
1.2.4.1 Using Quick Search . . . 60
1.2.4.2 Performing Text Searches . . . 62
1.2.4.3 Advanced Searching . . . 65
1.2.4.4 Saving Searches ('Issue Filters') . . . 83
1.2.4.5 Receiving Search Results via Email . . . 90
1.2.4.6 Using the Issue Navigator . . . 92
1.2.4.6.1 Customising your Issue Navigator . . . 94
1.2.4.6.2 Displaying Search Results in XML . . . 95
1.2.4.6.3 Receiving Search Results as an RSS Feed . . . 100
1.2.4.6.4 Exporting Search Results to Microsoft Word . . . 101
1.2.4.6.5 Exporting Search Results to Microsoft Excel . . . 102
1.2.4.6.6 Displaying Search Results as a Chart . . . 104
1.2.5 Generating Reports . . . 107
1.2.5.1 Version Workload Report . . . 108
1.2.5.2 Time Tracking Report . . . 109
1.2.5.3 User Workload Report . . . 112
1.2.5.4 Single Level Group By Report . . . 113
1.2.5.5 Created vs Resolved Issues Report . . . 114
1.2.5.6 Resolution Time Report . . . 118
1.2.5.7 Pie Chart Report . . . 122
1.2.5.8 Average Age Report . . . 125
1.2.5.9 Recently Created Issues Report . . . 129
1.2.5.10 Time Since Issues Report . . . 133
1.2.6 Browsing a Project . . . 137
1.2.6.1 Browsing a Project's Summary . . . 137
1.2.6.2 Browsing a Project's Popular Issues . . . 139
1.2.6.3 Browsing a Project's Bamboo Builds . . . 139
1.2.6.4 Browsing a Project's Issues . . . 141
1.2.6.5 Browsing a Project's Road Map . . . 142
1.2.6.6 Browsing a Project's Change Log . . . 143
1.2.6.7 Browsing a Project's FishEye Changesets . . . 143
1.2.6.8 Browsing a Version's Summary . . . 145
1.2.6.9 Browsing a Version's Issues . . . 146
1.2.6.10 Browsing a Version's Popular Issues . . . 147
1.2.6.11 Browsing a Version's Bamboo Builds . . . 148
1.2.6.12 Browsing a Component's Summary . . . 150
1.2.6.13 Browsing a Component's Issues . . . 150
1.2.6.14 Browsing a Component's Popular Issues . . . 151
1.2.6.15 Browsing a Component's Road Map . . . 152
1.2.6.16 Browsing a Component's Change Log . . . 153
1.2.7 Customising the Dashboard . . . 154
1.2.7.1 Adding the Activity Stream Gadget . . . 157
1.2.7.2 Changing the Look and Behaviour of a Gadget . . . 159
1.2.7.3 Managing Multiple Dashboard Pages . . . 163
1.2.7.4 Adding the Administration Gadget . . . 168
1.2.7.5 Adding the Assigned To Me Gadget . . . 169
1.2.7.6 Adding the Average Age Gadget . . . 169

1.2.7.7 Adding the Bamboo Plan Summary Gadget . . . 172
1.2.7.8 Adding the Bamboo Status Gadget . . . 175
1.2.7.9 Adding the Bugzilla ID Search Gadget . . . 175
1.2.7.10 Adding the Calendar Gadget . . . 176
1.2.7.11 Adding the Created vs Resolved Gadget . . . 177
1.2.7.12 Adding the Favourite Filters Gadget . . . 180
1.2.7.13 Adding the Filter Statistics Gadget . . . 180
1.2.7.14 Adding the Two-Dimensional Filter Statistics Gadget . . . 181
1.2.7.15 Adding the In Progress Gadget . . . 182
1.2.7.16 Adding the Introduction Gadget . . . 183
1.2.7.17 Adding the Pie Chart Gadget . . . 183
1.2.7.18 Adding the Project Table Gadget . . . 186
1.2.7.19 Adding the Project Gadget . . . 187
1.2.7.20 Adding the Projects Gadget . . . 187
1.2.7.21 Adding the Project Statistics Gadget . . . 189
1.2.7.22 Adding the Quick Links Gadget . . . 190
1.2.7.23 Adding the Recently Created Issues Gadget . . . 190
1.2.7.24 Adding the Resolution Time Gadget . . . 193
1.2.7.25 Adding the Road Map Gadget . . . 195
1.2.7.26 Adding the Saved Filter Gadget . . . 196
1.2.7.27 Adding the Text Gadget . . . 197
1.2.7.28 Adding the Time Since Issues Gadget . . . 197
1.2.7.29 Adding the Voted Issues Gadget . . . 201
1.2.7.30 Adding the Watched Issues Gadget . . . 201
1.2.8 Managing your User Profile . . . 202
1.2.8.1 Changing your Password . . . 203
1.2.8.2 Choosing a Language . . . 203
1.3 JIRA Administrator's Guide . . . 204
1.3.1 Getting Help . . . 206
1.3.2 Configuring the Layout and Design . . . 207
1.3.2.1 Customising the Look and Feel . . . 208
1.3.2.2 Choosing a Default Language . . . 210
1.3.2.3 Configuring the Default Issue Navigator . . . 211
1.3.2.4 Configuring the Default Dashboard . . . 212
1.3.2.4.1 Using Dashboard Gadgets . . . 212
1.3.2.4.2 Adding a Gadget to the Directory . . . 214
1.3.2.5 Configuring an Announcement Banner . . . 216
1.3.2.6 Enabling Logout Confirmation . . . 217
1.3.3 User and Group Management . . . 218
1.3.3.1 Managing Users . . . 218
1.3.3.2 Managing Groups . . . 221
1.3.3.3 Managing Project Roles . . . 223
1.3.3.4 Migrating User Groups to Project Roles . . . 226
1.3.3.5 Enabling Public Signup and CAPTCHA . . . 233
1.3.3.6 Integrating JIRA with LDAP . . . 235
1.3.3.7 Integrating JIRA with Crowd . . . 240
1.3.3.8 Configuring Trusted Applications . . . 240
1.3.4 Project Management . . . 241
1.3.4.1 Defining a Project . . . 241
1.3.4.2 Managing Project Role Membership . . . 243
1.3.4.3 Defining a Component . . . 246
1.3.4.4 Managing Versions . . . 249
1.3.4.4.1 Creating Release Notes . . . 253
1.3.4.5 Configuring Project Keys . . . 254
1.3.5 Configuring Security . . . 254
1.3.5.1 Managing Global Permissions . . . 255
1.3.5.2 Managing Project Permissions . . . 257
1.3.5.3 Configuring Issue Level Security . . . 264
1.3.6 Configuring Fields and Screens . . . 272
1.3.6.1 Configuring Built-in Fields . . . 274
1.3.6.1.1 Defining 'Issue Type' Field Values . . . 274
1.3.6.1.2 Defining 'Priority' Field Values . . . 280
1.3.6.1.3 Defining 'Resolution' Field Values . . . 282
1.3.6.1.4 Defining 'Status' Field Values . . . 282
1.3.6.1.5 Translating Resolutions, Priorities, Statuses and Issue Types . . . 283
1.3.6.2 Adding a Custom Field . . . 285
1.3.6.2.1 Creating Help for a Custom Field . . . 290
1.3.6.2.2 Configuring a Custom Field . . . 291
1.3.6.3 Specifying Field Behaviour . . . 295
1.3.6.3.1 Associating Field Behaviour with Issue Types . . . 301
1.3.6.3.2 Configuring Rich-Text Renderers . . . 305
1.3.6.4 Defining a Screen . . . 313
1.3.6.4.1 Associating a Screen with an Issue Operation . . . 319
1.3.6.4.2 Associating a Screen with an Issue Type . . . 322
1.3.7 Configuring Workflow . . . 326

1.3.7.1 Activating Workflow . . . 339
1.3.7.2 Adding a Custom Event . . . 341
1.3.8 Configuring Email Notifications . . . 344
1.3.8.1 Creating a Notification Scheme . . . 345
1.3.8.2 Customising Email Content . . . 348
1.3.8.3 Creating Issues and Comments from Email . . . 349
1.3.8.4 Configuring JIRA to Send SMTP Mail . . . 352
1.3.9 Migrating from Other Issue Trackers . . . 355
1.3.9.1 Importing Data From CSV . . . 357
1.3.9.2 Importing Data from Bugzilla . . . 363
1.3.9.2.1 Migrating from Bugzilla 3.0.3 . . . 372
1.3.9.2.2 Modifying the Bugzilla Importer . . . 372
1.3.9.3 Importing Data From Mantis . . . 376
1.3.9.4 Importing Data From FogBugz . . . 376
1.3.10 Moving or Archiving Individual Projects . . . 381
1.3.10.1 Archiving a Project . . . 382
1.3.10.2 Splitting a JIRA instance . . . 383
1.3.11 Integrating with a Source Control System . . . 383
1.3.11.1 Integrating JIRA with CVS and ViewCVS . . . 383
1.3.11.2 Integrating JIRA with Subversion . . . 387
1.3.11.3 Integrating JIRA with Perforce . . . 388
1.3.11.4 Integrating JIRA with ClearCase . . . 390
1.3.11.5 Integrating JIRA with FishEye . . . 390
1.3.12 Configuring Global Settings . . . 390
1.3.12.1 Configuring JIRA Options . . . 391
1.3.12.2 Setting properties and options on startup . . . 393
1.3.12.3 Advanced JIRA configuration with jira-application.properties . . . 397
1.3.12.4 Enabling File Attachments . . . 397
1.3.12.4.1 Enabling Thumbnails for Attachments . . . 400
1.3.12.5 Enabling Sub-tasks . . . 401
1.3.12.6 Enabling Issue Linking . . . 402
1.3.12.7 Enabling Trackback . . . 403
1.3.12.8 Enabling Time Tracking . . . 405
1.3.12.9 Configuring OAuth Consumers . . . 406
1.3.13 Server Administration . . . 408
1.3.13.1 Search Indexing . . . 408
1.3.13.1.1 Re-Indexing after Major Configuration Changes . . . 409
1.3.13.2 Backing Up Data . . . 409
1.3.13.2.1 Automating JIRA Backups . . . 410
1.3.13.3 Restoring Data . . . 412
1.3.13.3.1 Restoring a Project from Backup . . . 414
1.3.13.4 Using the Database Integrity Checker . . . 422
1.3.13.5 Optimising Performance . . . 424
1.3.13.6 Precompiling JSP pages . . . 427
1.3.13.7 Logging and Profiling . . . 428
1.3.13.8 Increasing JIRA Memory . . . 430
1.3.13.9 Database Indexing . . . 436
1.3.13.10 Generating a Thread Dump . . . 437
1.3.13.11 Using robots.txt to hide from Search Engines . . . 439
1.3.14 Appendix A - Extending JIRA . . . 440
1.3.14.1 Managing JIRA's Plugins . . . 441
1.3.14.2 Listeners . . . 443
1.3.14.3 Services . . . 444
1.3.14.4 Jelly Tags . . . 453
1.3.14.5 JIRA Toolkit (Customer Support Extensions) . . . 478
1.3.14.6 Developer Guides . . . 478
1.3.14.7 Building JIRA from Source . . . 478
1.3.14.8 API Documentation . . . 481
1.4 JIRA Administrators FAQ . . . 481
1.4.1 Installation Notes . . . 484
1.4.1.1 Application Server Notes . . . 485
1.4.1.1.1 Geronimo Notes . . . 486
1.4.1.1.2 JBoss 4 notes . . . 489
1.4.1.1.3 Jetty Notes . . . 489
1.4.1.1.4 JOnAS Notes . . . 489
1.4.1.1.5 Oracle Weblogic Notes . . . 490
1.4.1.1.6 Resin Notes . . . 490
1.4.1.1.7 Sun Java App Server (SJAS or Glassfish) Notes . . . 493
1.4.1.1.8 Tomcat 5.5 notes . . . 493
1.4.1.1.9 Tomcat 6.0 notes . . . 493
1.4.1.1.10 Websphere 5.1 Notes . . . 493
1.4.1.1.11 Websphere 6.x Notes . . . 493
1.4.1.2 Database Notes . . . 495
1.4.1.2.1 JIRA and DB2 . . . 495
1.4.1.2.2 JIRA and Firebird . . . 495

1.4.1.2.3 JIRA and HSQL . . . 496
1.4.1.2.4 JIRA and MS SQL Server . . . 497
1.4.1.2.5 JIRA and MYSQL . . . 505
1.4.1.2.6 JIRA and Oracle . . . 514
1.4.1.2.7 JIRA and PostgreSQL . . . 517
1.4.1.2.8 JIRA and Sybase . . . 520
1.4.1.2.9 Connecting JIRA to a different database than the one provided as default . . . 520
1.4.1.2.10 Database Disappears . . . 520
1.4.1.2.11 Incorrect database type specified . . . 521
1.4.1.2.12 Making databases read-only for upgrade backups . . . 522
1.4.1.2.13 Restarting from Setup Wizard in JIRA Standalone . . . 524
1.4.1.2.14 Surviving Connection Closures . . . 524
1.4.1.3 Configure JIRA as service on Mac OS X . . . 526
1.4.1.4 Configuring IIS with Tomcat . . . 527
1.4.1.5 Deploying JIRA in a clustered environment . . . 528
1.4.1.6 Installation Troubleshooting Guide . . . 529
1.4.1.7 Installing a LDAP server on Debian Linux for use with JIRA . . . 529
1.4.1.8 Installing Java on Ubuntu or Debian . . . 533
1.4.1.9 java.lang.NoClassDefFoundError . . . 535
1.4.1.10 JVM and Appserver configuration info . . . 535
1.4.1.10.1 Causes of OutOfMemoryErrors . . . 535
1.4.1.10.2 Issues with tomcat 4.x mail and activation jars . . . 542
1.4.1.10.3 Java crashes . . . 542
1.4.1.10.4 Setting up a database connection in Websphere 5.1 . . . 544
1.4.1.11 Known Java Issues . . . 548
1.4.1.12 LicenseFactory error after upgrading JIRA . . . 549
1.4.1.13 Licensing . . . 549
1.4.1.14 Logging request headers . . . 549
1.4.1.15 Solaris ClassNotFoundException . . . 549
1.4.1.16 Transaction Isolation Change exceptions with JBoss . . . 552
1.4.1.17 Windows cannot find -Xms128m . . . 552
1.4.2 Performance FAQ . . . 552
1.4.2.1 Finding out how many requests a web application currently has . . . 553
1.4.2.2 Indexing in JIRA . . . 553
1.4.2.2.1 Thread Dump in JIRA Log File . . . 554
1.4.2.3 Issue Caching Problems in early JIRA versions . . . 563
1.4.2.4 Is your JIRA Running Slowly . . . 563
1.4.2.4.1 Testing Database Access Speed . . . 565
1.4.2.4.2 Testing Disk Access Speed . . . 566
1.4.2.4.3 Windows Performance Manager . . . 568
1.4.2.5 Profiling Memory and CPU usage with YourKit . . . 569
1.4.2.5.1 Setting Up YourKit Agent Parameter . . . 573
1.4.2.5.2 Troubleshooting Profiling Memory and CPU usage with YourKit . . . 573
1.4.2.6 Slow JIRA Operations . . . 574
1.4.3 Usage FAQ . . . 574
1.4.3.1 How do I unlock my JIRA home directory? . . . 576
1.4.3.2 Changing the Project Key . . . 576
1.4.3.3 Mail error - Unable to relay . . . 576
1.4.3.4 Importing user from LDAP . . . 576
1.4.3.4.1 Neat JIRA LDAP tricks . . . 578
1.4.3.5 JIRA Standalone quick source modifications . . . 578
1.4.3.6 Commonly Asked CSV Questions and Known Issues . . . 578
1.4.3.7 Troubleshooting Issue Creation Via Email . . . 580
1.4.3.8 How to clear the resolution field when the issue is reopened . . . 580
1.4.3.9 How to re-order the list of issue operation in an issue . . . 580
1.4.3.10 How to resize Free Text Field customfield . . . 581
1.4.3.11 Using validators to make custom fields required on transition screens . . . 581
1.4.3.12 Project-specific email templates . . . 581
1.4.3.13 How to configure sub-task to have a specific screen? . . . 581
1.4.3.14 User access logging . . . 581
1.4.3.15 Connecting to SSL services . . . 583
1.4.3.16 Editing a custom field option . . . 588
1.4.3.17 Allow editing of Closed Issues . . . 588
1.4.3.18 Updating your JIRA License Details . . . 589
1.4.3.19 Linking to local file under Firefox . . . 590
1.4.3.20 Creating Issues via direct HTML links . . . 590
1.4.3.21 How to create a download link to a file . . . 593
1.4.3.22 Current Reporter Browse Project Permission . . . 593
1.4.3.23 How to display custom field of the sub-task in the parent issue screen? . . . 594
1.4.3.24 How the CreateOrCommentHandler works? . . . 594
1.4.3.25 International characters in notification email subjects . . . 595
1.4.3.26 Sending a JIRA data backup to support . . . 595
1.4.3.27 Letting customers only create issues . . . 597
1.4.3.28 Asking for an attachment on the Create Issue page . . . 598
1.4.3.29 Limiting the number of issues returned from a search view such as an RSS feed . . . 598

1.4.3.30 Anonymising data . . . 600
1.4.3.31 Single Sign-on . . . 601
1.4.3.32 Scheme Entity Relations Map . . . 601
1.4.3.33 Custom field column not visible in Issue Navigator . . . 601
1.4.3.34 XML format for import & export files . . . 602
1.4.3.35 Configuring project specific security . . . 602
1.4.3.35.1 Controlling project visibility . . . 602
1.4.3.35.2 Using Project Level Security with Project Roles . . . 602
1.4.3.35.3 Using Project Level Security with User Groups . . . 604
1.4.3.36 Changing Custom Field Types . . . 609
1.4.3.37 Re-order workflow transactions . . . 610
1.4.3.38 Creating an Unassigned Issue . . . 610
1.4.3.39 Workflow Properties . . . 610
1.4.3.40 Editing the work log . . . 611
1.4.3.41 Field Layout Schemes in JIRA 3.x . . . 612
1.4.3.42 How to display a different format for the Number customfield . . . 612
1.4.3.43 How to remove 'NONE' from a select list custom field . . . 613
1.4.3.44 Plans for JIRA's LDAP integration . . . 613
1.4.3.45 What are the differences between JIRA Enterprise, Professional and Standard editions? . . . 613
1.4.3.46 Obsolete guides . . . 613
1.4.3.46.1 Changing scope of existing custom fields . . . 614
1.4.3.46.2 Large text fields in Oracle . . . 615
1.4.3.46.3 Removing invalid characters from XML backups . . . 617
1.4.3.47 Changing JIRA's log output . . . 618
1.4.3.47.1 Logging email protocol details . . . 619
1.4.3.48 Changing the username in JIRA . . . 622
1.4.3.49 Changing the Due Date input format . . . 622
1.4.3.50 Extra CSV Import Features . . . 622
1.4.3.51 Allowing users to create issues anonymously . . . 622
1.4.3.52 Deleting a user account . . . 623
1.4.3.53 Changing the Issue Key format . . . 623
1.4.3.54 QuickSearch guesses the issue key prefix (sometimes) . . . 623
1.4.3.55 Outward Link Description and Inward Link Description . . . 623
1.4.3.56 What does JIRA mean? . . . 624
1.4.3.57 Can I store customer details, like company, address and contact information, in JIRA? . . . 624
1.4.3.58 Words ignored when searching . . . 624
1.4.3.59 Using JIRA to Manage reusable modules . . . 624
1.4.3.60 Retrieving the JIRA Administrator . . . 625
1.4.3.61 Merging 2 JIRA instances . . . 627
1.4.3.62 Group Name Guidelines for JIRA . . . 628
1.4.3.63 Where are the JIRA logs? . . . 628
1.4.3.64 Escalating issues (or sending email notifications) when the set turnaround time is exceeded . . . 629
1.4.3.65 Why doesn't JIRA have a Severity field like Bugzilla? . . . 629
1.4.3.66 How to search by number range in the Issue Navigator . . . 629
1.4.3.67 Troubleshoot Mail Notification Problems . . . 629
1.4.3.68 Resolved issues appearing in Open issues filters . . . 631
1.4.3.69 Parsing utf-7 emails . . . 631
1.4.3.70 Adding custom content to the front page . . . 632
1.4.3.71 Changing the default session timeout . . . 632
1.4.3.72 How is JIRA pronounced? . . . 632
1.4.3.73 Translating JIRA . . . 633
1.4.3.74 How to ensure the Road Map tab is visible . . . 636
1.4.3.75 How to have long component version names display properly in the Issue Navigator . . . 636
1.4.3.76 Login problems . . . 636
1.4.3.77 We already have users & groups defined elsewhere - can JIRA make use of these? . . . 636
1.4.3.78 Associating a new screen with issue operations . . . 636
1.4.3.79 CVS ssh Jira Integration . . . 637
1.4.3.80 JIRA as a Support System . . . 637
1.4.3.80.1 Jelly Escalation . . . 640
1.4.3.80.2 Simple Escalation . . . 641
1.4.3.81 Changing the default attachment size limit . . . 642
1.4.3.82 Importing data . . . 642
1.4.4 Tips of the Trade . . . 643
1.5 JIRA Installation and Upgrade Guide . . . 645
1.5.1 JIRA Requirements . . . 646
1.5.1.1 Installing Java . . . 647
1.5.2 Installing JIRA Standalone . . . 649
1.5.2.1 Installing JIRA Standalone on Windows . . . 650
1.5.2.2 Installing JIRA Standalone on Mac OS . . . 652
1.5.2.3 Installing JIRA Standalone on Unix or Linux . . . 653
1.5.2.4 Configuring JIRA Standalone . . . 654
1.5.2.4.1 Running JIRA over SSL or HTTPS . . . 654
1.5.2.4.2 Integrating JIRA with a Web Server . . . 657
1.5.2.4.3 Running JIRA Standalone as a Service . . . 665
1.5.2.4.4 Installing Confluence into JIRA Standalone . . . 671

1.5.2.4.5 Changing JIRA Standalone's port . . . 671
1.5.3 Installing JIRA WAR-EAR . . . 672
1.5.3.1 Installing JIRA on Tomcat 6.0 . . . 673
1.5.3.2 Installing JIRA on Tomcat 5.5 . . . 678
1.5.3.3 Installing JIRA on Oracle WebLogic . . . 682
1.5.3.4 Installing JIRA on IBM Websphere 6.x . . . 686
1.5.3.5 JIRA WAR-EAR Configuration Overview . . . 686
1.5.3.5.1 Configuring the Entity Engine for JIRA . . . 687
1.5.3.6 Switching Application Servers . . . 689
1.5.4 Running the Setup Wizard . . . 689
1.5.5 Connecting JIRA to a Database . . . 692
1.5.5.1 Recommended Databases . . . 693
1.5.5.2 Connecting JIRA to SQL Server 2005 . . . 694
1.5.5.3 Connecting JIRA to MySQL . . . 697
1.5.5.4 Connecting JIRA to PostgreSQL . . . 700
1.5.5.5 Connecting JIRA to Oracle . . . 702
1.5.5.6 Connecting JIRA to DB2 . . . 705
1.5.5.7 Connecting JIRA to Firebird . . . 707
1.5.5.8 Connecting JIRA to Sybase . . . 709
1.5.5.9 Connecting JIRA to HSQLDB . . . 711
1.5.5.10 Switching Databases . . . 713
1.5.6 Upgrading JIRA . . . 713
1.5.6.1 Disabling Auto-Export . . . 717
1.5.7 Important Directories and Files . . . 718
1.5.7.1 JIRA Home Directory . . . 720
1.5.7.2 JIRA Installation Directory . . . 721
1.5.7.3 Setting your JIRA Home Directory . . . 721
1.5.8 JIRA Releases . . . 722
1.5.8.1 Production Releases . . . 722
1.5.8.1.1 All JIRA Release Notes (version 3.x and later) . . . 725
1.5.8.1.2 All JIRA Upgrade Guides (version 3.x and later) . . . 726
1.5.8.1.3 JIRA 3.13 Release Notes . . . 752
1.5.8.1.4 JIRA 3.12 Release Notes . . . 774
1.5.8.1.5 JIRA 3.11 Release Notes . . . 791
1.5.8.1.6 JIRA 3.10 Release Notes . . . 800
1.5.8.1.7 JIRA 3.9 Release Notes . . . 808
1.5.8.1.8 JIRA 3.8 Release Notes . . . 814
1.5.8.1.9 JIRA 3.7 Release Notes . . . 819
1.5.8.1.10 JIRA 3.6 Release Notes . . . 840
1.5.8.1.11 JIRA 3.5 Release Notes . . . 862
1.5.8.1.12 JIRA 3.4 and 3.4.1 Release Notes . . . 868
1.5.8.1.13 JIRA 3.3 Release Notes . . . 878
1.5.8.1.14 JIRA 3.2 Release Notes . . . 885
1.5.8.1.15 JIRA 3.1 Release Notes . . . 896
1.5.8.1.16 JIRA 3.0 Release Notes . . . 899
1.5.8.1.17 JIRA 2.6 Release Notes . . . 904
1.5.8.2 JIRA Release Summary . . . 907
1.5.8.3 Beta Releases . . . 909
1.5.8.3.1 JIRA 4.0 Beta 2 Release Notes . . . 909
1.5.8.3.2 JIRA 4.0 Beta 1 Release Notes . . . 942
1.5.8.4 EAP Releases . . . 957
1.5.8.4.1 JIRA 3.11 EAP Release Notes . . . 958
1.5.8.5 Security Advisories . . . 959
1.5.8.5.1 JIRA Security Advisory 2007-12-24 . . . 961
1.5.8.5.2 JIRA Security Advisory 2008-02-21 . . . 963
1.5.8.5.3 JIRA Security Advisory 2008-08-26 . . . 965
1.5.8.5.4 JIRA Security Advisory 2008-10-29 . . . 967
1.5.8.5.5 JIRA Security Advisory 2008-12-09 . . . 969
1.5.8.5.6 JIRA Security Advisory 2009-04-02 . . . 971
1.6 JIRA Development Hub . . . 975
1.6.1 JIRA Plugin Guide . . . 976
1.6.1.1 Component Plugin Module . . . 978
1.6.1.2 Component Tab Panel Plugin Module . . . 978
1.6.1.3 Configuring Plugins with Object Configurable Parameters . . . 979
1.6.1.3.1 Adding checkbox as a type . . . 981
1.6.1.4 Custom Field Plugin Module . . . 982
1.6.1.4.1 Custom field Velocity context unwrapped . . . 983
1.6.1.4.2 How to add searching to a Custom Field? . . . 984
1.6.1.5 Downloadable Plugin Resources . . . 985
1.6.1.6 Installing and Configuring Plugins . . . 986
1.6.1.7 Issue Operations Plugin Module . . . 986
1.6.1.8 Issue Tab Panel Plugin Module . . . 987
1.6.1.9 Portlet Plugin Module . . . 988
1.6.1.9.1 Quick start guide for creating a portlet plugin . . . 989
1.6.1.10 Project Tab Panel Plugin Module . . . 992

1.6.1.11 Report Plugin Module . . . 993
1.6.1.12 RPC Endpoint Plugin Module . . . 994
1.6.1.13 Search Request View Plugin Module . . . 995
1.6.1.14 Servlet Plugin Module . . . 999
1.6.1.15 User Format Plugin Guide . . . 100
1.6.1.16 Version Tab Panel Plugin Module . . . 100
1.6.1.17 Web Resources . . . 100
1.6.1.18 Web UI Plugin Module . . . 100
1.6.1.18.1 How to create a custom preset filter . . . 101
1.6.1.19 Webwork plugin module . . . 101
1.6.1.20 Workflow Plugin Modules . . . 101
1.6.2 Understanding how JIRA works . . . 101
1.6.2.1 Customising JIRA code . . . 101
1.6.2.2 Database Schema . . . 102
1.6.2.2.1 Change History . . . 103
1.6.2.2.2 Configuration properties . . . 103
1.6.2.2.3 Custom fields . . . 104
1.6.2.2.4 Example SQL queries for JIRA . . . 104
1.6.2.2.5 Issue fields . . . 104
1.6.2.2.6 Issue status and workflow . . . 105
1.6.2.2.7 User and Group Tables . . . 105
1.6.2.3 JIRA Architectural Overview . . . 105
1.6.2.3.1 JIRA Webwork Actions . . . 106
1.6.2.4 PicoContainer and JIRA . . . 106
1.6.2.4.1 Non-managed classes . . . 106
1.6.2.5 Sample Code . . . 106
1.6.2.5.1 Creating and Editing an Issue . . . 106
1.6.2.5.2 How to retrieve an issue with an ID or an IssueKey . . . 106
1.6.2.5.3 How to search for Issues based on CustomField . . . 106
1.6.2.5.4 How to search for Issues from within a Plugin . . . 106
1.6.2.5.5 Retrieving issue's links . . . 106
1.6.2.5.6 Working with Custom Fields . . . 106
1.6.3 Developer Tutorials . . . 106
1.6.3.1 Jelly Examples . . . 107
1.6.3.2 JIRA Custom Installer Guide . . . 107
1.6.3.3 JIRA Plugin Tutorials . . . 107
1.6.3.3.1 Available Permissions . . . 107
1.6.3.3.2 How to create a JIRA Portlet . . . 107
1.6.3.3.3 How to create a JIRA Report . . . 108
1.6.3.3.4 How to create a new Custom Field Type . . . 108
1.6.3.3.5 How to create Custom Workflow Elements for JIRA 3 . . . 109
1.6.3.4 JIRA RPC Services . . . 109
1.6.3.4.1 Creating a SOAP Client . . . 109
1.6.3.4.2 Creating a XML-RPC Client . . . 110
1.6.3.4.3 Enabling the RPC plugin . . . 110
1.6.3.4.4 JIRA XML-RPC Overview . . . 110
1.6.3.4.5 Frequently Asked RPC Questions and Known Issues . . . 111
1.6.3.5 Modifying JIRA Templates and JSPs . . . 111
1.6.3.5.1 Adding Custom Fields to Email . . . 111
1.6.3.5.2 Adding Custom Field to Issue Summary . . . 112
1.6.3.5.3 Adding JavaScript to all pages E.g. Google Analytics . . . 112
1.6.3.5.4 Contents of the Velocity Context . . . 112
1.6.3.5.5 Customising interface based on user's role . . . 112
1.6.3.5.6 Customising JIRA Excel Output . . . 112
1.6.3.5.7 Customizing text . . . 112
1.6.3.5.8 Mapping custom events to new email templates . . . 112
1.6.3.5.9 Tutorial - How to Create a Custom Release Notes Template Containing Release Comments . . . 112
1.6.3.5.10 Velocity Context for Email Templates . . . 113
1.6.3.6 Services, Listeners and Handlers . . . 113
1.6.3.6.1 Adding your own email handling classes . . . 113
1.6.3.7 3.6 Technical Improvements . . . 113
1.6.3.7.1 Programmatically determining automatic assignee . . . 113
1.6.3.7.2 Resolutions per workflow step . . . 113
1.6.3.7.3 Subtask creation form enhancements . . . 113
1.6.4 JIRA Developer FAQ . . . 114
1.6.4.1 Can I have a different Excel template for each report? . . . 114
1.6.4.2 How do I create a report with custom issue set using IssueNavigator . . . 114
1.6.4.3 How do I extract CVS commits for an issue . . . 114
1.6.4.4 How do I find all Versions in a Project? . . . 114
1.6.4.5 How do I find issues with field X? . . . 114
1.6.4.6 How do I get access to manager X? . . . 114
1.6.4.7 How do I get access to the current request? . . . 114
1.6.4.8 How do I get a handle to object X (specifically IssueNavigator)? . . . 114
1.6.4.9 How do I get a handle to the 'components' of a JIRA project? . . . 114
1.6.4.10 How do I get a handle to the model (issues) then iterate? . . . 114

1.6.4.11 How do I get determine which Issues a user is allowed to see? . . . 114
1.6.4.12 How do I get more help with SOAP? . . . 114
1.6.4.13 How do I know what else lurks in the Velocity context? . . . 114
1.6.4.14 How do I remove an issue operation? . . . 114
1.6.4.15 How do I set the value for default comment security level? . . . 114
1.6.4.16 How do I write a servlet for JIRA? . . . 114
1.6.4.17 How to set a custom field value on an issue? . . . 115
1.6.4.18 I have a GenericValue issue. How do I convert this to an Issue object? . . . 115
1.6.4.19 Is it possible to have a report plugin only appear for specific projects? . . . 115
1.6.4.20 Why am I getting 'Exception looking up public key' exception while building a plugin? . . . 115
1.6.4.21 How to make a custom field "importable" for project imports . . .
1.6.5 JIRA Developer Forums . . . 115
1.7 JIRA Tutorial Videos . . . 115
1.8 Local JIRA documentation . . . 116

Home

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Is this your first visit? Read all about JIRA.

Installing/upgrading JIRA

Installation Guide

Upgrade Guide

Release Notes

Download JIRA

Using/administering JIRA

User's Guide

Administrator's Guide

Development Hub | API Documentation

About

JIRA is a bug tracking, issue tracking, and project management application . JIRA has been designed with a focus on task achievement, is instantly usable and is flexible to work with.

JIRA Issues inside your IDE

Use the Atlassian Connector for Eclipse or the Atlassian Connector for IntelliJ IDEA to work with your JIRA issues right there in your development environment. Do you use Bamboo, Crucible or FishEye too? With the connector you can manage your builds and code reviews within your IDE, or move quickly between the IDE and a FishEye view of your source repository. **Hint:** The Atlassian IDE Connectors are free.

Resources

If you have a question about using JIRA, please feel free to contact us at support. You may also want to check out the mailing list forums:

- JIRA Announcements | subscribe
- JIRA General Forum | subscribe
- JIRA Developers Forum | subscribe

Other handy links:

- Knowledge Base
- JIRA Extensions & Plugins Library / Atlassian Plugin Exchange
- Issue Tracker & Feature Requests for JIRA

Download

You can download the JIRA documentation in PDF, HTML or XML formats.

Previous Versions
Documentation for previous versions of JIRA

Recently Updated

Local JIRA documentation (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (10 minutes ago)
Re: Updating JIRA Plugins for JIRA 4.0 (JIRA 4.0 Beta)	by Matt Doar (25 minutes ago)
Home (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (32 minutes ago)
Re: Updating JIRA Plugins for JIRA 4.0 (JIRA 4.0 Beta)	by Anton Mazkovoi (43 minutes ago)
Advanced Searching (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (2 hours ago)
Re: GreenHopper Quick Start Guide (JIRA 4.0 Beta)	by Andrew Lui [Atlassian Technical Writer] (2 hours ago)
GreenHopper Quick Start Guide (JIRA 4.0 Beta)	by Andrew Lui [Atlassian Technical Writer] (2 hours ago)
Re: GreenHopper Quick Start Guide (JIRA 4.0 Beta)	by Anonymous (9 hours ago)
Creating Release Notes (JIRA 4.0 Beta)	by Emily Stumpf (18 Aug)
Configuring OAuth Consumers (JIRA 4.0 Beta)	by Andrew Lui [Atlassian Technical Writer] (18 Aug)
Re: Tips of the Trade (JIRA 4.0 Beta)	by Andrew Lui [Atlassian Technical Writer] (17 Aug)
Tips of the Trade (JIRA 4.0 Beta)	by Andrew Lui [Atlassian Technical Writer] (17 Aug)
Installing JIRA on Tomcat 5.5 (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (17 Aug)
Installing JIRA on Tomcat 6.0 (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (17 Aug)
Installing JIRA WAR-EAR (JIRA 4.0 Beta)	by Rosie Jameson [Atlassian Technical Writer] (17 Aug)

JIRA 101

JIRA 101
Thank you for choosing JIRA. To help you get up and running quickly, we've compiled some easy instructions for configuring and using JIRA 3.13.

Getting Started

1. Installing JIRA

First things first. If you haven't already got JIRA up and running, carry out the following steps:

For Windows: (click to expand)

You may want to watch the video showing how to do this.

1. Download the JIRA Standalone Windows Installer (.EXE) file from the Atlassian Download Center.
2. Run the .EXE file, choose an installation directory and a port ('8080' will do). We recommend that you choose to 'Run JIRA as a service'.
3. JIRA will start automatically when the Installer finishes, if you selected the option to launch JIRA at the end of the Installer.
4. To access JIRA, go to your web browser and type this address: http://localhost:8080. Windows 'Start' menu shortcuts will also be added which you can also use to start and stop JIRA.
5. Follow the Setup Wizard. This will guide you through the process of setting up your JIRA server, creating an Admin user and (optionally) setting up email.

For more help on the technical procedures in this section, see the JIRA Installation Guide.

If you need assistance, please create a support ticket.

Before using JIRA as a production system, you need to switch from the default HSQL database, which is provided for evaluation purposes only. Please see the documentation for details.

For Mac: (click to expand)

You may want to watch the video showing how to do this.

1. Download the JIRA Standalone TAR (.GZ) file from the Atlassian Download Center, and unzip it.
2. Run `bin/startup.sh` to start JIRA.
3. To access JIRA, go to your web browser and type this address: http://localhost:8080.
4. Follow the Setup Wizard. This will guide you through the process of setting up your JIRA server, creating an Admin user and (optionally) setting up email.

For more help on the technical procedures in this section, see the JIRA Installation Guide.

If you need assistance, please create a support ticket.

Before using JIRA as a production system, you need to switch from the default HSQL database, which is provided for evaluation purposes only. Please see the documentation for details.

For Linux: (click to expand)

1. Download the JIRA Standalone TAR (.GZ) file from the Atlassian Download Center, and unzip it.
2. Install Java and set JAVA_HOME.
3. Run `bin/startup.sh` to start JIRA.
4. To access JIRA, go to your web browser and type this address: http://localhost:8080.
5. Follow the Setup Wizard. This will guide you through the process of setting up your JIRA server, creating an Admin user and (optionally) setting up email.

For more help on the technical procedures in this section, see the JIRA Installation Guide.

If you need assistance, please create a support ticket.

Before using JIRA as a production system, you need to switch from the default HSQL database, which is provided for evaluation purposes only. Please see the documentation for details.

2. Adding Users

For each of your users, you will need to do the following: (click to expand)

1. Click '**Administration**' in the top navigation bar.
2. Click '**User Browser**' in the left navigation column, then click '**Add User**'.
3. Enter the **Username**, **Password**, **Full Name** and **Email Address**; and (optionally) tick the box to send the user an email containing their account details. Then click the **Create** button. For more details, please see the documentation.
4. The User Browser will be displayed. Locate the new user and click the '**Groups**' link in the '**Operations**' column.
5. If the user is going to need to work on issues, select the '**jira-developers**' group and click the '**Join**' button. (If the user is only going to log issues, and not work on them, then they don't need to belong to the '**jira-developers**' group.) For more about groups, please see the documentation.
    - Note: Adding your users to the '**jira-developers**' group will automatically add them to the Default Members for the '**Developers**' project role. For more about project roles, please see the documentation.

You may want to suggest to your users that they take a look at 'Mastering the Basics' (below). You may also want to point them to the documentation on:

- Changing your Password
- Exploring the JIRA Dashboard

Your users will need to access JIRA at **http://<JIRA machine name>:8080** (not http://localhost:8080).

3. Creating a Project

Before you can create issues, you need to create a project to contain them. Here's how: (click to expand)

1. Click '**Administration**' in the top navigation bar.
2. Click '**Projects**' in the left navigation column, then click '**Add Project**'.
3. In the '**Name**' field, type a descriptive name for your project (typically two or three words, e.g. '**Purchase Orders**').
4. In the '**Key**' field, type a meaningful prefix for issues in your project (typically three or four characters, e.g.'**ORD**'). Note that this cannot be changed later.
5. In the '**Project Lead**' field, select the user to whom issues should be assigned by default.
6. If you chose to set up email when you installed JIRA (see above), change the '**Notification Scheme**' field from '**None**' to '**Default Notification Scheme**'. This will allow JIRA to automatically send emails to appropriate people when particular events occur (e.g. 'Issue Created', 'Issue Resolved'). For more about email, please see the documentation.
7. Leave the rest of the fields with their default values for now. Click the '**Add**' button.

For more about creating a project, please see the documentation or watch the video.

4. Optional Settings

Enabling time-tracking: (click to expand)

If you want to be able to log the amount of time you spend working on issues, you need to first do the following:

1. Click '**Administration**' in the top navigation bar.
2. Click '**Time Tracking**' (under '**Global Settings**') in the left navigation column.
3. In the '**Hours Per Day**' field, enter the number of hours in your organisation's working day (e.g. 8).
4. In the '**Days Per Week**' field, enter the number of days in your organisation's working week (e.g. 5).
5. Leave the '**Time Format**' as 'pretty'.
6. Click the '**Activate**' button.

For more about configuring time-tracking, please see the documentation.

Enabling sub-tasks: (click to expand)

If you want to be able to split up a 'parent' issue into a number of sub-tasks which can be assigned and tracked separately, you need to first do the following:

1. Click '**Administration**' in the top navigation bar.
2. Click '**Sub-Tasks**' (under '**Global Settings**') in the left navigation column.
3. Click the '**Enable**' link.

For more about configuring sub-tasks, please see the documentation.

Enabling attachments: (click to expand)

If you want to be able to attach files and screenshots to your JIRA issues, you need to first do the following:

1. Click '**Administration**' in the top navigation bar.
2. Click '**Attachments**' (under '**Global Settings**') in the left navigation column.
3. Click the '**Edit Configuration**' link.
4. In the '**Allow Attachments**' field, select '**ON**'.
5. In the '**Attachment Path**' field, type the absolute or relative path to the directory where attachments will be stored. Note that this directory should be given appropriate security as described in Security Overview.
6. *(Optional)* In the '**Enable Thumbnails**' field, select '**ON**' if you wish to enable image attachments to be displayed as thumbnails. For details please see _'*Image attachment thumbnails*'_.
7. Click the '**Update**' button.

For more about configuring attachments, please see the documentation.

Mastering the Basics

5. Creating an Issue

▷ To enter a new issue into JIRA: (click to expand)

1. Click the '**Create New Issue**' link in the top navigation-bar.
2. Select the relevant Project and Issue Type, then click the '**Next**' button.
3. Type a short description of the issue in the '**Summary**' field, then click the '**Create**' button.

For more details, please see the documentation.

6. Logging Work on an Issue

▷ To record the time that you have spent working on an issue, and action you have taken: (click to expand)

1. Go to the issue, and select '**Log work done**' from the '**Operations**' menu in the left column.
2. In the '**Time Spent**' field, enter the amount of time to be logged. Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes (e.g. to enter two hours of work, type '2h').
3. In the '**Work Description**' field, type a description or comment about the work you have done
4. Click the '**Log**' button.

For more details about the other options on this screen, please see the documentation.

7. Resolving an Issue

▷ Resolving an issue sets its Status to 'Resolved', indicating that work is complete. To resolve an issue: (click to expand)

1. Go to the issue, and select '**Resolve Issue**' from the '**Operations**' menu in the left column.
2. Select In the '**Resolution**' field, select the 'Resolution' that best describes the outcome (e.g. 'Fixed').
3. (Optional) In the '**Comment**' field, type a description or comment about the issue's resolution.
4. Click the '**Resolve**' button.

For more information about how an issue moves from one Status to another, please see the documentation.

8. Searching for Issues

▷ To use Quick Search: (click to expand)

- Use the box in the top right corner of every page to quickly search JIRA. You can type an issue key (e.g. `TEST-1234`) to jump directly to an issue, or use syntax like `my open issues` to immediately return all issues which are assigned to you and haven't yet been resolved.

▷ To use Regular Search: (click to expand)

1. Click '**Find Issues**' on the top navigation bar to display JIRA's searching and filtering panel.
2. Select the Project, Issue Type, or any other issue attributes of interest. You can also perform comprehensive text searches.

See the documentation for more information about saving your searches ('issue filters'), and receiving search results via email.

▷ To use Advanced Search: (click to expand)

JQL (JIRA Query Language) is coming in JIRA 4.0, bringing AND, OR, NOT and much more.
Watch this space!

9. Charting, Reporting and Exporting

To generate a chart: (click to expand)

1. View your search results (see *'Searching'* above) in the Issue Navigator.
2. Click '**Chart**'. (Note: only available if the Charting Plugin is installed.)
3. Choose your preferred type of chart, and enter any required configuration details. For more details, please see the documentation.

To generate a report: (click to expand)

1. Click '**Browse Projects**' in the top navigation bar.
2. Select the project you are interested in, if presented with a list.
3. Click '**Reports**' at the right of the screen and select the report of interest
4. Enter any required configuration details, then click '**Next**' to display your report (e.g. Time-Tracking Report).

To export data to MS-Word, MS-Excel, XML or RSS: (click to expand)

1. View your search results (see *'Searching'* above) in the Issue Navigator.
2. Click '**Word**', or your preferred format. For more details, please see the documentation.

Customising JIRA

(Note that you need to be an Administrator to do the tasks in this section.)

Before you begin: (click to expand)

You may want to create a sample project named '**Purchase Orders**', in which to perform the tasks described in this section. For instructions, please see *'3. Creating a Project'* (above).

10. Adding a new Issue Type

Why would I do this? (click to expand)

The Issue Type is one of the first things a user must choose when they create an issue.
Depending on how your organisation is using JIRA, you might want to add a new Issue Type.
For example, if you are using JIRA to track purchase orders, the default Issue Types ('Bug', 'Improvement', 'New Feature', 'Task') might not be relevant. So you might want to add a new Issue Type called '**Order**'.

How do I do this? (click to expand)

To add a new Issue Type called '**Order**', and associate it with a project called '**Purchase Orders**':

1. Click '**Administration**' in the top navigation bar.
2. Click '**Issue Types**' in the left navigation column (under '**Issue Settings**').
3. In the '**Add New Issue Type**' form, in the '**Name**' field, type '**Order**. In the '**Description**' field, type '**A purchase order**'. Then click the '**Add**' button'. (For more about adding Issue Types, and icons, please see the documentation.)
4. Click the '**Issue Types Scheme**' tab at the top of the '**Manage Issue Types**' screen.
5. In the '**Add New Issue Type Scheme**' form, in the '**Name**' field, type '**Purchase Order Issue Type Scheme**'. Then click the '**Add**' button'. (For more about Issue Type Schemes, please see the documentation.)
6. In the '**Available Issue Types**' list, click the Issue Type called '**Order**' and drag it into the '**Issue Types for Current Scheme**' list. Then click the '**Save**' button.
7. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click '**Purchase Orders**'. The project details will be displayed.
8. Click the '**Select**' link next to the '**Issue Type Scheme**' field, select '**Purchase Order Issue Type Scheme**' and click the '**Associate**' button.

To test what you have done, create an issue in the '**Purchase Orders**' project. The only available Issue Type should be '**Order**'.

11. Adding a new Screen

Why would I do this? (click to expand)

Depending on how your organisation is using JIRA, you might want to add a purpose-built screen that will be displayed for particular types of issues, or for particular projects or workflows.
For example, if you are using JIRA to track purchase orders, some of the normal issue fields (e.g. 'Affects Version', 'Fix Version', 'Environment') might not be relevant. So you might want to create a simplified screen that omits these fields.

▷ How do I do this? (click to expand)

To add a new Screen called '**Purchase Order Screen**', and associate it with a project called '**Purchase Orders**':

1. Click '**Administration**' in the top navigation bar.
2. Click '**Screens**' in the left navigation column (under '**Issue Fields**').
3. Click the '**Copy**' link next to '**Default Screen**'.
4. In the '**Name**' field, type '**Purchase Order Screen**'. Then click the '**Add**' button'. (For more about Screens, please see the documentation.)
5. Click the '**Configure**' link next to '**Purchase Order Screen**'.
6. Tick the '**Remove**' box for the following fields: '**Affects Version**', '**Fix Version**', '**Environment**'. Then click the '**Remove**' button.
7. Click '**Screen Schemes**' in the left navigation column (under '**Issue Fields**').
8. In the '**Name**' field, type '**Purchase Order Screen Scheme**'. In the '**Default Screen**' field, select '**Purchase Order Screen**'. Then click the '**Add**' button'. (For more about Screen Schemes, please see the documentation.)
9. Click '**Issue Type Screen Schemes**' in the left navigation column (under '**Issue Fields**').
10. In the '**Name**' field, type '**Purchase Order Issue Type Screen Scheme**'. In the '**Screen Scheme**' field, select '**Purchase Order Screen Scheme**'. Then click the '**Add**' button'. (For more about Issue Type Screen Schemes, please see the documentation.)
11. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click '**Purchase Orders**'. The project details will be displayed.
12. Click the '**Select**' link next to the '**Issue Type Screen Scheme**' field, select '**Purchase Order Issue Type Screen Scheme**' and click the '**Associate**' button.

To test what you have done, view an issue in the '**Purchase Orders**' project. You shouldn't see the '**Affects Version**', '**Fix Version**' or '**Environment**' fields.

12. Adding a new Custom Field

▷ Why would I do this? (click to expand)

Depending on how your organisation is using JIRA, you might need to add a 'custom' field that will be displayed for particular types of issues, or for particular projects.
For example, if you are using JIRA to track purchase orders, you might create a custom field called '**Supplier**'.

▷ How do I do this? (click to expand)

To create a new custom field called '**Supplier**' and put it on the '**Purchase Order Screen**':

1. Click '**Administration**' in the top navigation bar.
2. Click '**Custom Fields**' in the left navigation column (under '**Issue Fields**'), then click '**Add Custom Field**'.
3. On the '**Create Custom Field - Step 1**' screen, in the '**Field Type**' field, choose '**Select List**'. Then click the '**Next**' button.
4. On the '**Create Custom Field - Step 2**' screen:
    a. in the '**Name**' field, type '**Supplier**'.
    b. in the '**Description**' field, type '**Choose the supplier for this Purchase Order**'.
    c. under '**Choose applicable issue types**' select '**Order**'. Then click the '**Finish**' button.
5. On the '**Associate field Order to screens**' screen, tick the check-box for '**Purchase Order Screen**'. Then click the '**Update**' button.
6. On the '**View Custom Fields**' screen, click the '**Configure**' link next to '**Supplier**'. Then click '**Edit Options**'.
7. Add three options: '**ABC Pty Ltd**', '**ACME Pty Ltd**', '**XYZ Pty Ltd**'. Choose '**ACME Pty Ltd**' as the default. Then click the '**Done**' button.

To test what you have done, create an issue in the '**Purchase Orders**' project. You should see a field called '**Supplier**' that has a drop-down box containing your three options.

Note: you may also want to watch the video on adding a custom field.

13. Adding a new Issue Status and Workflow

▷ Why would I do this? (click to expand)

Depending on how your organisation is using JIRA, you might need to add a new 'Status' , i.e. a new step in the issue 'workflow'.
For example, if you are using JIRA to track purchase orders, you might add a new Status called '**Purchase Approved**' .

▷ How do I do this? (click to expand)

To add a new Status called '**Purchase Approved**', and create a new workflow that has an extra step between '**Open**' and '**In Progress**':

1. Click '**Administration**' in the top navigation bar.
2. Click '**Statuses**' in the left navigation column (under '**Issue Settings**').
3. In the '**Name**' field, type '**Purchase Approved**'. Then click the '**Add**' button'. (For more about adding Statuses, and icons, please see the documentation.)
4. Click '**Workflows**' in the left navigation column.
5. Click the '**Copy**' link next to '**jira (Read-only System Workflow)**'.
6. In the '**Workflow Name**' field, type '**Purchase Order Workflow**'. Then click the '**Add**' button'. (For more about Workflow, please see the documentation.)
7. Click the '**Steps**' link next to '**Purchase Order Workflow**'.
8. In the '**Add New Step**' form:
    a. in the '**Name**' field, type '**Purchase Approved**'.
    b. in the '**Linked Status**' field, select '**Purchase Approved**'.
    c. click the '**Add**' button.
9. Click the '**Add Transition**' link next to '**Open**':
    a. in the '**Name**' field, type '**Approve Purchase**'.
    b. in the '**Destination Step**' field, select '**Purchase Approved**'.
    c. click the '**Add**' button.
10. Click the '**Add Transition**' link next to '**Purchase Approved**':
    a. in the '**Name**' field, type '**Start Progress**'.
    b. in the '**Destination Step**' field, select '**In Progress**'.
    c. click the '**Add**' button.
11. Click the '**Delete Transitions**' link next to '**Open**'. Select '**Start Progress**' and click the '**Delete**' button.
12. Click '**Workflow Schemes**' in the left navigation column (under '**Issue Fields**'). Then click '**Add Workflow Scheme**'.
13. In the '**Name**' field, type '**Purchase Order Workflow Scheme**'. In the '**Default Screen**' field, select '**Purchase Order Screen**'. Then click the '**Add**' button'. (For more about Workflow Schemes, please see the documentation.)
14. Click the '**Workflows**' link next to '**Purchase Order Workflow Scheme**', click '**Assign Workflow**' and select '**Purchase Order Workflow**'. Then click the '**Add**' button.
15. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click '**Purchase Orders**'. The project details will be displayed.
16. Click the '**Select**' link next to the '**Workflow Scheme**' field, select '**Purchase Order Workflow Scheme**' and click the '**Associate**' button.

To test what you have done, create an issue in the '**Purchase Orders**' project. After you save the issue, the left column should contain a link called '**Approve Purchase**', but not a link called '**Start Progress**'.

Note: you may also want to watch the video on customising JIRA Workflow.

14. Using Permission Schemes, Groups and Project Roles

▷ Why would I do this? (click to expand)

A Permission scheme allows you to grant people 'permission' to work on issues in a project. The new project that you created previously is using JIRA's **Default Permission Scheme**. If you end up creating lots of projects, you might need to grant different people permission to work on different projects.
For example, if your organisation requires all software development issues to be tested by a Quality Assurance person before being closed, you could create a permission scheme called '**Software Development Permission Scheme**' in which you assign the 'Close Issue' permission to the appropriate people. You would then associate your new permission scheme with all your software development projects.
There are a number of ways to do this, depending on your requirements:

- Project roles enable you to associate *different people* with particular functions, for particular projects.
- Groups enable you to associate *the same people* with a particular function, for all projects that use this permission scheme.

▷ How do I do this? (click to expand)

To add a new permission scheme called '**Software Development Permission Scheme**', and a project role called '**Quality Assurance**':

1. Create a project role called '**Quality Assurance**':
    a. Click '**Administration**' in the top navigation bar.
    b. Click '**Project Roles**' (under '**Users, Groups and Roles**') in the left column.
    c. In the '**Name**' field, type '**Quality Assurance**'. Then click the '**Add Project Role**' button.
2. Create a permission scheme called '**Software Development Permission Scheme**', in which you assign the 'Close Issue' permission to the '**Quality Assurance**' project role:
    a. Click '**Permission Schemes**' (under '**Schemes**) in the left column.
    b. Click the '**Copy**' link next to '**Default Permission Scheme**'. A new permission scheme called '**Copy of Default Permission Scheme**' will be created.
    c. Click the '**Edit**' link next to '**Copy of Default Permission Scheme**'. On the '**Edit**' screen,
        i. change the '**Name**' to '**Software Development Permission Scheme**'
        ii. change the '**Description**' to '**Permission scheme for software development projects**'. Then click the '**Update**' button.
    d. Click the '**Permissions**' link next to '**Copy of Default Permission Scheme**'. On the '**Edit Permissions**' screen,
        i. for the '**Close Issues**' permission, click the '**Delete**' link next to '**Project Role (Developers)'**.
        ii. for the '**Close Issues**' permission, click the '**Add**' link. Click '**Project Role**' and choose '**Quality Assurance**'. Then click the '**Add**' button.
3. Associate the '**Software Development Permission Scheme**' with all your software development projects. Do the following for each relevant project:
    a. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click the name of the relevant project. The project details will be displayed.
    b. Click the '**Select**' link next to the '**Permission Scheme**' field, select '**Software Development Workflow Scheme**' and click the '**Associate**' button.
4. For each software development project, add the appropriate people to the '**Quality Assurance**' project role:
    a. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click the name of the relevant project. The project details will be displayed.
    b. Click the '**View members**' link next to '**Project Roles**'.
    c. Select the '**Edit**' link next to '**Quality Assurance**' and add the appropriate people.

Or, to add a new permission scheme called '**Software Development Permission Scheme'**, and a group called '**quality-assurance**':

1. Create a group called '**quality-assurance**', and add the appropriate people to it.
    a. Click '**Administration**' in the top navigation bar.
    b. Click '**Groups**' (under '**Users, Groups and Roles**') in the left column.
    c. In the '**Name**' field, type '**quality-assurance**'. Then click the '**Add Group**' button.
2. Create a permission scheme called '**Software Development Permission Scheme**', in which you assign the 'Close Issue' permission to the '**quality-assurance**' group.
    a. Click '**Permission Schemes**' (under '**Schemes**) in the left column.
    b. Click the '**Copy**' link next to '**Default Permission Scheme**'. A new permission scheme called '**Copy of Default Permission Scheme**' will be created.
    c. Click the '**Edit**' link next to '**Copy of Default Permission Scheme**'. On the '**Edit**' screen,
        i. change the '**Name**' to '**Software Development Permission Scheme**'
        ii. change the '**Description**' to '**Permission scheme for software development projects**'. Then click the '**Update**' button.
    d. Click the '**Permissions**' link next to '**Copy of Default Permission Scheme**'. On the '**Edit Permissions**' screen,
        i. for the '**Close Issues**' permission, click the '**Delete**' link next to '**Project Role (Developers)'**.
        ii. for the '**Close Issues**' permission, click the '**Add**' link. Click '**Group**' and choose '**quality-assurance**'. Then click the '**Add**' button.
3. Associate the '**Software Development Permission Scheme**' with all your software development projects. Do the following for each relevant project:
    a. Click '**Projects**' in the left navigation column. Then in the '**Name**' column, click the name of the relevant project. The project details will be displayed.
    b. Click the '**Select**' link next to the '**Permission Scheme**' field, select '**Software Development Workflow Scheme**' and click the '**Associate**' button.

Important Next Steps

(Note that you need to be an Administrator to do the tasks in this section.)

15. Connecting to an External Database

Before using JIRA as a production system, you need to switch from the default HSQL database, which is provided for evaluation purposes only. Please see the documentation for details.

16. Backing up Data

To back up your JIRA data, and establish processes for regular backups, please see the documentation.

Thank you for using JIRA.

Thanks for choosing JIRA. We're always happy to help. Feel free to email or call us with any questions you have.

JIRA User's Guide

This manual contains information on how to use JIRA, the issue tracking and project management system that you access from your web browser.

Manage your JIRA issues inside your IDE
Use the Atlassian Connector for Eclipse or the Atlassian Connector for IntelliJ IDEA to work with your JIRA issues right there in your development environment. Do you use Bamboo, Crucible or FishEye too? With the connector you can manage your builds and code reviews within your IDE, or move quickly between the IDE and a FishEye view of your source repository. **Hint:** The Atlassian IDE Connectors are free.

JIRA Concepts

- What is an Issue?
- What is a Project?
- What is Workflow?

Getting Started

- Logging into JIRA
- Exploring the JIRA Workspace
- Using Keyboard Shortcuts

Working with Issues

- Creating an Issue
- Setting Security on an Issue
- Attaching a Screenshot
- Editing Rich-Text Fields
- Scheduling an Issue
- Logging Work on an Issue
- Creating a Sub-Task
- Watching and Voting on an Issue
- Commenting on an Issue
- Linking Issues
- Cloning an Issue
- Moving an Issue
- Viewing an Issue's FishEye Changesets
- Viewing the Bamboo Builds related to an Issue
- Modifying Multiple ('Bulk') Issues

Searching for Issues

- Using Quick Search
- Performing Text Searches
- Advanced Searching
- Saving Searches ('Issue Filters')
- Receiving Search Results via Email
- Using the Issue Navigator
    - Customising your Issue Navigator
    - Displaying Search Results in XML
    - Receiving Search Results as an RSS Feed
    - Exporting Search Results to Microsoft Word
    - Exporting Search Results to Microsoft Excel
    - Displaying Search Results as a Chart

Generating Reports

- Version Workload Report
- Time Tracking Report
- User Workload Report
- Single Level Group By Report
- Created vs Resolved Issues Report
- Resolution Time Report
- Pie Chart Report
- Average Age Report

- Recently Created Issues Report
- Time Since Issues Report

Browsing a Project

- Browsing a Project's Summary
- Browsing a Project's Popular Issues
- Browsing a Project's Bamboo Builds
- Browsing a Project's Issues
- Browsing a Project's Road Map
- Browsing a Project's Change Log
- Browsing a Project's FishEye Changesets
- Browsing a Version's Summary
- Browsing a Version's Issues
- Browsing a Version's Popular Issues
- Browsing a Version's Bamboo Builds
- Browsing a Component's Summary
- Browsing a Component's Issues
- Browsing a Component's Popular Issues
- Browsing a Component's Road Map
- Browsing a Component's Change Log

Customising the Dashboard

- Adding the Activity Stream Gadget
- Changing the Look and Behaviour of a Gadget
- Managing Multiple Dashboard Pages
- Adding the Administration Gadget
- Adding the Assigned To Me Gadget
- Adding the Average Age Gadget
- Adding the Bamboo Plan Summary Gadget
- Adding the Bamboo Status Gadget
- Adding the Bugzilla ID Search Gadget
- Adding the Calendar Gadget
- Adding the Created vs Resolved Gadget
- Adding the Favourite Filters Gadget
- Adding the Filter Statistics Gadget
- Adding the Two-Dimensional Filter Statistics Gadget
- Adding the In Progress Gadget
- Adding the Introduction Gadget
- Adding the Pie Chart Gadget
- Adding the Project Table Gadget
- Adding the Project Gadget
- Adding the Projects Gadget
- Adding the Project Statistics Gadget
- Adding the Quick Links Gadget
- Adding the Recently Created Issues Gadget
- Adding the Resolution Time Gadget
- Adding the Road Map Gadget
- Adding the Saved Filter Gadget
- Adding the Text Gadget
- Adding the Time Since Issues Gadget
- Adding the Voted Issues Gadget
- Adding the Watched Issues Gadget

Managing your User Profile

- Changing your Password
- Choosing a Language

Note: for information on configuring JIRA, please see the *JIRA Administrator's Guide*.

JIRA Concepts

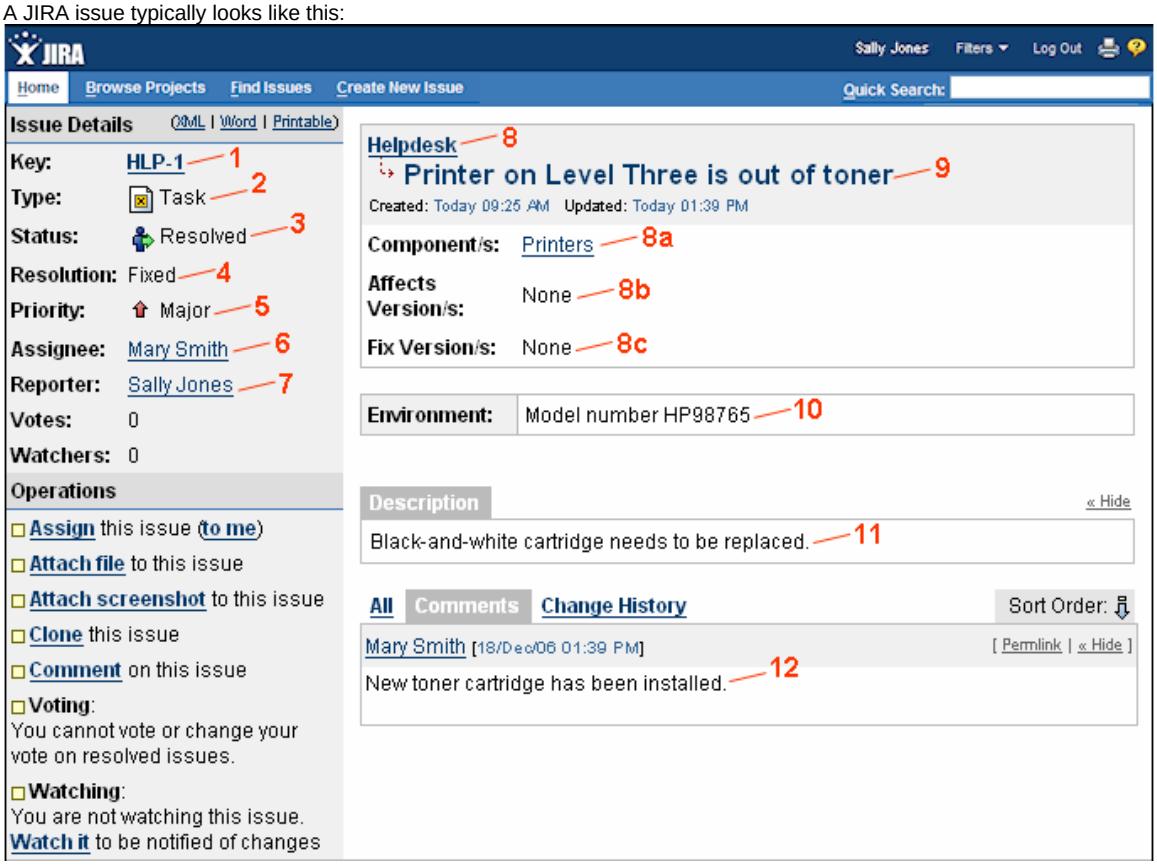
The following pages contain information on key concepts in JIRA:

- What is an Issue?
- What is a Project?
- What is Workflow?

What is an Issue?

Different organisations use JIRA to track different kinds of issues. Depending on how your organisation is using JIRA, an issue could represent a software bug, a project task, a helpdesk ticket, a leave request form, etc.

A JIRA issue typically looks like this:



> Your JIRA issues may look different to the above screenshot if your administrator has customised JIRA for your organisation.

The numbered fields shown in the above screenshot are:

1. **Key** — a unique identifier for this issue.
2. **Type** — see below for a list of types.
3. **Status** — the stage the issue is currently at in its lifecycle ('workflow'). See below for a list of statuses.
4. **Resolution** — a record of the issue's resolution (if the issue has been resolved)
5. **Priority** — the importance of the issue in relation to other issues. (See below for a list of priorities).
6. **Assignee** — the person to whom the issue is currently assigned.
7. **Reporter** — the person who entered the issue into the system.
8. **Project** — the 'parent' project to which the issue belongs.
    a. **Component(s)** *(if applicable)* — project component(s) to which this issue relates.
    b. **Affects Version(s)** *(if applicable)* — project version(s) for which the issue is (or was) manifesting.
    c. **Fix Version(s)** *(if applicable)* — project version(s) for which the issue was (or will be) fixed.
9. **Summary** — a brief one-line summary of the issue.
10. **Environment** *(if applicable)* — the hardware or software environment to which the issue relates.
11. **Description** — a detailed description of the issue.
12. **Comments** — added by people who are working on the issue.

Additionally, if your administrator has enabled 'Time-Tracking', the following fields will appear above the 'Description' field:

- **Original Estimate** — the total amount of time required to resolve the issue, as estimated when the issue was created. (This field cannot be modified once work has been logged on the issue).
- **Remaining Estimate** — the remaining amount of time required to resolve the issue. This field only appears when work has been logged on the issue.
- **Time Spent** — the sum of all the individual work logs for this issue. This field only appears when work has been logged on the issue.

Some of the most important fields - 'Type', 'Priority', 'Status' and 'Resolution' - are described as follows:

Issue Type

JIRA can be used to track many different types of issues. The default types are listed below, but please note that your JIRA administrator may have customised this list to suit your organisation.

Bug — A problem which impairs or prevents the functions of the product.
Improvement — An enhancement to an existing feature.
New Feature — A new feature of the product.
Task — A task that needs to be done.
Custom Issue — A custom issue type, as defined by your organisation if required.

Priority

An issue's priority indicates its relative importance. The default priorities are listed below; note that both the priorities and their meanings can be customised by your JIRA administrator to suit your organisation.

Blocker — Highest priority. Indicates that this issue takes precedence over all others.
Critical — Indicates that this issue is causing a problem and requires urgent attention.
Major — Indicates that this issue has a significant impact.
Minor — Indicates that this issue has a relatively minor impact.
Trivial — Lowest priority.

Status

Each issue has a status, which indicates where the issue currently is in its lifecycle ('workflow'). An issue starts as being 'Open', then generally progresses to 'Resolved' and then 'Closed'; or, depending on circumstances, it may progress to other statuses. Please also note that your JIRA administrator may have customised the available statuses to suit your organisation.

Open — This issue is in the initial 'Open' state, ready for the assignee to start work on it.
In Progress — This issue is being actively worked on at the moment by the assignee.
Resolved — A Resolution has been identified or implemented, and this issue is awaiting verification by the reporter. From here, issues are either 'Reopened' or are 'Closed'.
Reopened — This issue was once 'Resolved' or 'Closed', but is now being re-examined. (For example, an issue with a *Resolution* of 'Cannot Reproduce' is Reopened when more information becomes available and the issue becomes reproducible). From here, issues are either marked *In Progress, Resolved* or *Closed.*
Closed — This issue is complete.

Resolution

An issue can be resolved in many ways, only one of them being 'Fixed'. The default resolutions are listed below; note that your JIRA administrator may have customised these to suit your organisation.

Fixed — A fix for this issue has been implemented.
Won't Fix — This issue will not be fixed, e.g. it may no longer be relevant.
Duplicate — This issue is a duplicate of an existing issue. Note: it is recommended you create a link to the duplicated issue.
Incomplete — There is not enough information to work on this issue.
Cannot Reproduce — This issue could not be reproduced at this time, or not enough information was available to reproduce the issue. If more information becomes available, please reopen the issue.

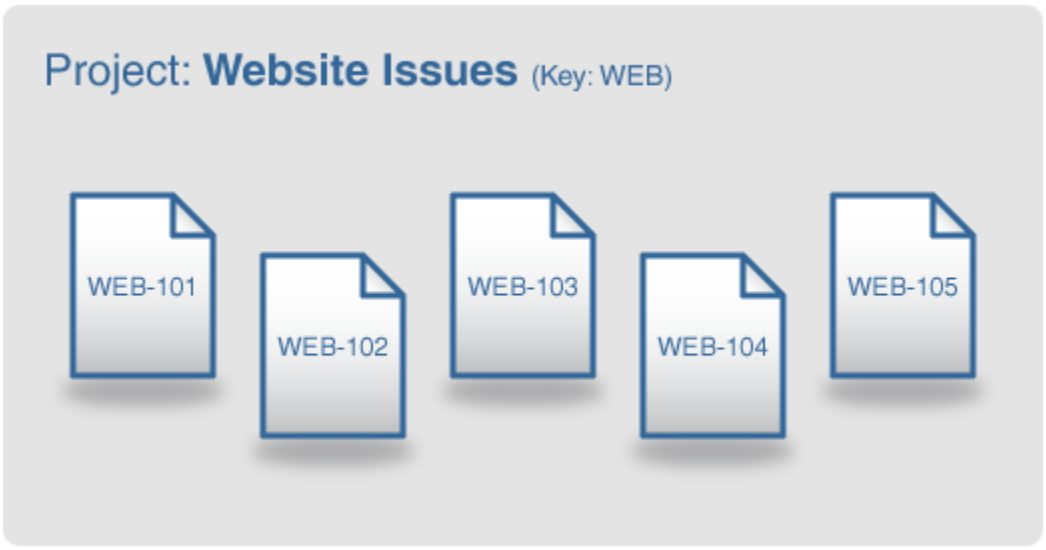
What is a Project?

A JIRA project is a collection of issues, and is defined according to your organisation's requirements. For example, a JIRA project could be:

- a software development project
- a marketing campaign
- a helpdesk system
- a leave request management system
- a website enhancement request system

Every issue belongs to a project. Each project has a *name* (e.g. **Website Issues**) and a *key* (e.g. **WEB**). The project key becomes the first part of

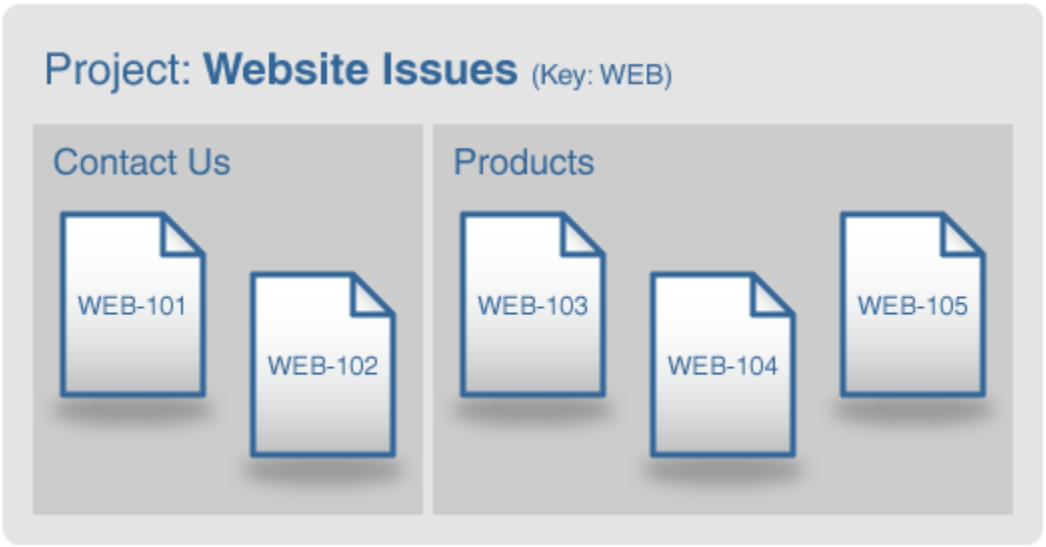
that project's *issue keys,* e.g. **WEB-101, WEB-102,** etc:



What is a component?

A project component is a logical grouping of issues within a project. Each project may consist of various components (or none), depending on your organisation's needs.

For example, a software development project could consist of components called 'Documentation', 'Backend', 'Email Subsystem', 'GUI'. A website enhancement request system might consist of components called 'Products', 'Contact Us', etc:



An issue can belong to zero, one or multiple components within a project.

What is a version?

For some types of projects, particularly software development, it is useful to be able to associate an issue with a particular project *version* (e.g. 1.0 beta, 1.0, 1.2, 2.0).

Issues have two fields that relate to versions:

- **Affects Version(s)** — this is the version(s) in which the issue is manifesting. For instance, a software bug might affect versions 1.1 and 1.2.
- **Fix Version(s)** — this is the version(s) in which the issue was (or will be) fixed. For instance, the bug affecting versions 1.1 and 1.2 might be fixed in version 2.0. Note that issues which do not have a **Fix Version** are classified as 'Unscheduled', as shown in the screenshot above.

Versions can be in one of three states: **Released, Unreleased** or **Archived.** Versions can also have a *Release Date,* and will automatically be highlighted as 'overdue' if the version is Unreleased when this date passes.

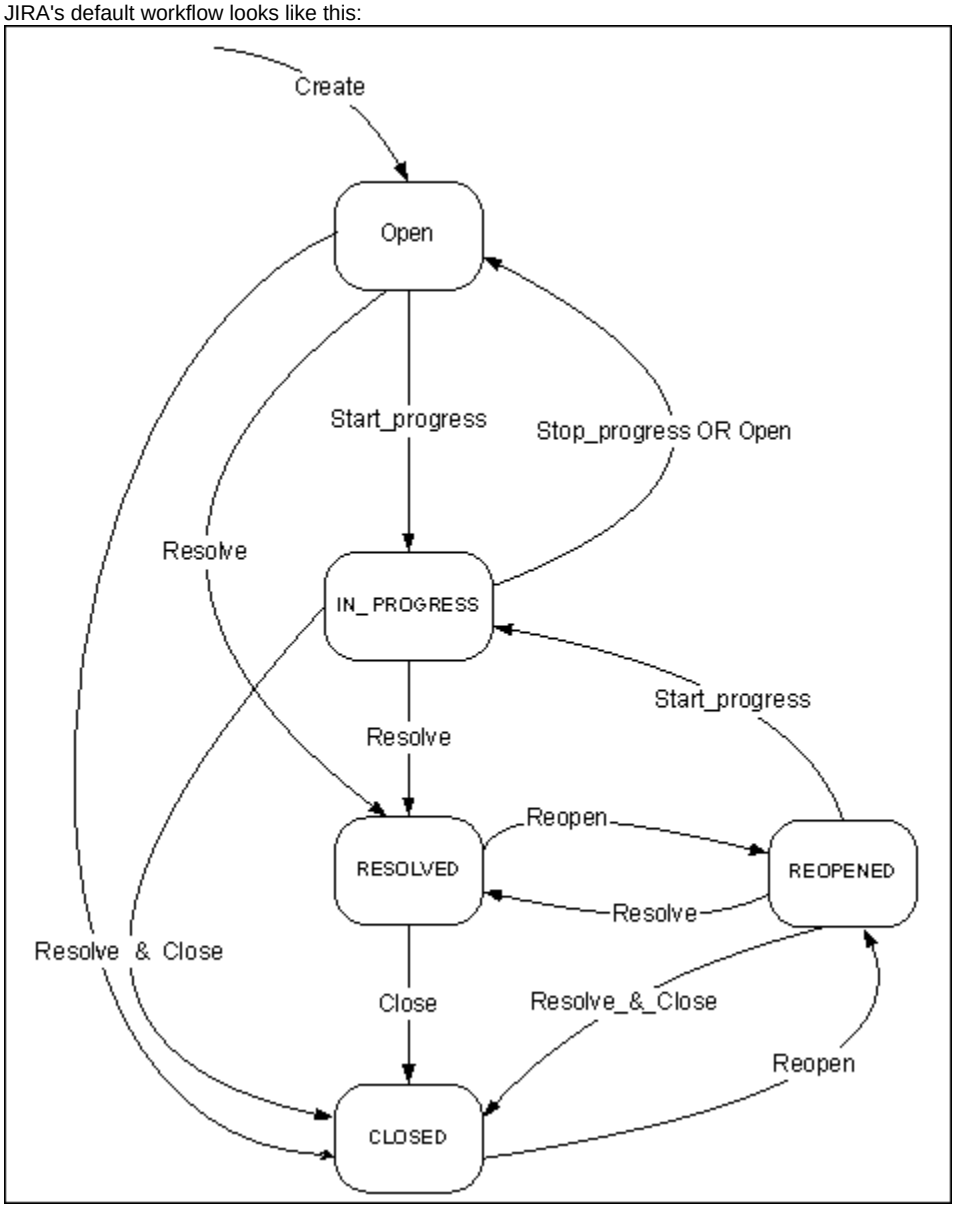
Additional Resources

- See 'Browsing a Project' for information on looking up a project's structure and issues.
- See the *JIRA Administrator's Guide* for information on defining projects, components and versions.
- Watch the 'Adding a Project' tutorial video to see how a project is added in JIRA. Please note the JIRA version and JIRA edition of the tutorial video before watching.

What is Workflow?

Workflow is the movement of an issue through various **Statuses** during its lifecycle.

JIRA's default workflow looks like this:



If you are using JIRA Professional or Enterprise edition, JIRA's workflow may be customised by your JIRA administrator.

Getting Started

The following pages contain information to help you get started using JIRA:

- Logging into JIRA
- Exploring the JIRA Workspace
- Using Keyboard Shortcuts

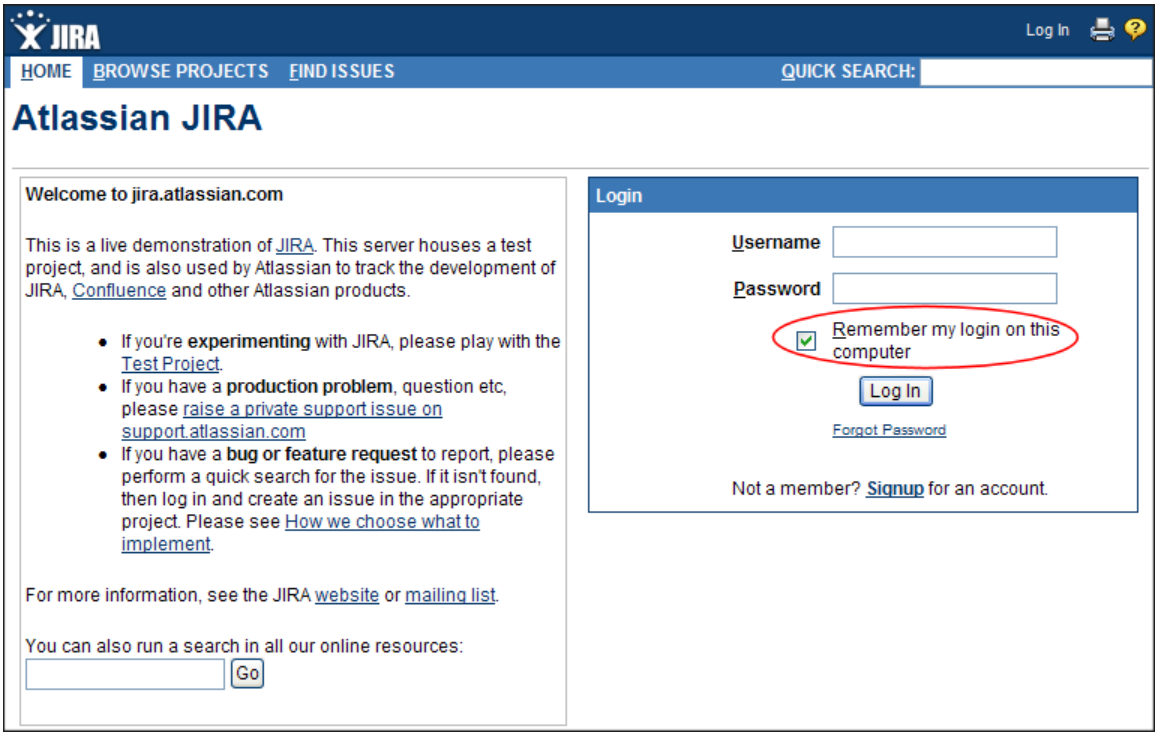
Logging into JIRA

Many JIRA instances will have permissions implemented that restrict issues and issue actions to certain users and user groups. Some JIRA instances may not permit anonymous access altogether. In these scenarios, you will be prompted to login to JIRA.

The **Login** panel will display on the System Dashboard if you haven't logged into JIRA. It has three functions:

1. **Logging into JIRA:** To login to JIRA, enter your **'Username'** and **'Password'** and click the **'Log In'** button.
   *Checking the **'Remember my login on this computer'** checkbox will prevent you from being automatically logged out of JIRA. However, your session will not be preserved, e.g. last search, current project, etc.*
2. **Resetting your password:** To reset your password, click the **'Forgot Password'** link. The reset password page will display. Enter your **'Username'** and a new password will be emailed to the email address specified in your JIRA user profile. If you have forgotten your email address, you will need to contact your JIRA administrator for help.
3. **Signing up for an account:** If you do not have a user account and the JIRA administrator has enabled public signup, you can create your own user account by clicking the **'Signup'** link in the 'Not a member? Signup for an account' text. Enter your details and click the **'Sign up'** button to create your account.

Please note that your JIRA screen may look different from this screenshot, as the dashboard and colours may have been customised by your JIRA administrator. The links, however, will be the same.

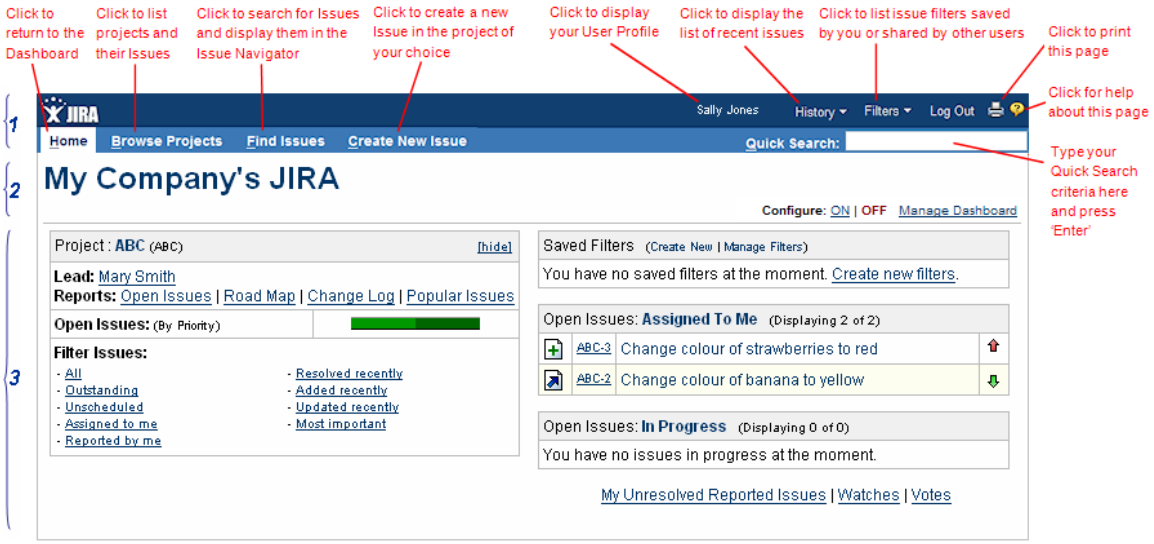


Exploring the JIRA Workspace

The **Dashboard** is the first page you see when you login to JIRA. It has three areas:

1. The navigation bar (at the top of the screen) — this is the same on every page in JIRA. It contains links which give you quick access to many of JIRA's most useful functions.
2. The name of your JIRA system (e.g. 'My Company's JIRA') — this area can also contain a 'welcome' message from your JIRA administrator.
3. The main area of the screen, below the top navigation bar — this area can be customised to display many different types of information, depending on your areas of interest. For details, please see Customising the Dashboard.

Please note that your JIRA screen may look different from this screenshot, as the logo and colours may have been customised by your JIRA administrator. The links, however, will be the same.



You can return to the dashboard from anywhere in JIRA by clicking the **'Home'** link on the top navigation bar.

Using Keyboard Shortcuts

Keyboard shortcuts provide a quick and easy way of navigating though JIRA without having to take your fingers off the keyboard.

Activating Shortcuts

Shortcuts are activated by simultaneously pressing the modifier key and the specific access key. For example, pressing 'Alt + C' in Firefox (on Windows) will create a new issue.

The access key is an identifying key for a particular page element. The access key is commonly identified by the underlined character on the element label.

Access Keys

The shortcuts that are currently available, and their default access keys, are:

Operation	Default Access Key	Description
Home	H	Redirects you to your Dashboard.
Browse Projects	B	Redirects you to the Browse Projects page.
Find Issues	F	Redirects you to the Issue Navigator.
Create New Issue	C	Redirects you to the Create Issue page.
Administration	A	Redirects you to the Administration page (if you have appropriate permission).
Quick Search	Q	Directs your cursor to the Quick Search text box in the top right-hand corner.
Comment	M	On the View Issue screen, opens and focuses on the comment box.
Edit	E	On the View Issue screen, edits the issue (if you have appropriate permission).
Next	N	On the View Issue screen (if you got there via the Issue Navigator), navigates to the next issue.
Previous	P	On the View Issue screen (if you got there via the Issue Navigator), navigates to the previous issue.
Submit	S	Submits any form.
Cancel	`(backquote)	Cancels any form.

Remember Me	R	Automatically checks the 'Remember me' checkbox on the Login page.

Modifier Keys

The modifier key will differ with each operating system and browser. For example, when running Firefox on Mac OSX you will need to type 'Ctrl' + 'S' to submit a form, while on Windows you will need to type 'Alt' + 'S'. The following table identifies the modifier keys for the various combinations:

Browser	Mac OSX	Windows	UNIX/Linux
Internet Explorer	Ctrl	Alt	Alt
Mozilla Firefox 2	Ctrl	Alt + Shift	Alt + Shift
Mozilla Firefox 1	Ctrl	Alt	Alt
Opera	Shift + Esc	Shift + Esc	Shift + Esc
Safari	Ctrl	Ctrl	Ctrl

In Internet Explorer, **links** will only be highlighted by shortcut keys; you will need to press Enter to proceed. Buttons, however, are activated through the shortcut.

In Opera, **buttons** will only be highlighted by shortcut keys; you will need to press Enter to proceed.

Working with Issues

The following pages contain information on working with issues:

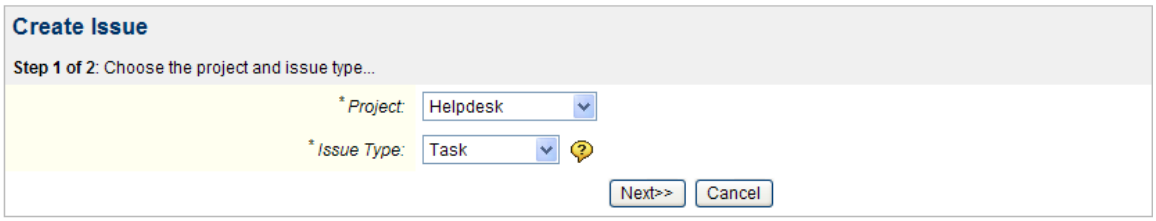
- Creating an Issue
- Setting Security on an Issue
- Attaching a Screenshot
- Editing Rich-Text Fields
- Scheduling an Issue
- Logging Work on an Issue
- Creating a Sub-Task
- Watching and Voting on an Issue
- Commenting on an Issue
- Linking Issues
- Cloning an Issue
- Moving an Issue
- Viewing an Issue's FishEye Changesets
- Viewing the Bamboo Builds related to an Issue
- Modifying Multiple ('Bulk') Issues

Creating an Issue

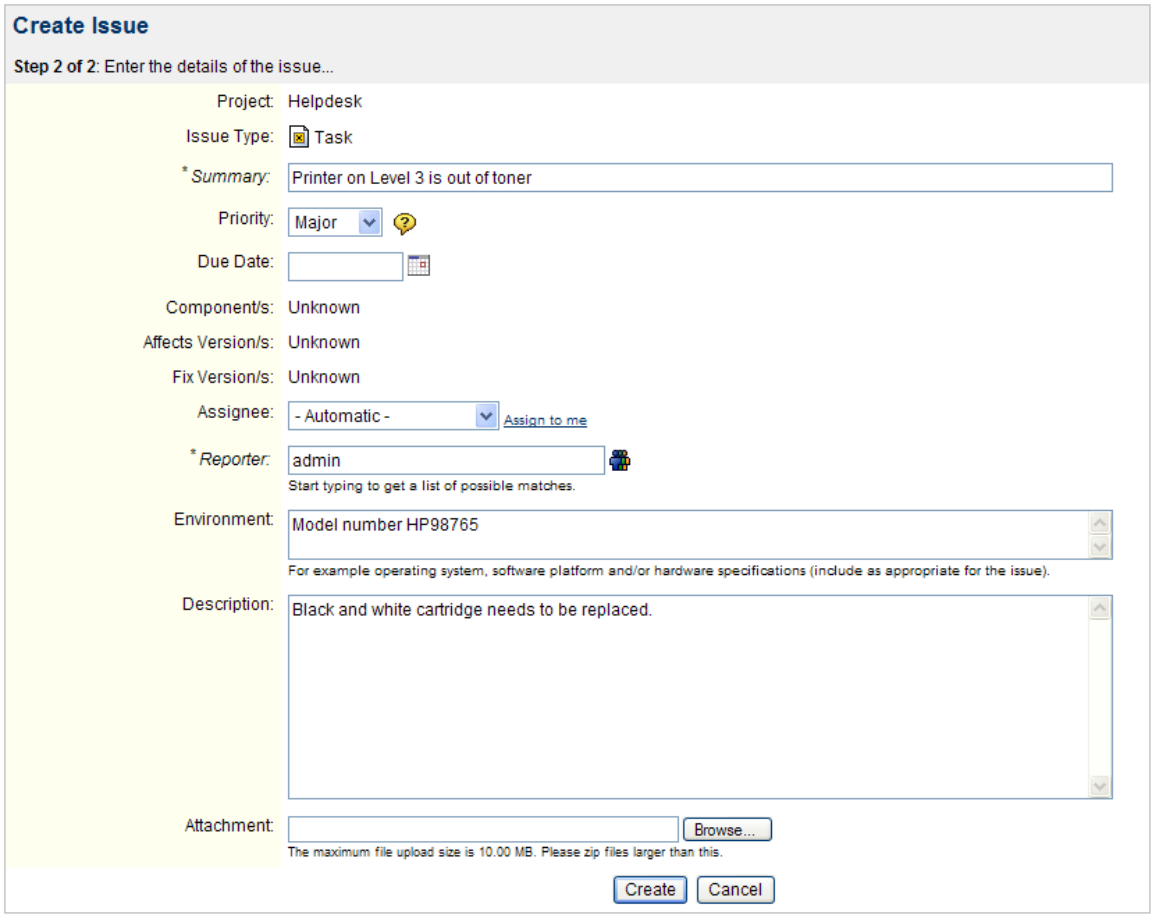
To create a JIRA issue, you will need to have the *Create Issue permission* in the relevant project. If you don't have this permission, please contact your JIRA administrator.

To create a new JIRA issue:

1. Click the '**Create Issue**' link at the top of the screen.
2. The 'Step 1. Choose the project and issue type' screen will be displayed. Select the relevant project and issue type, then click the '**Next**' button. Note, this page will not display if the project and issue type can be defaulted, i.e.
    - There is only one project and one issue type for that project.
    - If you click '**Create Issue**' while browsing a project, and there is only one issue type for that project.



3. The 'Step 2. Enter the details of the issue' screen will be displayed. Type a summary of the issue and complete any other required fields, which are italicised and highlighted by an asterix.



Note that this screen may look different if your JIRA administrator has customised it.

4. Click the '**Create**' button at the bottom of the page. The new issue will be created and you will see the 'View Issue' screen, showing the issue details that you have provided. You may also receive an email containing details and a link to your new issue.

To see a list of all issues that you have created, which have not yet been resolved, go to the Dashboard and click the 'My Unresolved Reported Issues' link.

With appropriate configuration by your JIRA administrator, it is also possible to create issues via email.

Setting Security on an Issue

Setting the security level on an issue restricts the access of that issue to only people who are a member of that security level. If you are not a member of that security level then you cannot access that issue and it will not appear in any filters, queries or statistics.

The security level of an issue can be set either when the issue is being created or afterwards when the issue is being edited.

Issue security is only available in *JIRA Enterprise Edition.*

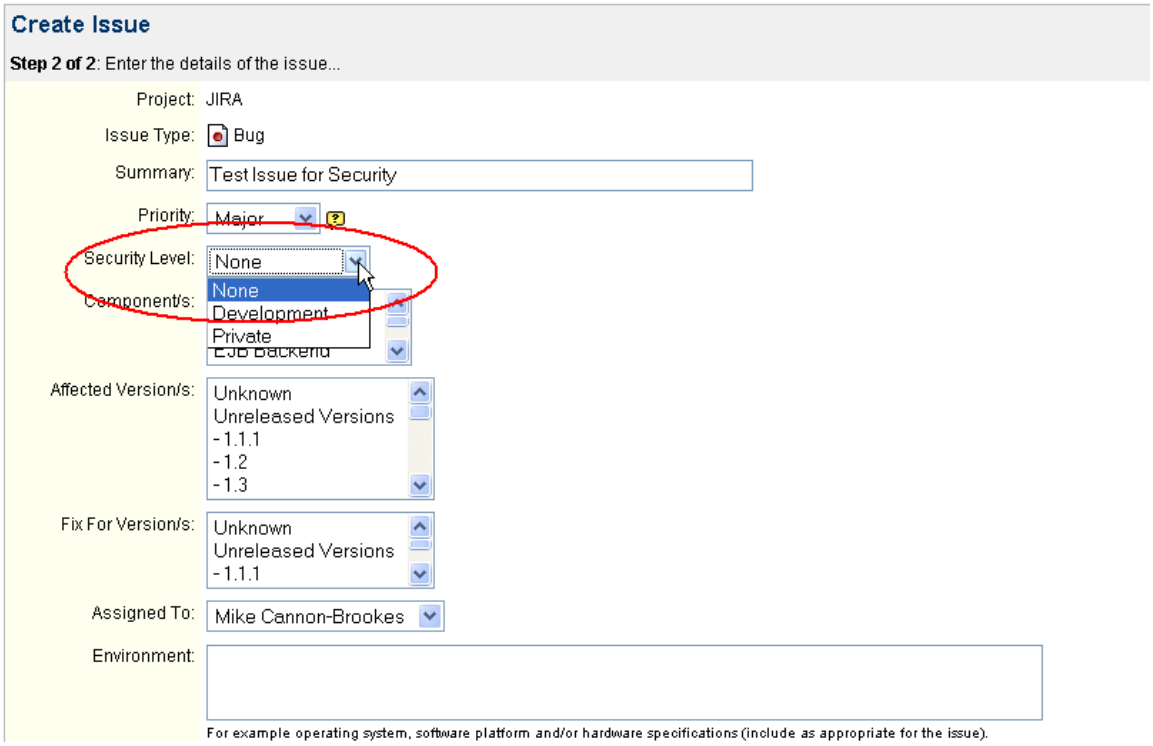
Permissions

To be able to set the Issue Level Security for an issue the user must have the 'Set Issue Security' permission. This is set up by the administrator for the current project as explained in the Permissions section.

Setting Security on an Issue

A person can only set the security on an issue to a level that they are a member of. This prevents the issue from being set to a level that nobody is a member of and effectively becoming "lost"

If the person creating/editing the issue has the permission 'Set Issue Security' then they will be presented with a drop down of all the security levels that they are a member of. Selecting one of these levels sets the security of the issue to this level. The issue will then only be accessible to members of that security level



N.B. If the user does not have the 'Set Issue Security' permission then the default Issue Level Security will be used. This may mean that the issue created is not visible to the person that created it (Issue Level Security should be configured so this does not happen).

Attaching a Screenshot

JIRA allows you to attach screenshots you have captured to an issue, if the administrator has configured JIRA and your permissions appropriately.

> To be able to attach screenshots, your JIRA administrator must have enabled file attachments. You will also need the 'Create Attachments' permission in the appropriate projects. Otherwise, the link to attach screenshots will not appear in your issues.

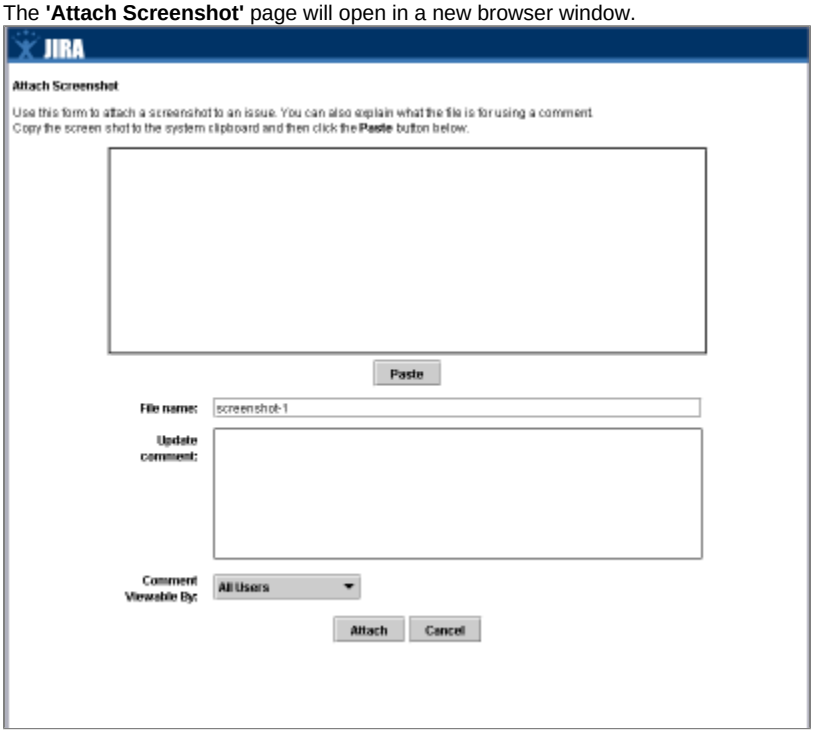
> The functionality to attach screenshots is currently only available on Windows and Mac clients.

Attaching a Screenshot

To attach a screenshot:

1. Open the JIRA issue that you wish to attach a screenshot to. Click the **'Attach screenshot to this issue'** link in the **'Operations'** menu.

2. The **'Attach Screenshot'** page will open in a new browser window.



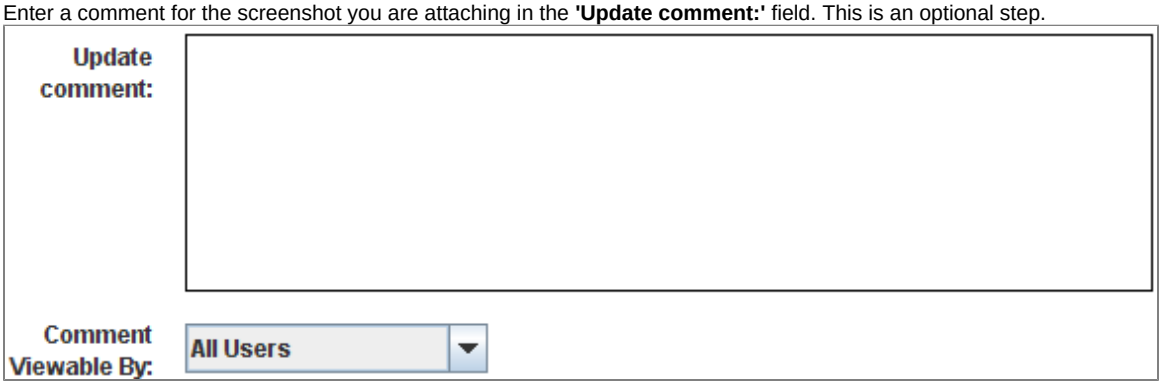
If this is the first time you have used this function, a security warning will also display in a dialog box asking you whether you want to trust the applet or not.

Choose the **'Yes'** option to trust the applet and access the **'Attach Screenshot'** page.

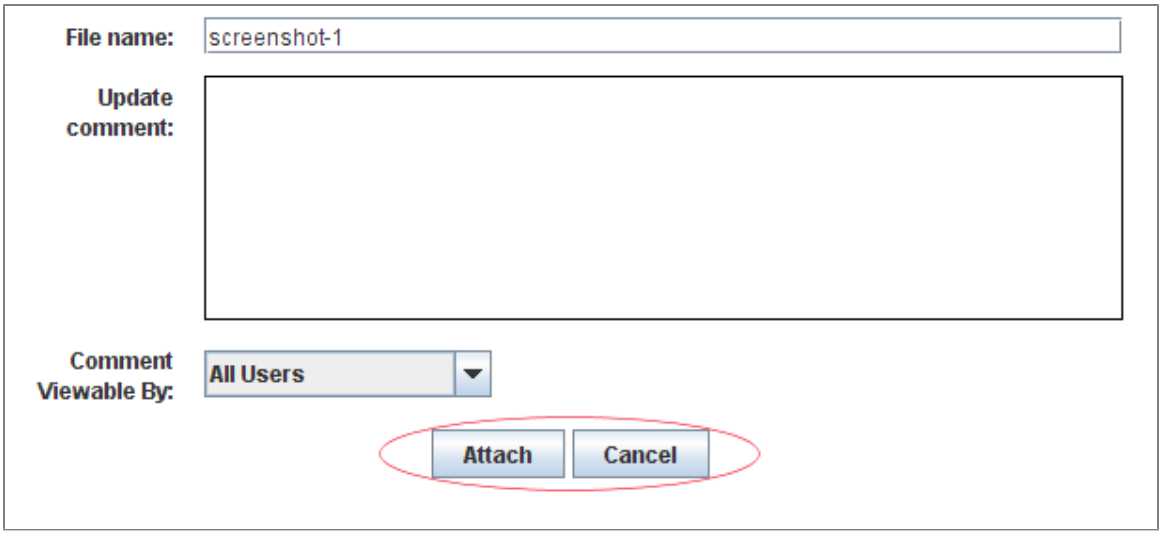
> JIRA uses an applet to run the **'Attach Screenshot'** functionality in the new browser window.

3. Ensure that you have captured an image to your operating system's clipboard and click the **'Paste'** button to paste the image. Your captured image should display in the blank area above the **'Paste'** button.
   !screenshotapplet-paste.png|title="Screen shot of JIRA screenshot applet paste control"!Please note that clicking the **'Attach'** button before an image has been pasted will not attach anything to the JIRA issue.
4. Enter a file name for the screenshot you are attaching in the **'File name:'** field. The file name will be initially defaulted to '`screenshot-1`'.

   A valid file name cannot contain any of these characters: `'\', '/','\"', '%', ':', '$', '?', '*'`.
   If an invalid file name is entered, an error message will display when the **'Attach'** button is clicked and the screenshot will not be attached to the JIRA issue.
5. Enter a comment for the screenshot you are attaching in the **'Update comment:'** field. This is an optional step.

   If you have entered a comment, then you can also set the security level for the comment by selecting the appropriate value from the **'Comment Viewable By:'** dropdown. The security level for the comment will initially be defaulted to '`All Users`'.
   The comment entered will be added to the JIRA issue, with the selected security level, when the screenshot is attached.
6. Click the **'Attach'** button to attach the captured image to your JIRA issue.
   The window will close and you will be returned to your original JIRA issue.
   At any time you may click the **'Cancel'** button to close the window without attaching anything to the JIRA issue.



Capturing Screenshots

The method for capturing screenshots differs on each operating system, as described below:

Capturing a screenshot on Windows

- New screenshot capture — To capture a screenshot into the system clipboard, use either of the following keyboard combinations:
  - press **ALT-PRINTSCREEN** to capture your currently selected window; or
  - press **CTRL-ALT-PRINTSCREEN** to capture the whole desktop
- Existing image — Open your existing image in your favourite imaging application and select the copy option from the appropriate menu to capture the image into the system clipboard.

Capturing a screenshot on Mac OSX

- New screenshot capture — To capture a screenshot into the system clipboard, use either of the following keyboard combinations:
  - press **CTRL-APPLE-SHIFT-4** to capture your currently selected window; or
  - press **CTRL-APPLE-SHIFT-3** to capture the whole desktop
- Existing image — Open your existing image in your favourite imaging application and select the copy option from the appropriate menu to capture the image into the system clipboard.

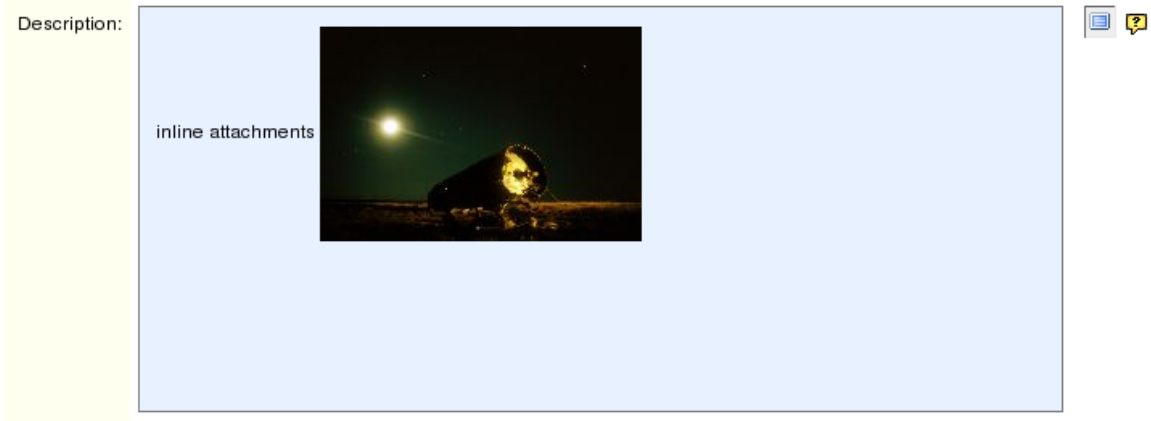
Editing Rich-Text Fields

When you create, edit or comment on a JIRA issue, some fields may display two small icons at the right of the text area: a blue screen (the **'Preview'** icon) and a yellow question-mark (the **'Help'** icon). The presence of these icons indicates that this field supports JIRA's Text Formatting Notation, allowing you to use rich-text features such as:

- Italic, bold, underlined text.
- Multiple levels of headings.
- Bullets, numbered lists, tables and quotations.
- Images.
- Macros (see below).

For example, to include an image in the field, you would first attach the image to the issue, then type the following into the field:

To preview what the field will look like after you save it, click the **'Preview'** icon:



Click the **'Help'** icon to see a popup window containing the Text Formatting Notation Help.

Using Macros

JIRA ships with the following macros:

Macro	Description	Enabled by default

Anchor Macro	Create an anchor that allows people to link to a specific point in a page. Usage:
	`{anchor:bookmark1}`
	`... text here ...`
	`[#bookmark1]`
		yes

Code Macro	Format blocks of source-code or XML. The default language is Java but you can specify JavaScript, ActionScript, XML, HTML and SQL too. Usage:
	`--- Java example ---`
	`{code:title=Bar.java|borderStyle=solid}`
	`// Some comments here`
	`public String getFoo()`
	`{`
	`  return foo;`
	`}`
	`{code}`
	`*--- XML example ---*`
	`{code:xml}`
	`<test>`
	`  <another tag="attribute"/>`
	`</test>`
	`{code}`
		yes

Quote Macro	Generate blockquotes that may contain multiple paragraphs or complex markup. Usage:
	`{quote}`
	`This is text from another source`
	`{quote}`
		yes

No Format Macro	Create blocks of text where other wiki formatting is not applied. Usage:
	`{noformat}`
	`This is text  \#&@!@(*!&*@()#*@\ \\`
	`{macros} _wont_work_here`
	`{noformat}`
		yes

Panel Macro	Draw a panel with the following optional parameters:
	• title: Title of the panel
	• borderStyle: The style of the border this panel uses (solid, dashed and other valid CSS border styles)
	• borderColor: The color of the border this panel uses
	• borderWidth: The width of the border this panel uses
	• bgColor: The background color of this panel
	• titleBGColor: The background color of the title section of this panel
	`{panel}Some text{panel}`
	`{panel:title=My Title}Some text with a title{panel}`
	`{panel:title=My Title| borderStyle=dashed| borderColor=#ccc|`
	`titleBGColor=#F7D6C1| bgColor=#FFFFCE}`
	`a block of text surrounded with a *panel*`
	`yet _another_ line`
	`{panel}`
		true

Colour Macro	Change the colour of the contained text. Usage:
	`{color:red}`
	`look ma, red text!`
	`{color}`
	You can use any of the sixteen standard HTML colours ('aqua','black','blue','fuschia','green','gray','lime', 'maroon','navy','olive','purple','red','silver','teal','white','yellow'), or any hexadecimal colour code (e.g. '#336699').
		yes

Lorem Ipsum Macro	Insert paragraphs of "lorem ipsum" space-filler text. Usage: `{loremipsum}`	yes

HTML Macro	Use HTML code within a Jira Issue. Usage:
	`{html}`
	`<p>You'll find a lot more in <A href="chapter2.html">chapter two</a>.`
	`See also this <a href="../images/forest.gif">map of the enchanted`
	`forest.</A></p>`
	`{html}`
		no

The JIRA Text Formatting Notation and macros will only be available if your JIRA administrator has configured the relevant renderers.

Scheduling an Issue

On this page:

- Enabling Issue Scheduling
- Scheduling An Issue
- Searching
  - Fixed Date Searchers
  - Relative Period Search
    - Due Date Popup
    - Relative Period Search Syntax

Enabling Issue Scheduling

By default JIRA ships with Issue Scheduling feature disabled. To enable Issue Scheduling at least one group or project role must be given the "Schedule Issues" permission. To assign this permission to one or more groups please refer to the Permissions documentation.

Scheduling An Issue

An issue can be scheduled when it is created, or at a later stage using the edit issue feature, by populating the "Due Date" field of the issue. Keep in mind that only users with "Schedule Issues" permission can populate the "Due Date" field.

Searching

It is possible to search for issues using the Issue Navigator. Choose "Find Issues" in the title bar menu to bring up the Issues Navigator. There are two ways to search for issues based on the "Due Date" field. The first way is using fixed date values, the second is using periods that are relative to the current date.

Fixed Date Searchers

There are two text fields in the Issue Navigator that allow searching based on the "Due Date" field. To perform a search for all issues that are due after a certain date enter the date into the "Due After" text field of the Issue Navigator. For example to find all issues that are due after 1st June 2003, enter 1-6-2003 into the "Due After" field.

It is also possible to use the Calendar popup to select a date by clicking the calendar icon on the right of the "Due After" text field. To search for issues that are due before a certain date, enter the date into the "Due Before" date. For example, to find all issues that are due before 1st July 2003, enter 1-7-2003 into the "Due Before" text field. It is also possible to search for issues that are due between two dates by populating the "Due After" and the "Due Before" text fields.

Relative Period Search

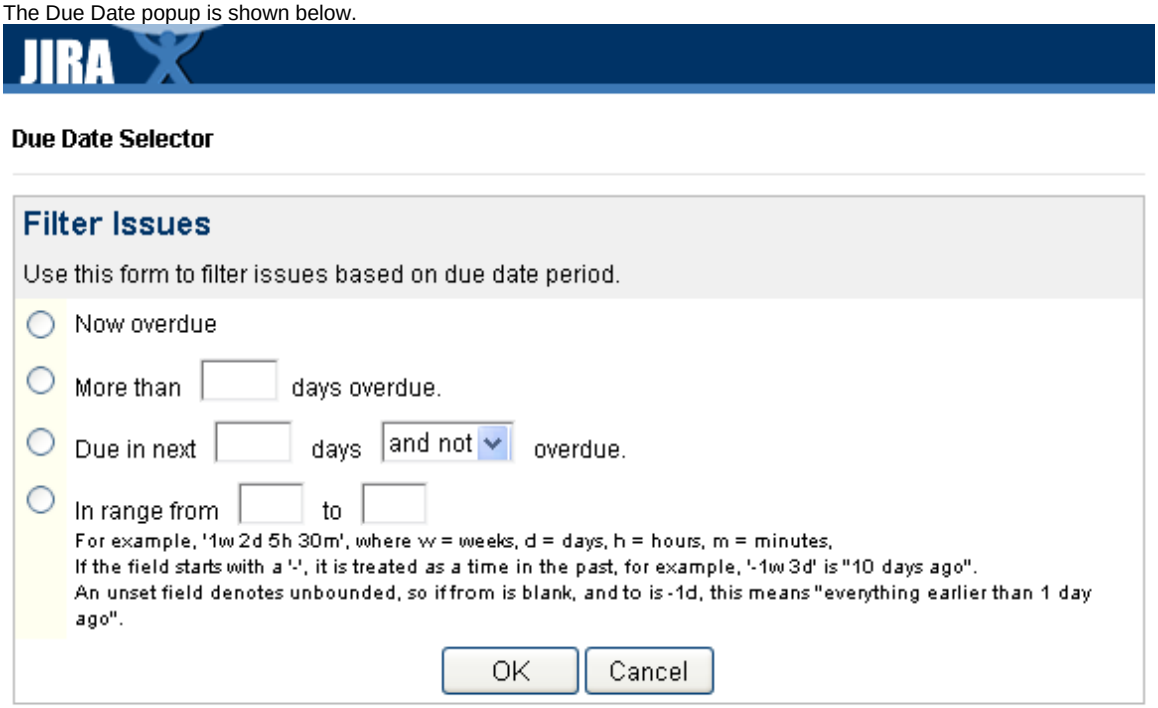
It is possible to perform a search that is relative to the time when it is run. These kind of searches are more useful when they are saved. For more instructions on how to save search filters please refer to saving search filters. For example, it is possible to do a search for issues that are due seven days from now. To do this, enter 7d into the "Due Date To" text field of the Issue Navigator. If the search is saved and run the next day, the issues that are due in seven days from the time that the search is run will be retrieved. Thus, this search will find all issues that are due within a week every time it is run.

The values that are entered into the "Due Date From" and "Due Date To" fields have to conform to a special syntax (described below). However it

is also possible to use the Due Date popup by clicking the icon to the right of the "Due Date To" text field to specify the search period.

Due Date Popup

The Due Date popup is shown below.



- To search for issues that are overdue at the time of the search select the first radio button and click '**OK**'.
- To search for issues that are overdue by more than a certain number of days, populate the text field in the second row, (select the second radio button, if it is not selected for you automatically) and click '**OK**'.
- To search for issues that are due in the next certain amount of days and are not overdue at the time of the search, populate the text field in the third row with the number of days, and choose '**and not**' from the select box in the third row. Select the third radio button, if one it was not selected automatically, and click '**OK**'.
- To search for issues that are due in the next certain amount of days and are overdue at the time of the search, populate the text field in the third row with the number of days, and choose '**and**' from the select box in the third row. Select the third radio button, if one it was not selected automatically, and click '**OK**'.
- The fourth row of the popup is used for arbitrary period searches. Use the '**to**' text field to specify the upper bound of the search, and the '**from**' text field to specify the lower bound of the search. A blank text field means no bound. Populating the text fields in the fourth row, actually has the same effect as populating the 'Due Date From' and 'Due Date To' text boxes in the Issue Navigator. The fields in the popup expect entries in the same syntax as the ones in the Issue Navigator. The syntax is described below

Relative Period Search Syntax

The "Due Date From" and "Due Date To" fields use a special syntax to denote time period bounds. The syntax uses numbers and abbreviations that follow the numbers to represent what the numbers actually mean. The abbreviations are "w" for weeks, "d" for days, "h" for hours and "m" for minutes. For example, to specify 10 days in the future use "10d" or "1w and 3d". To specify a period bound in the past prefix the value with the "-" sign. For example to specify 2 days, 4 hours and 3 minutes ago, use "-2d 4h 3m".

Logging Work on an Issue

On this page:

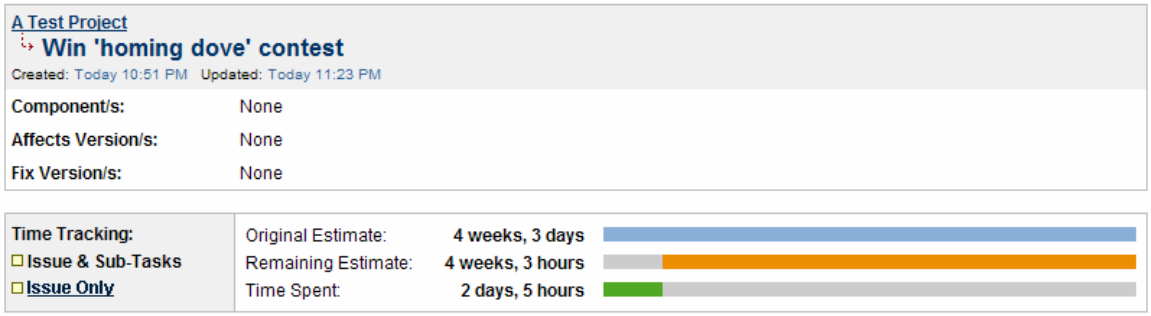
- About time-tracking
- Specifying an original time estimate
- Logging work on an issue
- Editing a work log entry
- Deleting a work log entry

About time-tracking

You can only log work on an issue if your administrator has enabled 'time-tracking', and if you have the correct permissions in the project to which the issue belongs.

If time-tracking has been enabled for your JIRA system, three coloured bars will be displayed for each issue, representing the following amounts of time:

- **Original Estimate** (blue) — the amount of time the issue was expected to take to resolve, when it was first created.
- **Remaining Estimate** (orange) — the remaining amount of time the issue is currently expected to take to resolve.
- **Time Spent** (green) — the amount of time logged working on the issue so far.



You can:

- Click **'Issue & Sub-Tasks'** to see the aggregate times for the issue plus all of its sub-tasks.
- Click **'Issue Only'** to see the times for the issue only.

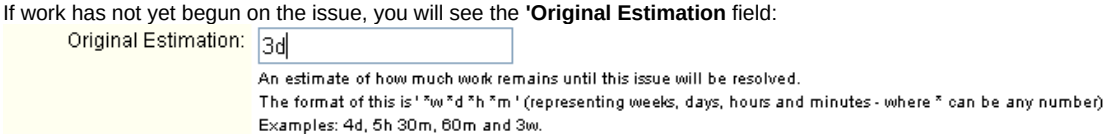
When you *log work* on an issue (see below), you:

1. Log the time you have spent, in weeks/days/hours/minutes (this will be added to the issue's total **'Time Spent')**
2. Enter a description of the work you have done
3. Adjust the **'Remaining Estimate'** (i.e. the *remaining amount of time* you think the issue will take to resolve)

Specifying an original time estimate

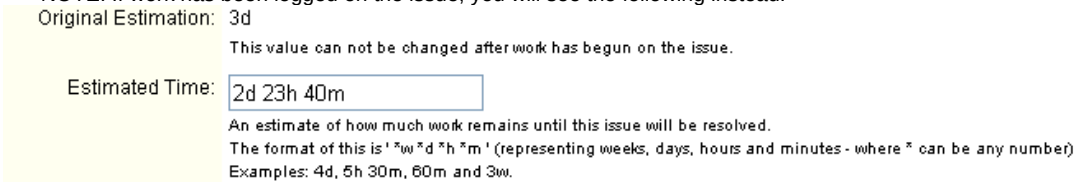
If the **'Original Estimation'** (i.e. the *total amount of time* you think the issue will take to resolve) has not yet been specified for the issue, you may want to specify it before you log work on the issue. Once work has been logged on an issue, the **'Original Estimation'** cannot be changed.

1. Navigate to the issue and view its details.
2. Select **'Edit'** from the **'Operations'** menu on the left-hand side of the screen.
3. If work has not yet begun on the issue, you will see the **'Original Estimation** field:



(You may edit this value up until work is logged against the issue.)

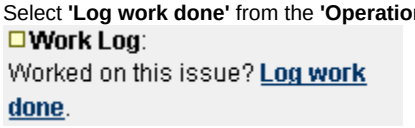
NOTE: If work has been logged on the issue, you will see the following instead:



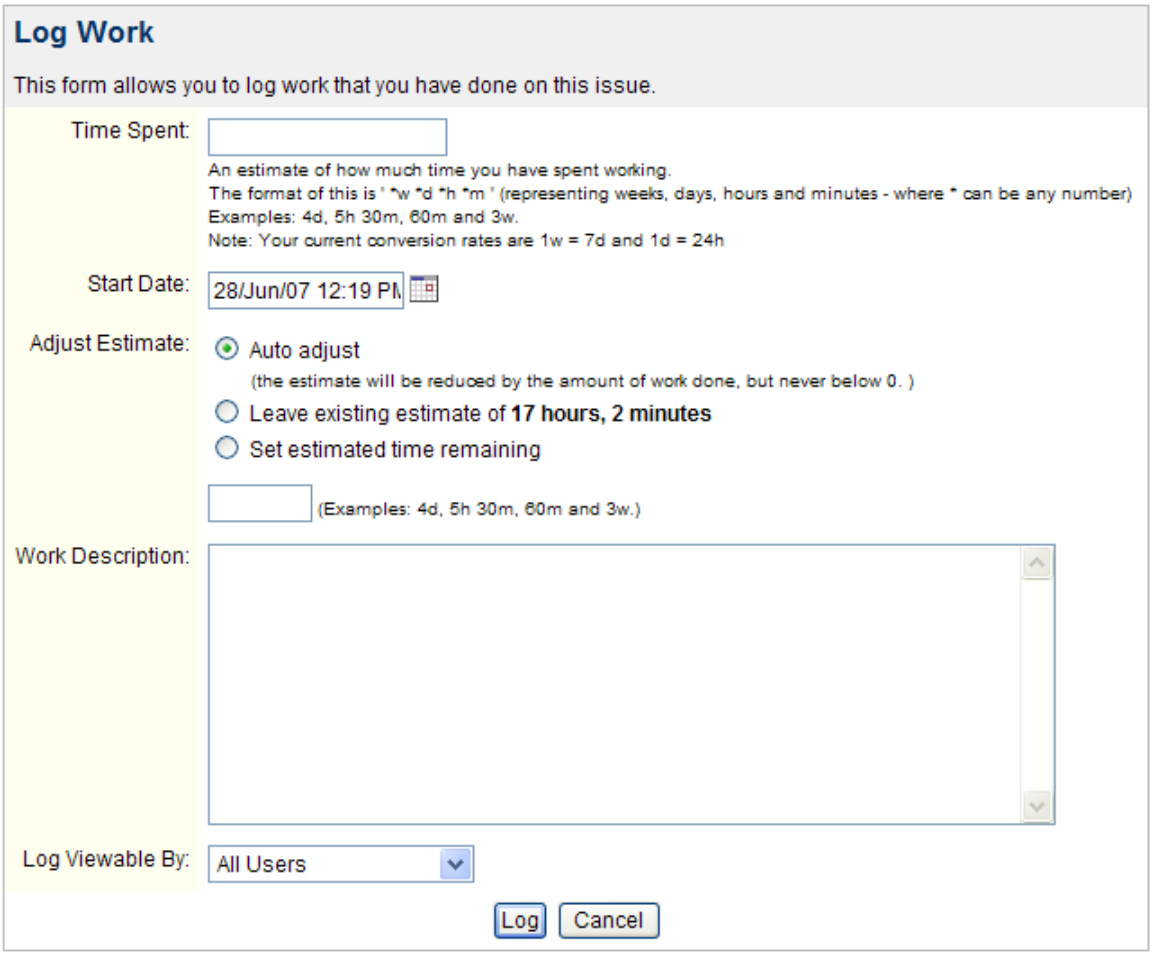
4. In the **'Original Estimation'** field, enter the amount of time you believe will be required to resolve the issue. Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes. For example, to specify 'six hours', type '6h'.
5. Click the **'Update'** button at the bottom of the screen.

Logging work on an issue

1. Navigate to the issue and view its details.
2. Select **'Log work done'** from the **'Operations'** menu on the left-hand side of the screen:



3. The **'Log work'** screen will be displayed:



4. In the **'Time Spent'** field, enter the amount of time to be logged. Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes. For example, to log two hours of work, type '2h'.

   If you type a number without specifying a time unit (e.g. if you type '2' instead of '2h'), the default time unit as specified by your JIRA administrator will apply.
5. In the **'Start Date'** field, click the calendar icon to select the date/time when you started work. The calendar popup will be displayed, where you can:
   - scroll back ('<') or forward ('>') to choose a different date.
   - click the hour to increase it (or <Shift> click to decrease it).
   - click the minute to increase it (or <Shift> click to decrease it).
   - click 'am' / 'pm' to toggle between them.
6. The **'Adjust Estimate'** field affects the value of the issue's **'Remaining Estimate'.** Select one of the following:
   - **'Auto Adjust'** — Select this if you want to automatically subtract your **'Time Spent'** from the issue's current **'Remaining Estimate'.**
   - **'Leave existing estimate of ... '** — Select this if don't want to change the issue's **'Remaining Estimate'.**
   - **'Set estimated time remaining'** — Select this if you want to manually change the issue's **'Remaining Estimate'.** If you select this option, enter your new estimate into the blank field below. Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes. For example, to specify 'thirty minutes', type '30m'.
7. In the **'Work Description'** field, type a description or comment about the work you have done.
8. In the **'Log Viewable By'** field, you can either set this work log to be viewable only by members of a particular project role; or you can allow all users to view it.

   NOTE: For users to view a work log, they must have the **'Browse Project'** permission to view the issue, as well as being a member of the **'Log Viewable By'** users.
9. Click the **'Log'** button to return to the issue, and verify that:
   - the **'Time Spent'** that you just entered has been added to the issue's total **'Time Spent'** field.
   - the **'Remaining Estimate'** that you just entered (or chose) matches the issue's **'Remaining Estimate'** field.

Note: you can log work on an issue if you have been granted the ' Work On Issues ' permission.

Once work has been logged on an issue, various reports based on the time-tracking information become available.

Editing a work log entry

1. Navigate to the issue and view its details.
2. Select the **'Work Log'** tab (below the **'Description'** field).
3. Locate the work log entry you wish to edit.
4. Click the **'Edit'** link, located at the right of the work log entry.
5. The **'Log work'** screen will be displayed. Edit the fields as described under *Logging work on an issue* (above).
6. Click the **'Log'** button to return to the issue, and verify that:
    - the word 'Edited' is displayed to indicate that the work log entry has been edited. You can hover your mouse over the word 'Edited' to see who edited the work log, and when.
    - the issue's total **'Time Spent'** field has been adjusted as per the **'Time Spent'** that you just edited.
    - the issue's **'Remaining Estimate'** field has been adjusted as per the **'Remaining Estimate'** that you just edited.

Note: you can edit your own work log entries if you have been granted the 'Edit Own Work Logs' permission. You can edit other people's work log entries if you have been granted the 'Edit All Work Logs' permission.

Deleting a work log entry

1. Navigate to the issue and view its details.
2. Select the **'Work Log'** tab (below the **'Description'** field).
3. Locate the work log entry you wish to edit.
4. Click the **'Delete'** link, located at the right of the work log entry.
5. Confirm the deletion by clicking the **'Delete'** button when prompted.
6. The **'Delete Worklog'** screen will be displayed. The **'Adjust Estimate'** field affects the value of the issue's **'Remaining Estimate'.** Select one of the following:
    - **'Auto Adjust'** — Select this if you want to automatically *add* the work log entry's **'Time Spent'** back to the issue's current **'Remaining Estimate'.**
    - **'Leave existing estimate of ... '** — Select this if don't want to change the issue's **'Remaining Estimate'.**
    - **'Set estimated time remaining'** — Select this if you want to manually change the issue's **'Remaining Estimate'.** If you select this option, enter your new estimate into the blank field below. Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes. For example, to specify 'thirty minutes', type '30m'.
7. Click the **'Delete'** button to confirm the deletion and return to the issue. Verify that:
    - the issue's **'Work Log'** tab no longer displays the work log entry that you just deleted.
    - the issue's **'Change History'** tab displays the **'Worklog Id'** (but not the description) of the deleted work log entry.
    - the issue's **'Time Spent'** field has been decreased by the value of the deleted work log entry's **'Time Spent'.**
    - the issue's **'Remaining Estimate'** field has been adjusted according to your choice in Step 6 (above).

Note: you can delete your own work log entries if you have been granted the 'Delete Own Work Logs' permission. You can delete other people's work log entries if you have been granted the 'Delete All Work Logs ' permission.

Creating a Sub-Task

Sub-task issues are generally used to split up a parent issue into a number of tasks which can be assigned and tracked separately. Splitting issues into smaller tasks often provides a better picture of the progress on the issue, and allows each person involved in resolving the issue to better understand what part of the process they are responsible for.

All the sub-tasks related to a parent issue are summarised on the parent issue's main screen (see "Working with Sub-Tasks" below). Sub-tasks always belong to the same project as their parent issue.

Sub-tasks have all the same fields as standard issues, e.g. Summary, Description, Reporter, Assignee, Status. Note that sub-tasks have a different set of issue types from the standard issue types.

Sub-tasks cannot have sub-tasks of their own. However, if you need to break up a sub-task into smaller sub-tasks, you could achieve this by first converting the sub-task to a standard issue (see below). You would then be able to create sub-tasks for it.

Note
Sub-tasks are only supported in the Professional and Enterprise editions of JIRA, and are only available if they have been enabled by your JIRA administrator. To create sub-tasks, you will also need to have the 'Create Issue' permission in the parent issue's project.

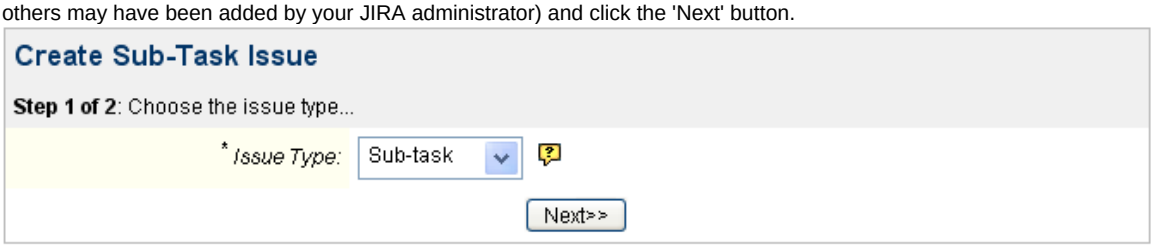
On this page:

- Creating a sub-task
- Working with sub-tasks
- Searching for sub-tasks
    - Adding the 'Sub-Tasks' column to your Issue Navigator
- Converting a standard issue to a sub-task
- Converting a sub-task to a standard issue

Creating a sub-task

1. Navigate to the issue which you would like to be the parent issue of the sub-task you are about to create.

2. Select 'Create sub-task' from the 'Operations' menu on the left-hand side of the screen.
3. The 'Step 1. Choose the project and issue type' screen will be displayed. Select the sub-task issue type (the default is 'Sub Task', but others may have been added by your JIRA administrator) and click the 'Next' button.

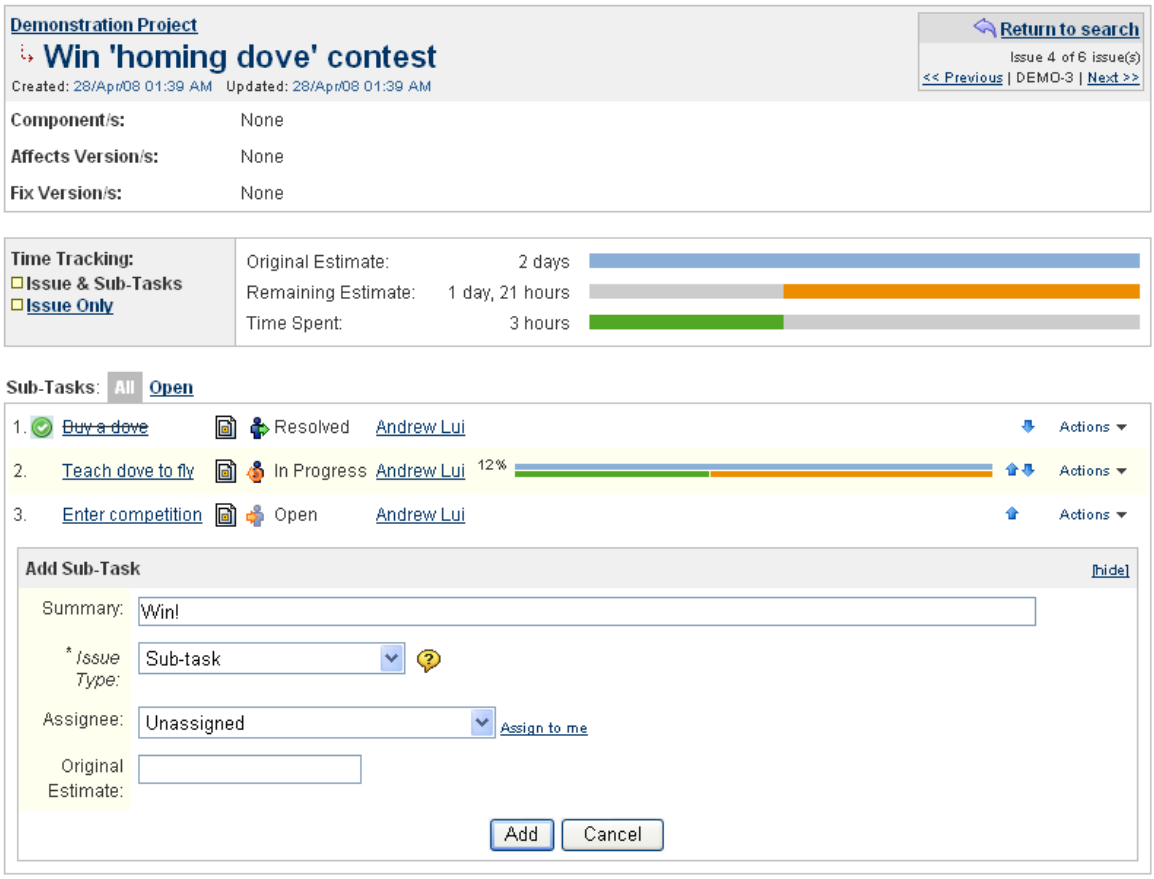


4. The 'Step 2. Enter the details of the issue' screen will be displayed, similar to creating a standard issue. Type a summary for the sub-task and complete any other required fields, which are italicised and highlighted by an asterix.
   Note that there is no option to set security on a sub-task, as sub-tasks inherit their parent issue's security levels (if any have been set).
5. Click the 'Create' button at the bottom of the page. The sub-task will be created and you will see the 'View Issue' screen, showing the sub-task's details that you have provided. The parent issue's summary and issue key are displayed above the sub-task's summary. You can easily return to the parent issue by clicking its summary or issue key.

In the above screenshot, the new sub-task is **DOVE-2 (Buy a dove)** and its parent issue is **DOVE-1 (Win 'homing dove' contest).** Both belong to the **Dove** project.

Working with sub-tasks

If an issue has sub-tasks, then the issue screen will show a list of all the issue's sub-tasks:



- The sub-task list has two *views:* 'All' and 'Open'. The *All* view lists all sub-tasks, regardless of status, while the *Open* view only shows sub-tasks that have not been resolved (i.e. do not have a Resolution). You can switch views by clicking the 'All' and 'Open' links.
- You can reorder sub-tasks by using the up and down arrows to, for example, organise the list in the order of intended execution or priority.
- You can perform *actions* on the sub-tasks by clicking the 'Actions' link in the right-most column, then selecting an option from the drop-down list, e.g. 'Resolve Issue', 'Close Issue', 'Reopen Issue'.
- Once an issue has one or more sub-tasks, you can quickly create additional sub-tasks by clicking the 'Add Sub-Task' link, typing a description in the 'Summary' field and clicking the 'Add' button. (The other sub-task fields are optional, although 'Issue Type' will default to 'Sub-task' and 'Assignee' will default to 'Unassigned'.)

Additionally, if your JIRA administrator has enabled time-tracking, coloured bars will be displayed showing the 'Original Estimate', 'Remaining Estimate' and 'Time Spent' for the issue and its sub-tasks.

Searching for sub-tasks

When sub-tasks are enabled, two extra entries will appear in the 'Issue Type' drop-down list in the Issue Navigator's search form. These entries are: 'Standard Issue Types' and 'Sub-Task Issue Types'.

- To search standard issues only, click the *Standard Issue Types* entry.
- To search sub-task issues only, click *Sub-Task Issue Types* entry.
- To search for one specific type of issue or sub-issue, select just one Issue Type or one Sub-Task Issue Type.

If no entries are selected from the *Issue Type* field then the search will return all the standard issues as well as sub-task issues that meet the rest of the search criteria.

The search results indicate sub-task issues by displaying the parent issue's issue key above the sub-task's summary, as shown below:

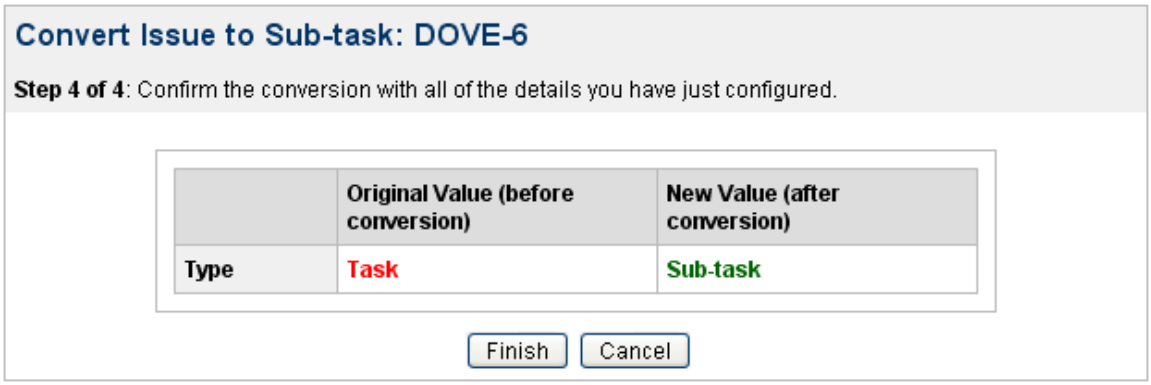
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Sub-Tasks
	DOVE-1	Win 'homing dove' contest	A. D. Ministrator	A. D. Ministrator		Open	UNRESOLVED	12/Oct/04	12/Oct/04	~~DOVE-2~~ ~~DOVE-3~~ DOVE-4 DOVE-5
	DOVE-2	DOVE-1 ↳ Buy a dove	A. D. Ministrator	A. D. Ministrator		Closed	Fixed	12/Oct/04	12/Oct/04	
	DOVE-3	DOVE-1 ↳ Teach dove to fly	A. D. Ministrator	A. D. Ministrator		Closed	Fixed	12/Oct/04	12/Oct/04	
	DOVE-4	DOVE-1 ↳ Enter competition	A. D. Ministrator	A. D. Ministrator		In Progress	UNRESOLVED	12/Oct/04	12/Oct/04	
	DOVE-5	DOVE-1 ↳ Win	A. D. Ministrator	A. D. Ministrator		Open	UNRESOLVED	12/Oct/04	12/Oct/04	

Adding the 'Sub-Tasks' column to your Issue Navigator

To add the 'Sub-Tasks' column to your Issue Navigator as shown in the above screenshot, please see Customising your Issue Navigator columns.

Converting a standard issue to a sub-task

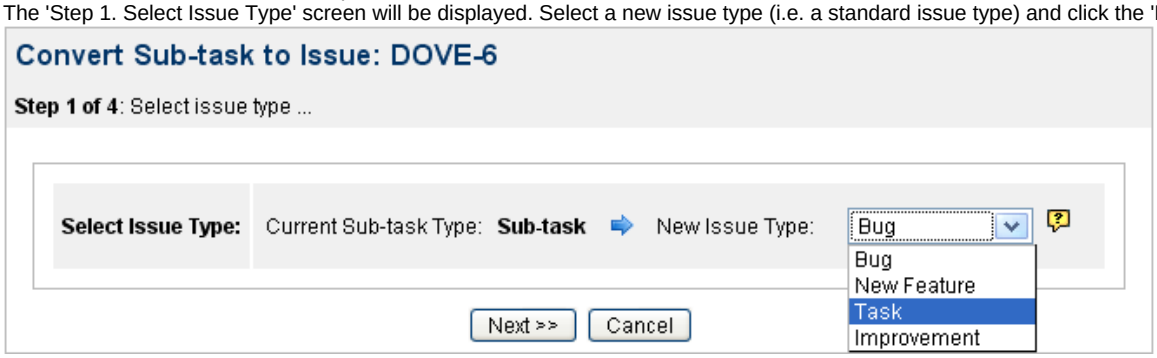
1. Navigate to the issue which you would like convert to a sub-task.
2. Select 'Convert to sub-task' from the 'Operations' menu on the left-hand side.
3. The 'Step 1. Select Parent Issue and Sub-Task Type' screen will be displayed:
   !subtasks-convertback1.png|title="Convert Issue to Subtask - Step 1. Select Issue Type"!Type or select the appropriate parent issue type, select the new issue type (i.e. a sub-task issue type) and click the 'Next' button.
4. If the issue's current status is not an allowed status for the new issue type, the 'Step 2. Select New Status' screen will be displayed. Select a new status and click the 'Next' button.
5. The 'Step 3. Update Fields' screen will be displayed. If the new issue type requires any additional fields, you will be prompted to enter them. Otherwise you will see the message 'All fields will be updated automatically':
   !subtasks-convertback3.png|title="Convert Issue to Subtask - Step 3. Update Fields"!Click the 'Next' button.
6. The 'Step 4. Confirmation' screen will be displayed. If you are satisfied with the new details for the issue, click the 'Finish' button:



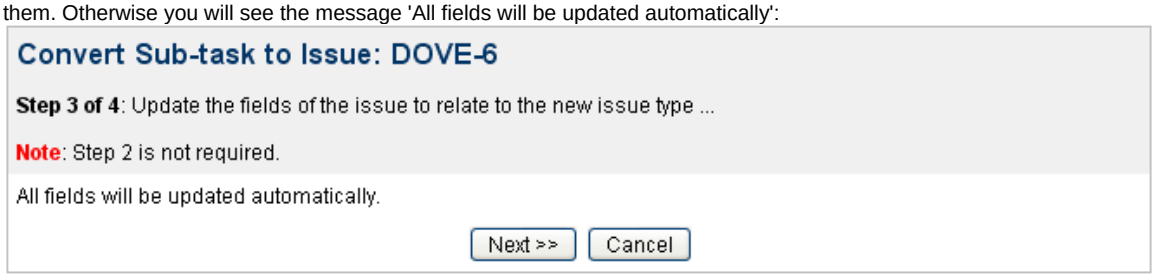
7. The issue will be displayed. You will see that it is now a sub-task, that is, its parent's issue number is now displayed at the top of the screen (see 'Creating a sub-task' above).

Converting a sub-task to a standard issue

1. Navigate to the sub-task issue which you would like convert to a standard issue.
2. Select 'Convert to issue' from the 'Operations' menu on the left-hand side of the screen.
3. The 'Step 1. Select Issue Type' screen will be displayed. Select a new issue type (i.e. a standard issue type) and click the 'Next' button.

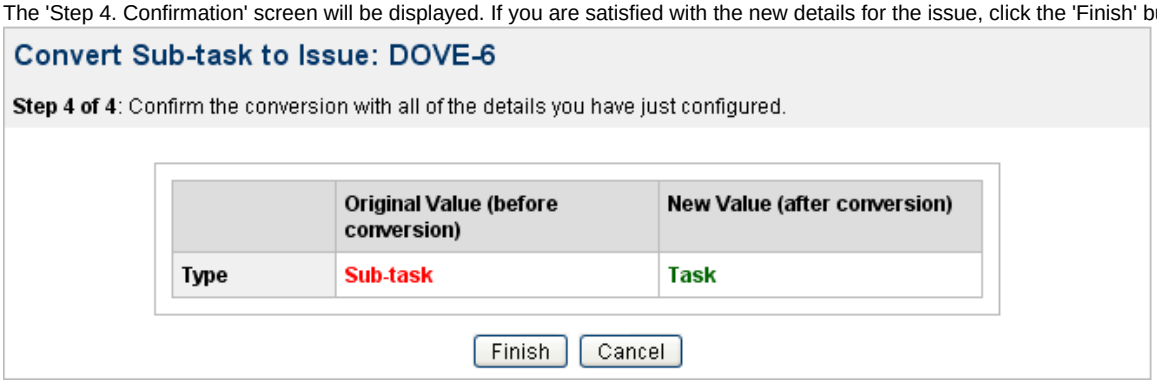


4. If the sub-task's current status is not an allowed status for the new issue type, the 'Step 2. Select New Status' screen will be displayed. Select a new status and click the 'Next' button.
5. The 'Step 3. Update Fields' screen will be displayed. If the new issue type requires any additional fields, you will be prompted to enter them. Otherwise you will see the message 'All fields will be updated automatically':



Click the 'Next' button.

6. The 'Step 4. Confirmation' screen will be displayed. If you are satisfied with the new details for the issue, click the 'Finish' button:



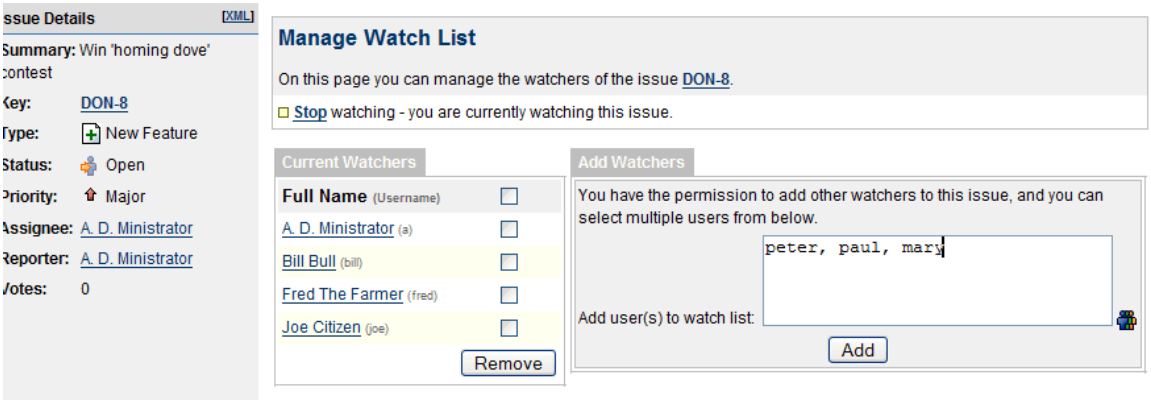
7. The issue will be displayed. You will see that it is no longer a sub-task, that is, there is no longer a parent issue number displayed at the top of the screen.

Watching and Voting on an Issue

JIRA allows a user to cast a vote for a particular issue — "voicing" their preference for that issue to be resolved or completed. JIRA also allows a user to watch a particular issue, signing up for notifications of any updates relating to that issue.

Building on these two features, JIRA provides the ability to view the voter and watcher lists for an issue. Further, a user with the correct permission can manage the watcher list.

The voter and watcher lists can be accessed by clicking the relevant link (beside the **"Votes"/"Watchers"** total) in the issue summary when viewing an issue.



Permissions

JIRA incorporates two permissions to govern who may view/edit the voter and watcher data:

- **View Voters and Watchers** — permits a user to view both the voter and watcher lists
- **Manage Watcher List** — permits a user to view/edit the watcher list

These permissions can be granted through a Permission Scheme.

> **Note**
> It is possible to add multiple users to the watcher list through the multi-user picker.

> **Note**
> It is not possible to edit the voter list.

Commenting on an Issue

Adding comments to an issue is a useful way to record additional detail about an issue, and collaborate with team members.

You can add comments to an issue as long as you have both of the following permissions:

- 'Browse Project' permission — to view the issue to be commented on
- 'Add Comments' permission — to add a comment to the issue

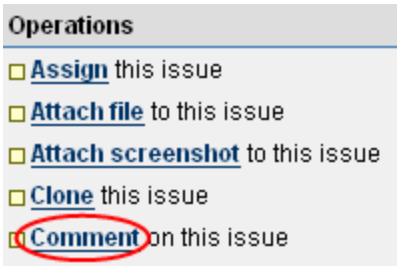
When adding a comment, you can also set the comment to be **Viewable By** members of a particular project role only; or you can allow all users to view it. For users to view a comment, they must have the 'Browse Project' permission to view the issue, and for each comment they must be a member of the **Viewable By** users (see 'Adding a Comment' below).

On this page:

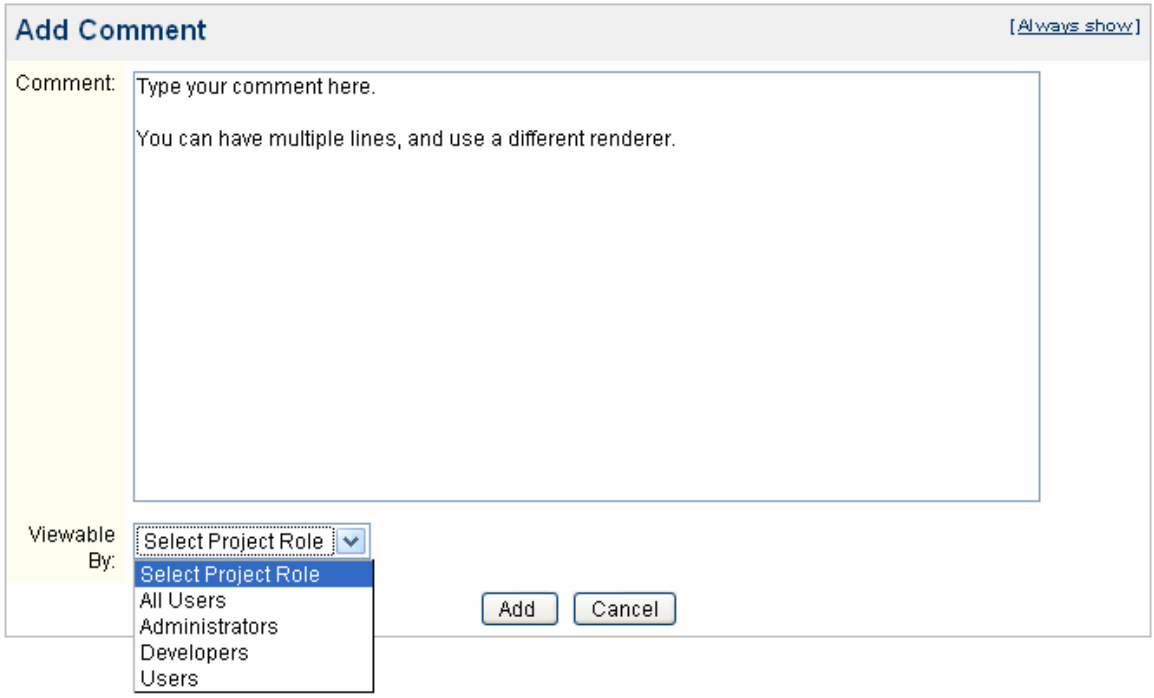
- Adding a comment
- Hiding/showing a comment
- Editing a comment
- Deleting a comment
- Linking to a comment

Adding a comment

1. Browse to the issue you wish to comment on.
2. Click the **Comment** link under the **Operations** menu.



3. Type your comment, and select which users can view this comment. Click the **Add** button.



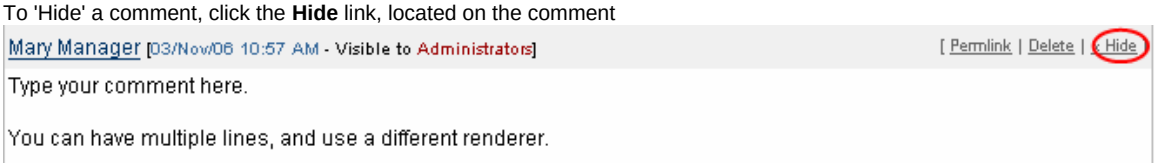
The **Viewable By** list will include all project roles[1] to which you belong. (Note: in this example, 'All users' means everybody who uses JIRA, while 'Users' means everybody who is a member of the 'Users' project role in this project.)
[1]Depending on how your JIRA administrator has configured 'Comment visibility' , the **Viewable By** list may include groups as well as project roles.

The comment body can be rendered differently if your JIRA administrator has set up a Renderer. For details, see 'Editing rich-text fields'.

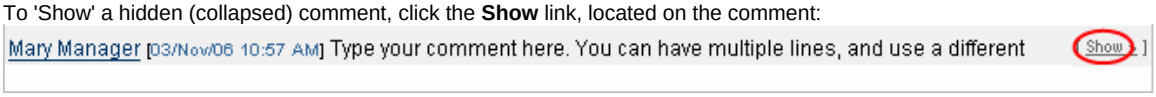
Hiding/showing a comment

Note that 'Hiding' or 'Showing' a comment refers to whether it is collapsed or fully visible (not to whether it is 'Viewable').

1. Browse to the comment you wish to hide/show.
2. To 'Hide' a comment, click the **Hide** link, located on the comment



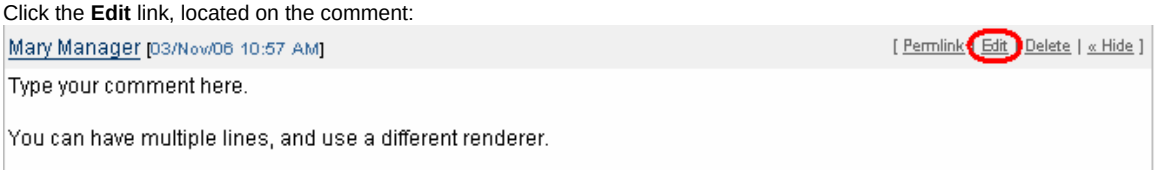
3. To 'Show' a hidden (collapsed) comment, click the **Show** link, located on the comment:



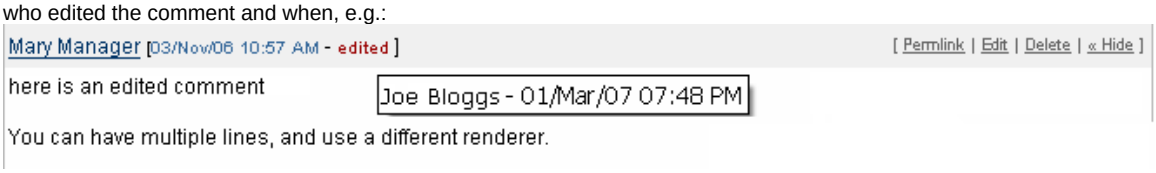
Editing a comment

Comments are editable if you are using JIRA version 3.8 or later. To edit a comment,

1. Browse to the comment you wish to edit.
2. Click the **Edit** link, located on the comment:



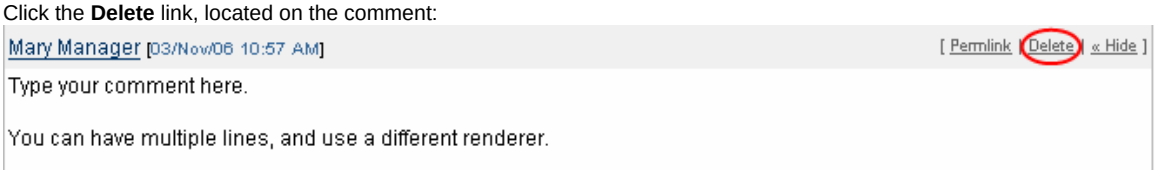
3. Edit the comment as required.
4. Click the **'Save'** button.
5. The word 'Edited' will be displayed to indicate that the comment has been edited. You can hover your mouse over the word 'Edited' to see who edited the comment and when, e.g.:



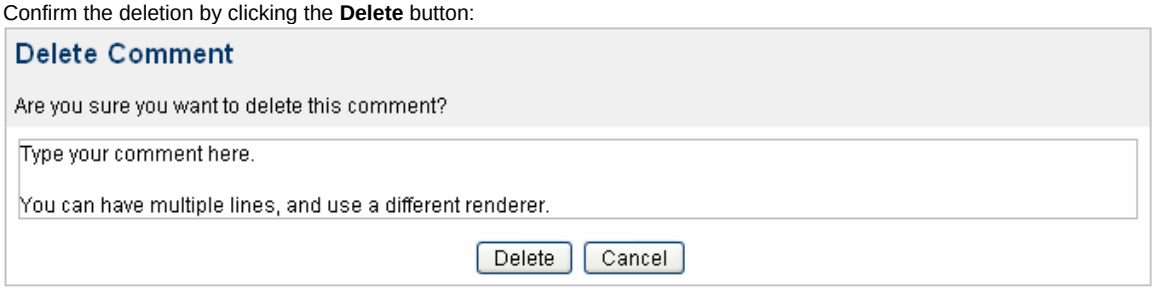
Note: you can edit your own comments if you have been granted the 'Edit Own Comments' permission. You can edit other people's comments if you have been granted the 'Edit All Comments' permission.

Deleting a comment

1. Browse to the comment you wish to delete.
2. Click the **Delete** link, located on the comment:



3. Confirm the deletion by clicking the **Delete** button:

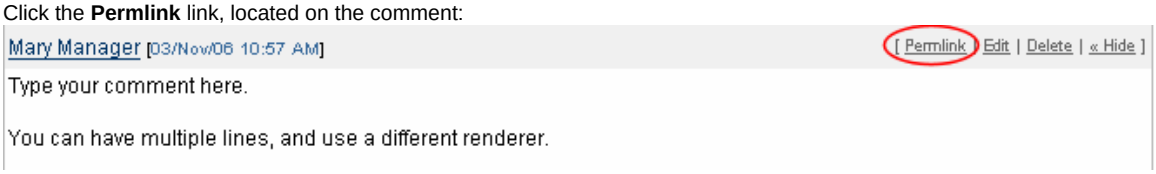


Note: you can delete your own comments if you have been granted the 'Delete Own Comments' permission. You can delete other people's comments if you have been granted the 'Delete All Comments' permission.

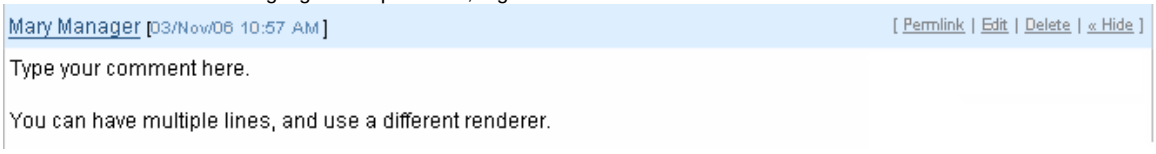
Linking to a comment

Sometimes you may want to link to a specific comment within a JIRA issue. To do this,

1. Browse to the comment you wish to link to.
2. Click the **Permlink** link, located on the comment:



3. The comment will now be highlighted in pale blue, e.g.:



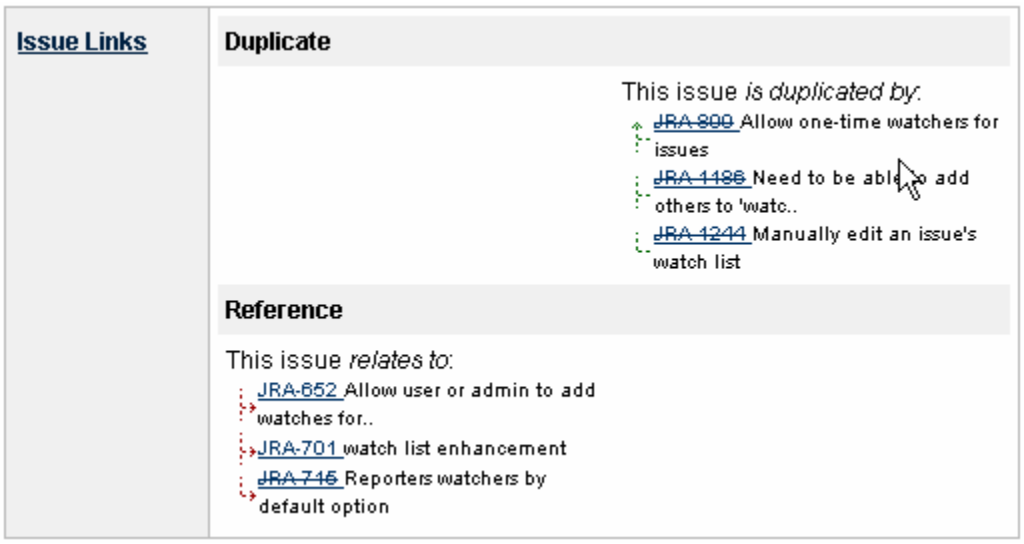
4. The URL in your browser's address bar will now look something like this:
   http://jira.atlassian.com/browse/TST-123?focusedCommentId=94796#action_94796
5. Copy the URL from your browser's address bar and paste it into wherever you want to link from (e.g. an email).

Linking Issues

Issue linking allows you to create an association between issues. For example:

- An issue may *duplicate* another.
- An issue may *incorporate* (be a superset of) another.
- An issue may *depend* on another.

Within an issue, links to other issues look like this:



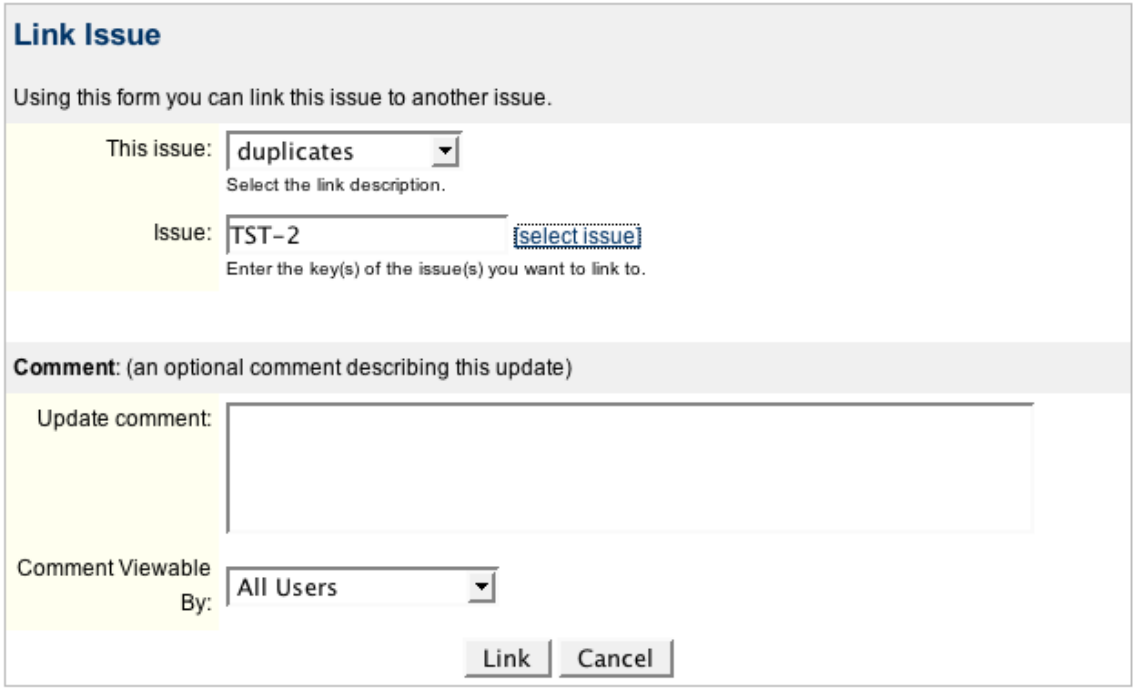
The strikethrough text indicates issues that have been resolved, e.g. in the above screenshot, issues JIRA-800, JRA-1186, JRA-1244 and JRA-715 have been resolved, while JRA-652 and JRA-701 have not.

> Linking is only available if it has been enabled by your JIRA administrator. To create links between issues, you will also need to have the *Link Issues permission* in the project(s) to which the issues belong.

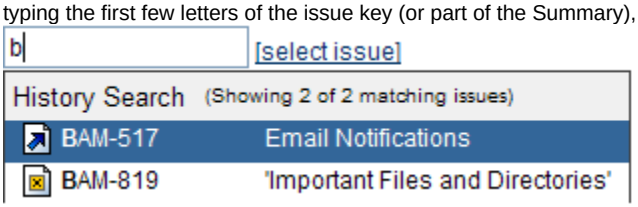
Creating a link

To create a link between issues:

1. View the issue from which you want to create a link.
2. Click the word '**Link**' under the **Operations** menu in the left-hand column of the screen. The 'Link Issue' form will be displayed. Select the type of link to be created, and the issue(s) to be linked to.

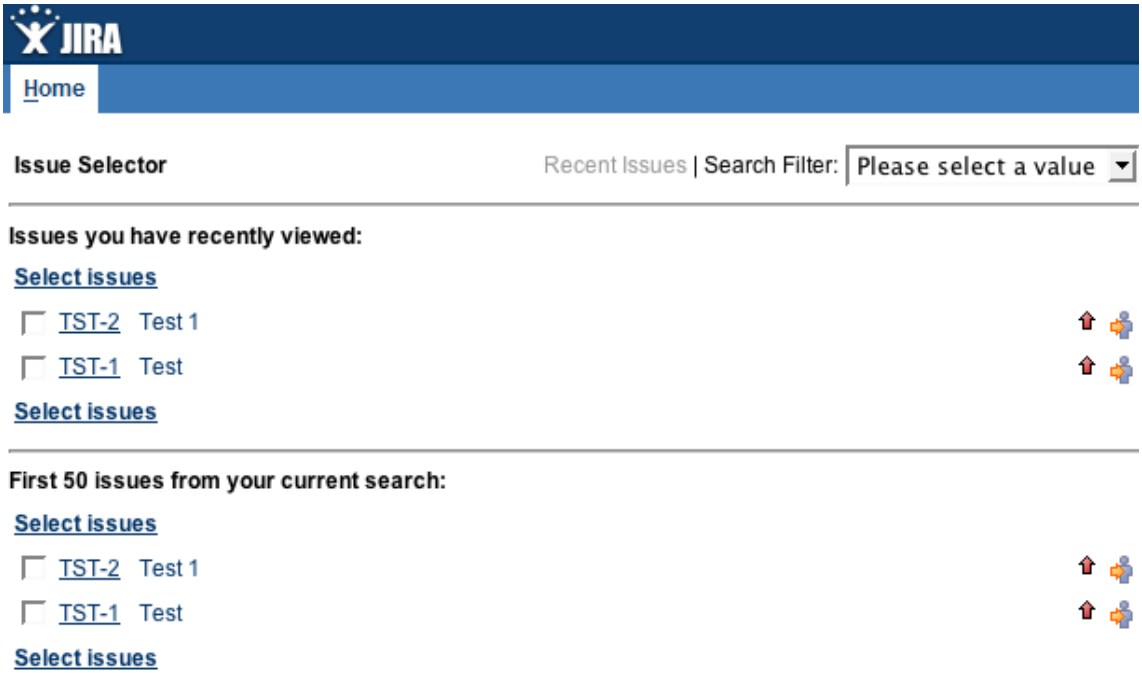


3. You can link to single or multiple issues. There are two ways to specify the issue(s):
    - Type the full issue key (e.g. **ABC-123)** - or a comma-separated list of issue keys, if linking multiple issues.
      NOTE: If you have previously browsed an issue, you can find the issue number by:
        - typing the first few letters of the issue key (or part of the Summary), which will display a drop-down list:

        - clicking the '**History**' dropdown, at the top right of the page and selecting the issue:

      or:
    - Click the **'Select Issue'** link to use the '**Issue Selector**' popup, which allows you to select either from issues recently viewed, or from issues returned from your chosen saved search filter (click the 'Please select a value' drop-down to choose a saved search filter).
        - To select a single issue, click the issue key.
        - To select multiple issues, click the '**Select multiple issues**' link. Checkboxes will appear, enabling multiple issues to be selected:



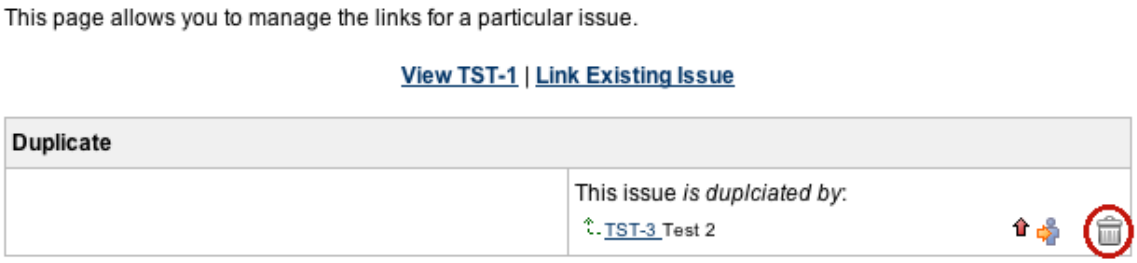
Select the issues, then click the '**Select issues**' link to close the popup and return to the 'Link Issue' form.

4. Click the '**Link**' button.

Deleting a link

To delete a link:

1. Inside a linked issue, click on the '**Manage Links**' link.
2. In the following screen, click on the small trashcan icon in the bottom-right corner.



Cloning an Issue

Clone Issue allows a user to create a duplicate of an issue within the same project. The clone issue is a replica of the original issue - containing the same information stored in the original issue - e.g. summary, affects versions, components, etc. The clone issue can also be linked to the original issue using a 'clone' link.

A clone issue is a separate entity from the original issue. Operations on the original issue have no affect on the clone issue and vice versa. The only connection takes the form of the link (if created) between the original and the clone issue.

A clone issue retains the following information:

- Summary - with optional prefix (specified in jira-application.properties)
- Description
- Assignee
- Environment
- Priority
- Issue Type
- Security
- Reporter
- Components
- Affects Versions
- Fix For Versions

Details such as time tracking and comments are not cloned.

Creating a Clone Issue

A clone issue can be created by selecting the **'Clone'** link available in the issue operations menu. It is possible to edit the clone issue summary before creating the clone issue.

Clone Issue Linking

The clone issue can be linked to the original issue when creating the clone issue using a pre-existing link type.

JIRA queries the property file for the `jira.clone.linktype.name` setting.

- If this property is unset, JIRA will not create a link between the original and clone issues.
- If this property is set, JIRA will use the pre-existing link type with the specified name as the link type. If a link type with this name does not exist, a link is not created between the original and clone issues.

The link type with the name specified in the properties file should be created before creating clone issues if linking is required.

Clone Issue Summary Prefix

The clone issue summary can be prefixed with a string - e.g. "Clone Issue - ". This string is specified in jira-application.properties with the `jira.clone.prefix` property and is prefixed to the issue summary.

Cloning and Sub-Tasks

Sub-Tasks can be cloned in the same manner as other issue types.

If the original issue has associated sub-tasks, clone sub-tasks will also be created for the clone issue. However, the clone sub-task summaries will not include the property file specified prefix.

It is not possible to clone an issue between projects — i.e. create a clone of an issue from one project and place it in a different project. This enhancement will be addressed in a future release.

If the current user does not have the **'Modify Reporter'** permission, the clone issue will be created with the current user as the reporter.

Moving an Issue

JIRA allows you to easily move a single issue from one project to another by using the 'Move Issue' wizard.

Please note that you must have the appropriate project permissions to move an issue from one project to another, i.e.

- You must have the 'Move Issues' permission for the project which has the issue that you want to move.
- You must have the 'Create Issues' permission for the project that you wish to move your issue to.

If you do not have either of this permissions, please contact your JIRA administrator to have these added to your user profile.

If you wish to move multiple issues between projects at the same time, please refer to the documentation on bulk moving issues .

Moving an Issue

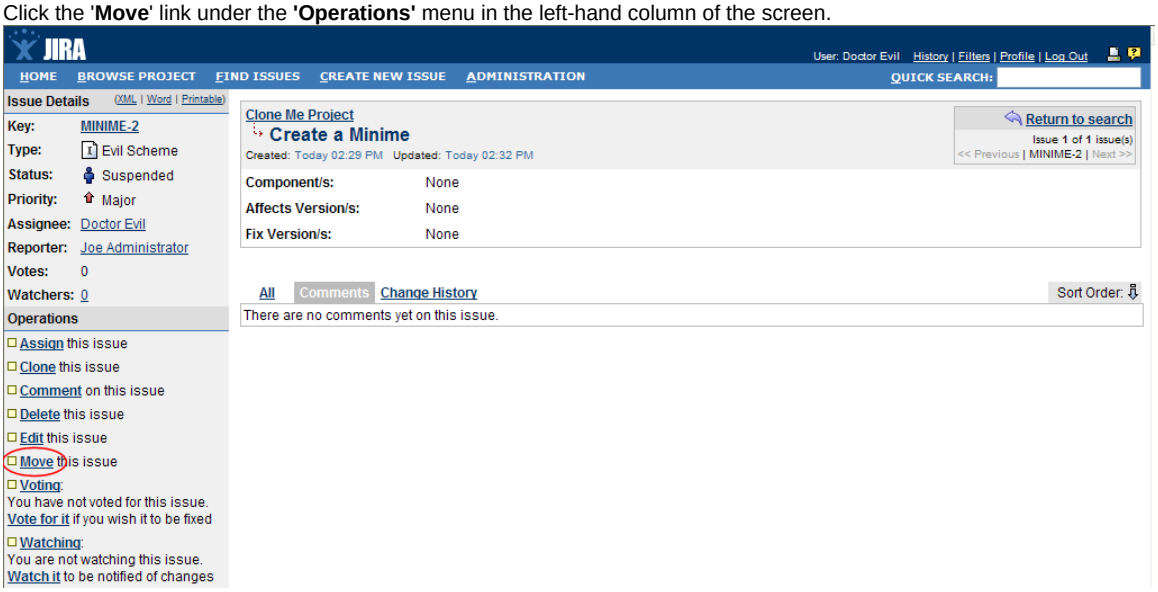
The 'Move Issue' wizard allows you to specify another project in your JIRA instance to move your selected issue to. As there may be significant differences in the configuration of your original project and target project, the 'Move Issue' wizard allows you to change certain attributes of the issue. These include:

- **Issue Type** — If your issue is a custom issue type that does not exist in your target project, you must select a new issue type. You can also choose to arbitrarily change the issue type, if you wish.
- **Issue Status** — You may have set up custom issue statuses as part of a workflow. If you have assigned a custom status to your issue and it does not exist in your target project, you must select a new issue status for your issue. You cannot arbitrarily change the issue status, i.e. the option to change the issue status will only appear if you are required to change it.
- **Custom Fields** — If you have defined **required** [custom fields] for your issue, which do not exist in your target project, you must set

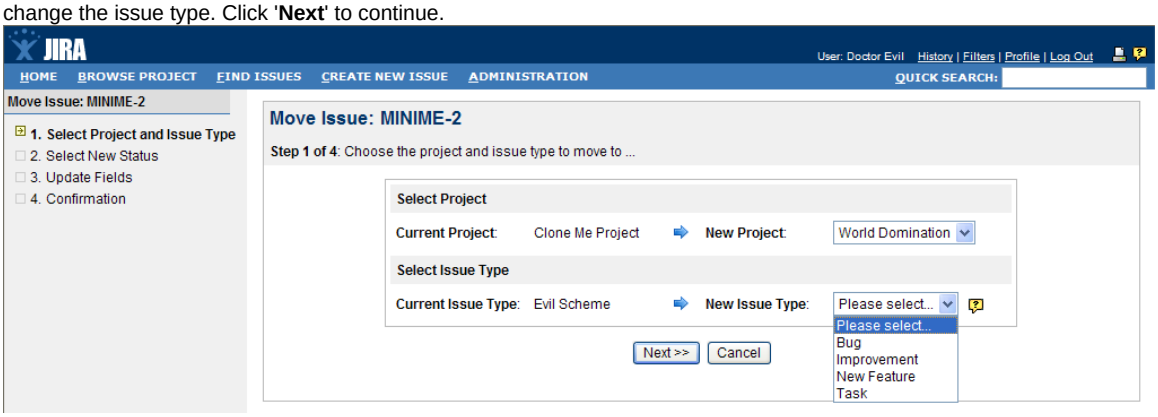
values for them. You will only be prompted to change the enter values for **required custom fields** in the target project, that are missing values. If you wish to change the existing values for other fields on your issue, you can do this after the move is complete.

To move an issue:

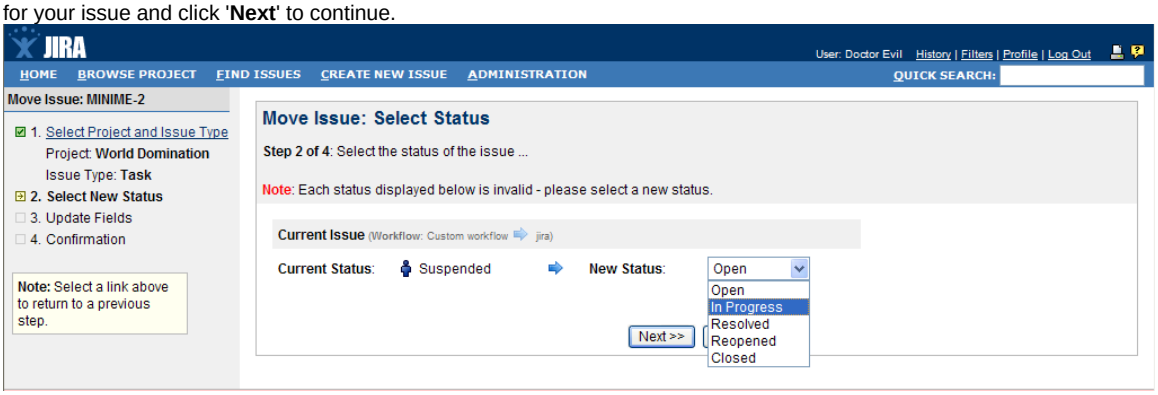
1. View the issue that you wish to move.
2. Click the '**Move**' link under the **'Operations'** menu in the left-hand column of the screen.



3. The first page of the 'Move Issue' wizard will display. Select the project that you wish to move your issue to, and if required/desired, change the issue type. Click '**Next**' to continue.

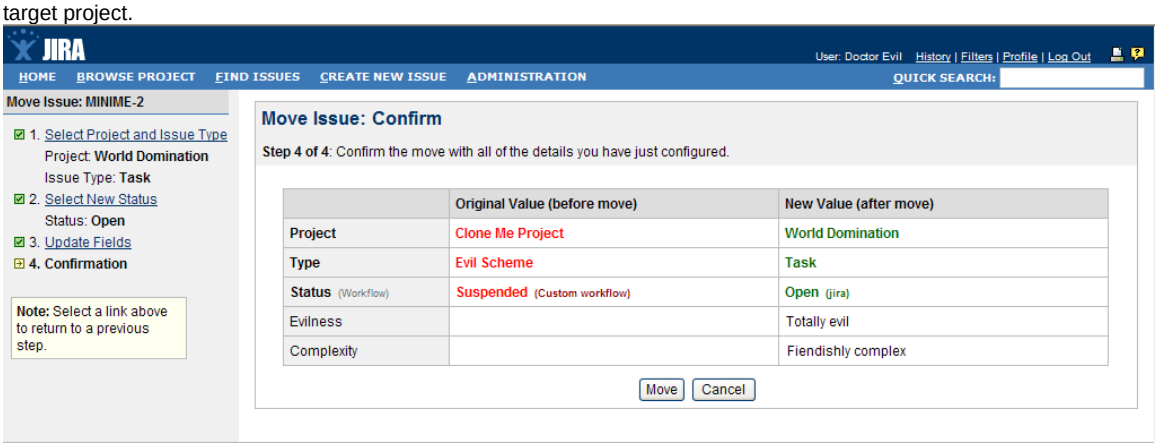


4. If you are required to change the status of your issue (see explanation above), the 'Select Status' page will display. Select the new status for your issue and click '**Next**' to continue.



5. If you are required to specify the values for any required custom fields (see explanation above), the 'Update Fields' page will display. Specify the desired values for each field, and click '**Next**' to continue.

6. The 'Confirmation' page will display with all of your changes. If you wish to revise any of your changes, you can click the appropriate step in the left-hand menu to return to that page of the wizard. Once you are happy with your changes, click '**Move**' to move the issue to the target project.



7. Your issue will be moved to the target project and displayed on screen. You can now edit the issue to make further changes, if you wish.

Viewing an Issue's FishEye Changesets

Viewing an Issue's FishEye Changesets

JIRA allows you view the changesets related to an issue (that is, where the JIRA issue key was referenced in the commit message), if you are using a source-code repository together with Atlassian FishEye.

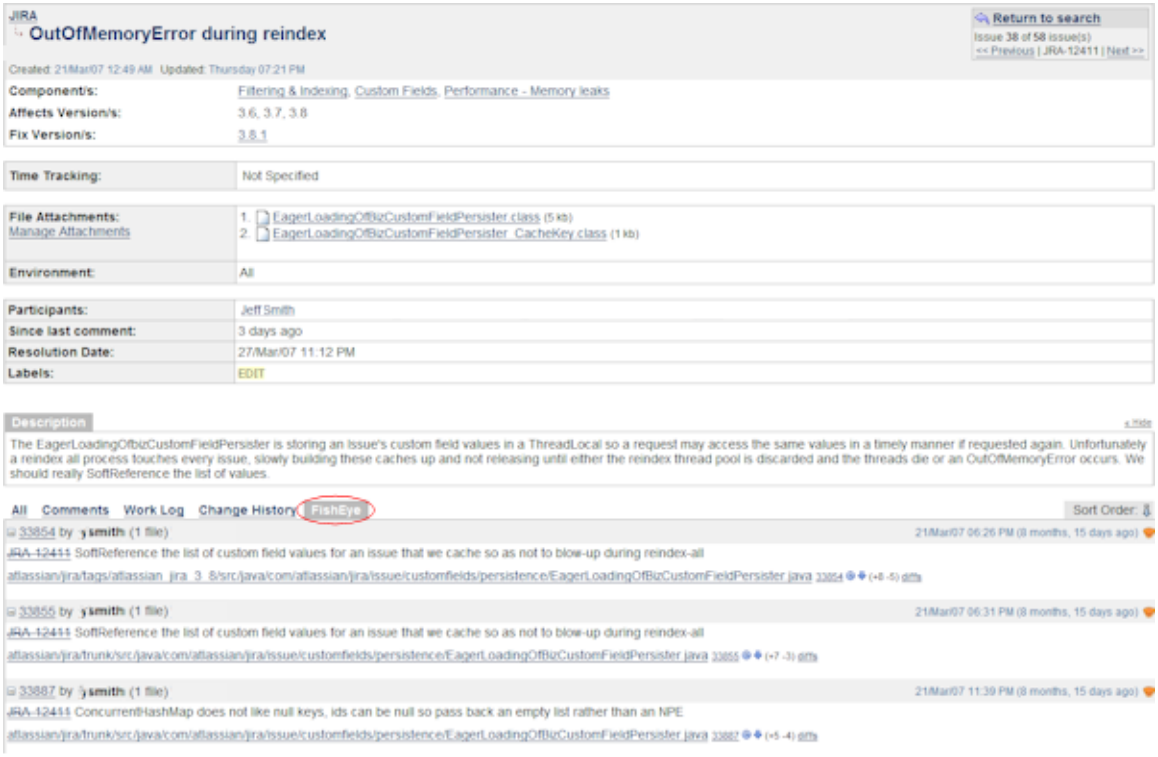
The **FishEye** tab provides you with an expandable list of changesets for the issue. This allows you to view the commit message and list of source-code files in each commit. You can also:

- view diffs and history for a file.
- download files.
- create a Crucible review and see the review status, if you are using Atlassian Crucible.

To be able to view the changesets for an issue, your JIRA administrator must have configured the FishEye plugin appropriately. You will also need the 'View Version Control' permission in the appropriate projects.

Viewing an Issue's Changesets

1. Open the relevant issue in JIRA.
2. Click the **FishEye** tab.
3. The list of changesets related to the issue will display, similar to the screenshot below:



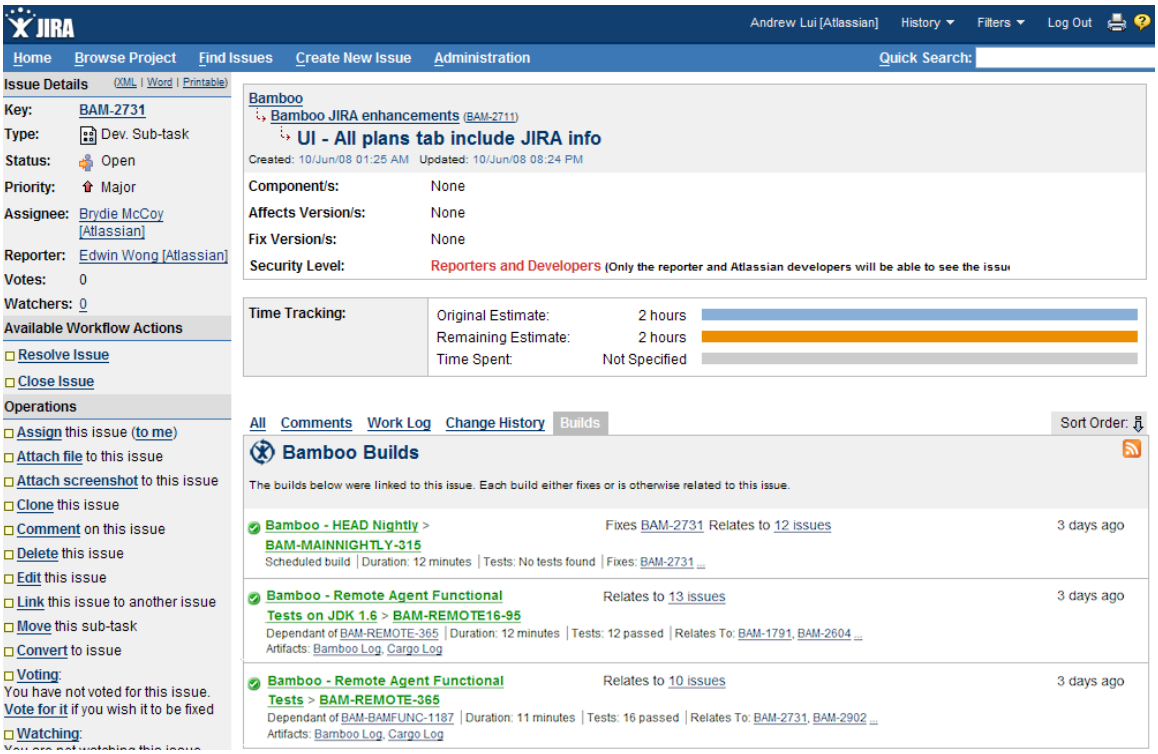
See also

- Browsing a Project's FishEye Changesets

Viewing the Bamboo Builds related to an Issue

If your organisation uses Atlassian's Bamboo and your administrator has integrated Bamboo with JIRA, you will be able to view the Bamboo builds related to an issue (that is, the JIRA issue is linked to the build in Bamboo).

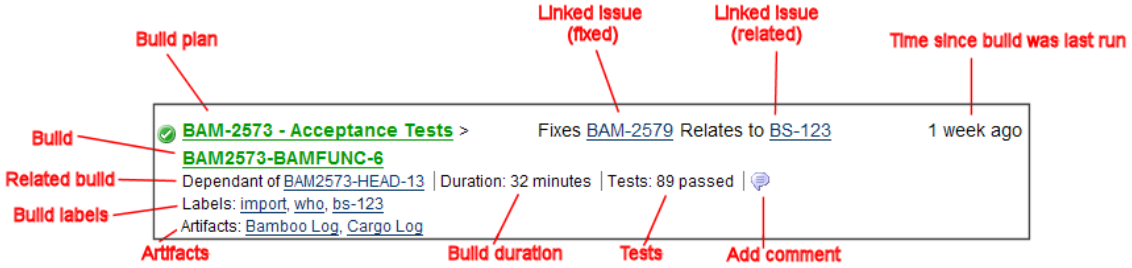
The **Builds** tab provides you with a list of the builds which the issue has been linked to, either as 'Fixed' or 'Related'. (See the Bamboo documentation for instructions on linking issues to builds).



Each entry in the list will display information about the related build, including:

- the build name and name of the build plan
- when the build was last run
- summary information, such as related builds, duration of the build, tests passed
- build labels (if any)
- links to build artifacts (if any)

See the diagram below for details:



To view the Bamboo builds related to an issue,

1. Open the issue in JIRA.
2. Click the **'Builds'** tab.
3. The builds related to the issue will display.

> If you cannot see the Builds tab, your administrator may need to add the **'View Version Control'** permission to your project.

See also:

- Browsing a Project's Bamboo Builds
- Browsing a Version's Bamboo Builds

Modifying Multiple ('Bulk') Issues

Bulk Operations enables multiple operations to be performed on multiple issues at once. The bulk operations are performed on the result set of a search. The following list details the available bulk operations:

- **Workflow Transition**
  This operation allows multiple issues to be transitioned through workflow at once - e.g. resolve a collection of issues.
- **Delete**
  This operation allows multiple issues to be deleted at once.
- **Move**
  This operation allows multiple issues to be moved at once. Please see the Bulk Move section for further details.
- **Edit**
  This operation allows multiple edit operations to be performed. Please see the Bulk Edit section for further details.

On this page:

- Bulk Change Global Permission
- Disable Mail Notification for Bulk Operations
- Steps to Perform Bulk Operations
- Bulk Move
  - Issue Selection
  - Select Projects and Issue Types
  - Select Projects and Issue Types for Sub-Tasks
  - Workflow Status Mapping
  - Field Updates
  - Retain Original Values
  - Bulk Move Confirmation
- Bulk Edit
  - Available Operations
  - Unavailable Operations

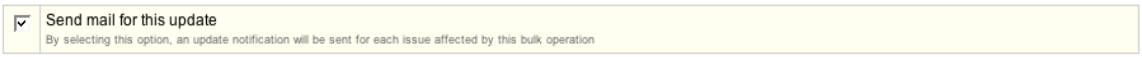
Bulk Change Global Permission

It is necessary for a user to be granted the appropriate project specific permission and the global Bulk Change permission in order to execute a bulk operation. For example, a user must be granted both the **Move Issue** and **Bulk Change** permissions in order to execute the **Bulk Move** operation.

The project specific permissions are still respected for the collection of issues selected for the bulk operation.

Disable Mail Notification for Bulk Operations

It is possible to disable mail notification for a particular bulk operation by de-selecting the "Send Notification" checkbox in the bulk operation wizard. The user must be an administrator or project administrator of all the associated projects for the bulk operation in order for this option to be available.



Steps to Perform Bulk Operations

1. From the Issue Navigator, perform a search with the required filters to produce an issue result set.
2. The **Bulk Change** option in the "**Tools**" menu of the Issue Navigator provides access to the bulk operations. The **Bulk Change** link is only available to users who have been granted the global Bulk Change permission. If the result set spans a number of pages, it is possible to select all issues within the result set to be considered for the bulk operation. Alternatively, all issues on the current page can be selected for the bulk operation.
3. The next screen allows the selection of the issues that the bulk operation is to be performed on.
4. The next screen allows the bulk operation to be selected - **Workflow Transition, Delete, Move** or **Edit.**
5. If the **Delete** operation is selected, the final step is confirmation of the delete operation on the issues selected.
6. If the **Edit** operation is selected, the next screen provides a list of the available edit operations that can be performed on the issues selected. Some operations may be unavailable, please check the Bulk Edit section for further details.
7. After selecting the required **Edit** operation(s), the final step is confirmation of the edit operation(s) on the selected issues.
8. If the **Move** operation is selected, the next screens allow a target project (and issue type if Enterprise edition) to be selected, with the ability to migrate workflow statuses and update required fields as necessary. Further details can be found in the Bulk Move section.
9. If the **Workflow Transition** operation is selected, the next screen shows the available workflow transitions that can be performed on the issues. The transitions are grouped by workflow - along with a list of the affected issues for each workflow transition. Once an operation is selected, the appropriate field screen for that operation is displayed - allowing any necessary field edits that are required to complete the transition. It should be noted that only those issues associated with the selected transition will be updated. It is only possible to select one transition per bulk workflow transition operation.

Bulk Move

The Bulk Move operation allows multiple issues to be moved at once. It is possible to move a selection of issues to a new project with the ability to select a new issue type in certain cases. The issues are selected through the Issue Navigator as discussed above.

The operation is completed as follows:

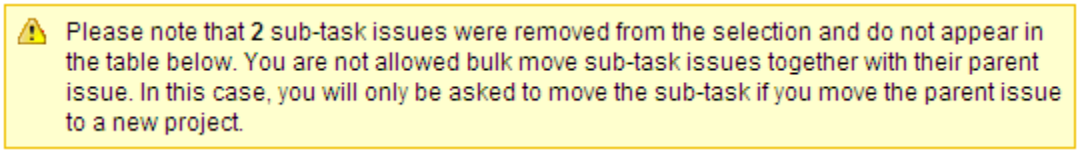
1. Select Projects and/or Issue Types
2. Select Projects and/or Issue Types for Sub-Tasks
3. Select status migration mappings for invalid statuses
4. Select values for required fields and fields with invalid values
5. Confirm changes to be made and complete the operation
   Note that steps 3 and 4 will occur once for each different target project and issue type combination.

Issue Selection

The bulk move operation can be performed on both standard issues and sub-task issues. Standard issues can be moved to another project and issue type, whereas a sub-task can only have its issue type changed.

It is **not** possible to select *both* a sub-task and its parent to bulk move. This is so to adhere to the parent/sub-task relationship (i.e. the sub-task is always located in the same project as the parent issue). Any sub-tasks of selected parent issues which were also selected will be automatically discarded from the move.

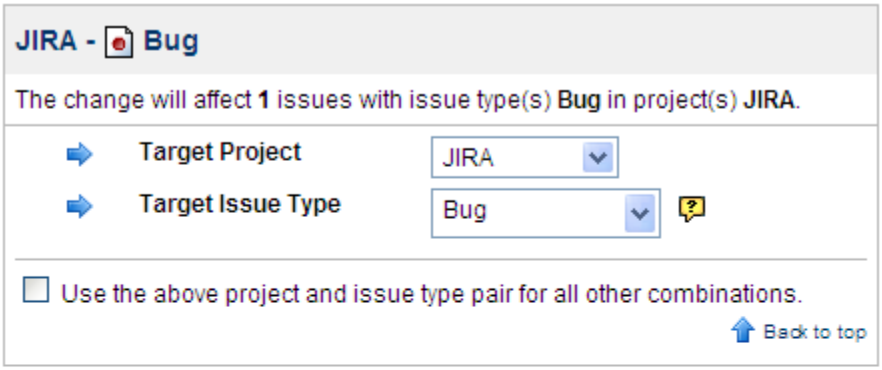
For example, you have issue B being a sub-task of issue A and you try to bulk move both A and B simultaneously. You will see a warning message (see below) and will be prompted to select a target project and issue type for issue A. If you select a new project for A, you will be prompted to move the sub-task to a new issue type based on issue A's new project. If you *don't* change the project for issue A, the sub-task will not be required to be moved.

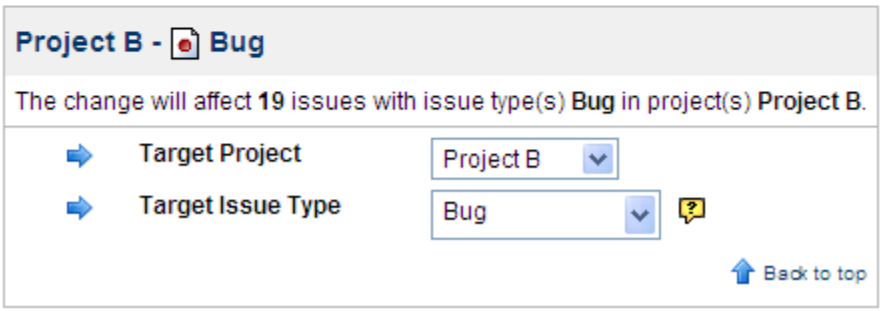


Select Projects and Issue Types

The first step of the bulk move wizard is to choose what projects and issue types you'll move your issues to. The target project and issue type will determine whether extra steps will be required to migrate statuses and fields.

This screen shows all selected issues grouped by their current project and issue type. You can either select a new project and issue type for each one or choose to move all standard issues to a single project and issue type. To do this, select the check box with the label *Use the above project and issue type pair for all other combinations* and the selected project / issue type will apply. Note that this will not apply to sub-tasks since they cannot be moved to a standard issue type.

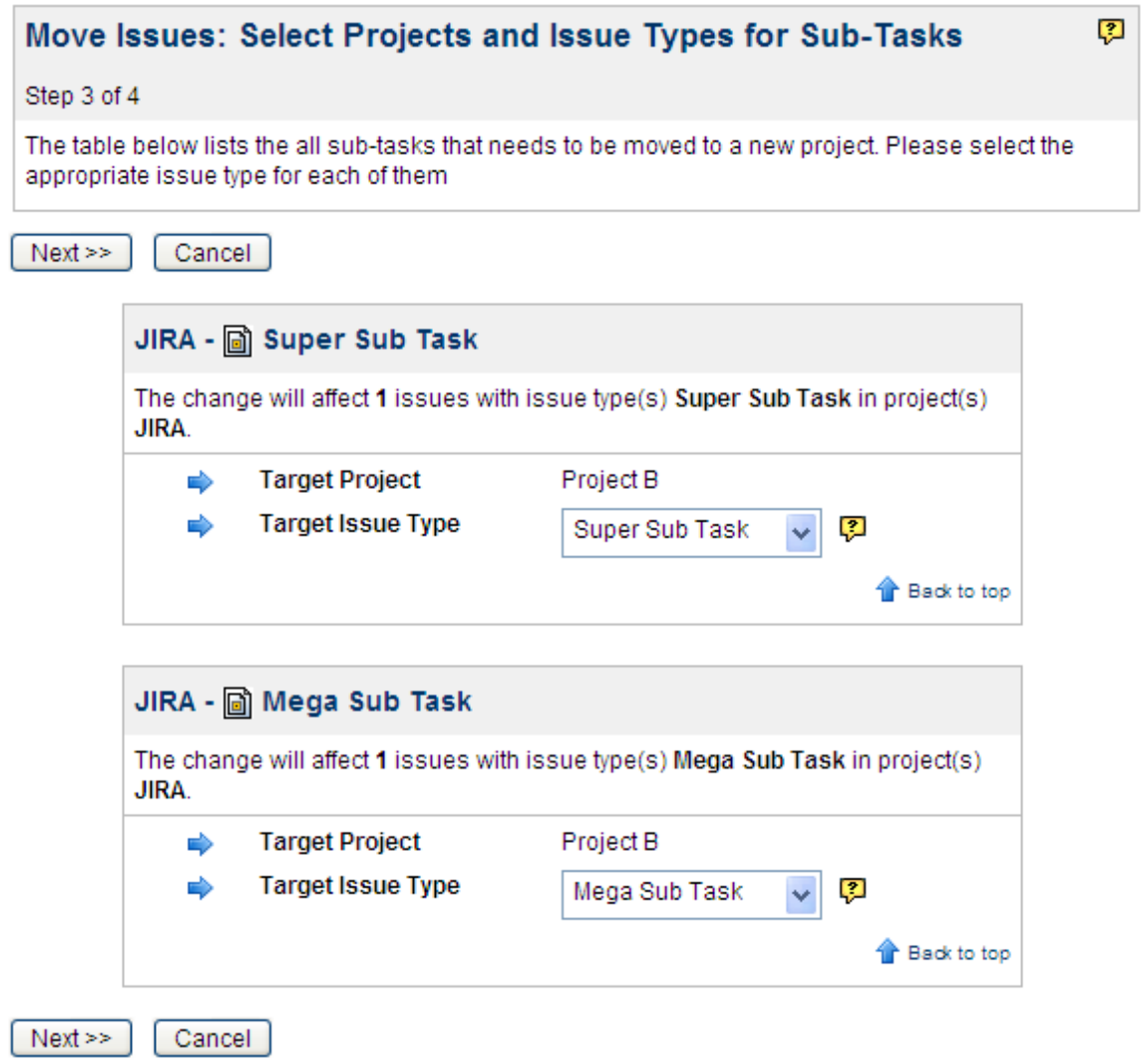




Select Projects and Issue Types for Sub-Tasks

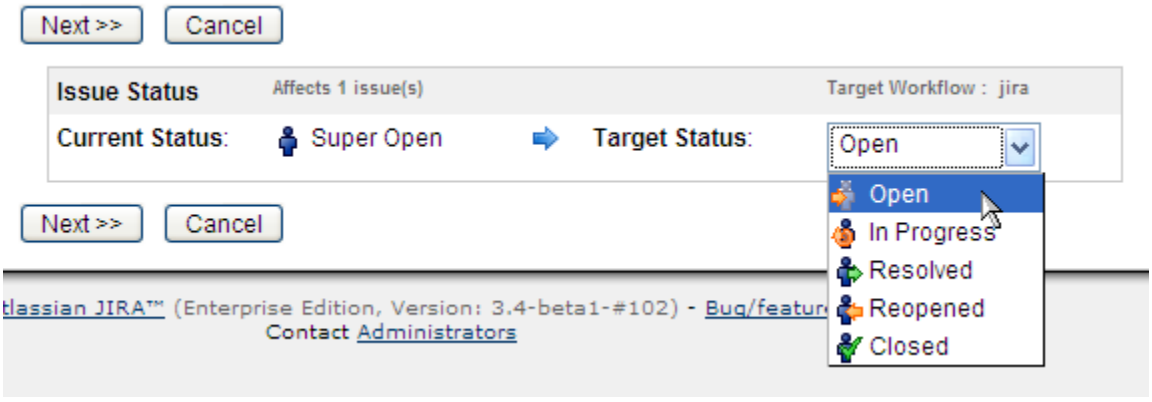
If you are moving issues with sub-tasks to another project, you'll also need to move the sub-tasks to the new project. On this screen you can elect

to change the issue types of the sub-tasks being moved if you need to.



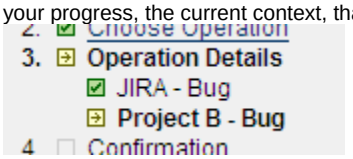
Workflow Status Mapping

As multiple workflows can be active simultaneously within Enterprise Edition, some statuses associated with the collection of selected issues may not be valid in the target workflow. In this case, JIRA allows the user to specify a mapping from invalid statuses to those available in the the target workflow.



This step of the wizard will only appear if you have invalid statuses. If you are moving issues to different projects and issue types at the same time, you will complete this step as well as the next for each of the different target project and issue type pairs. To help you easily keep track of

your progress, the current context, that is the target project and issue type, is highlighted in the left-hand, "progress" pane.



Field Updates

In order to adhere to the field configuration scheme associated with the target project and issue type, it may be necessary to update/populate required fields. For example, if one of the selected issues does not have a value for a required component and version fields, JIRA prompts the user to enter a suitable value.

Field Name	Field Value	Retain
*Component/s:	New Component 1 New Component 2 New Component 3	☐
*Fix Version/s:	Unreleased Versions New Version 2 New Version 3 New Version 4 Released Versions New Version 1	☐
*Affects Version/s:	Released Versions New Version 1 Unreleased Versions New Version 2 New Version 3 New Version 4	☑

Retain Original Values

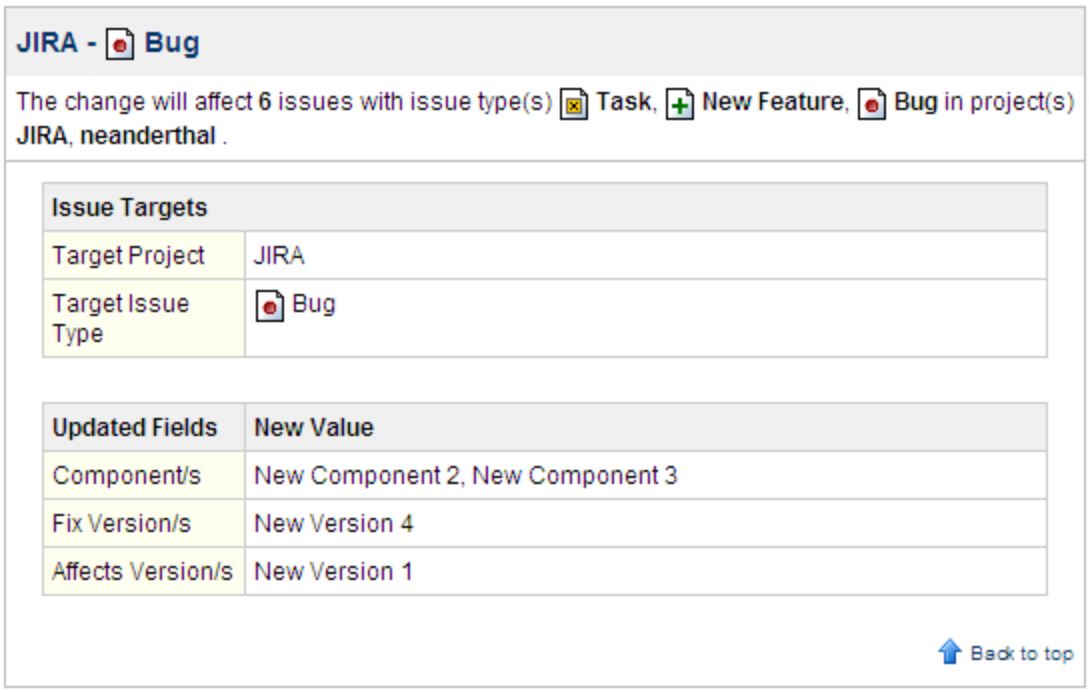
It is possible to retain original field values that are valid in the target destination by checking the **Retain** checkbox associated with the field. For example, some issues may already include a valid custom field value - these values can be retained, while issues that require an update will adopt the value specified on the field update screen.

- **Checked:** the original value is retained where possible. The field will not be updated with the specified new value.
- **UnChecked:** all fields will be updated with the specified new value.

Bulk Move Confirmation

When all move parameters - e.g. target project, status mappings and field updates - have been specified for all issues, the user is presented with a confirmation screen displaying all changes that will be made to the issues being moved. The following details are displayed:

- **Issue Targets:** the target project and issue type
- **Workflow:** the target workflow and invalid status mappings
- **Updated Fields:** new values for fields that require updating
- **Removed Fields:** values to be removed in fields that are not valid in the target
  The issues will only be moved once the **Confirm** button is clicked from the confirmation page. If the operation is exited anytime before this step, no changes will be made to the issues.



Bulk Edit

The bulk edit operations available depend on the issues selected and the nature of the field it changes.

Available Operations

The following table lists out all the possible operations. Please note that *all* the conditions must be true for the corresponding operation to be available.

Available Operations	Conditions
Change Affects Version/s	• Selected issues belong to one project, and that project has version/s • This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues
Change Assign To	• This field is not hidden in any field configurations the selected issues belong to • Current user has 'assign issue' permission for all the selected issues
Change Comment	• This field is not hidden in any field configurations the selected issues belong to • Current user has 'comment issue' permission for all the selected issues
Change Component/s	• Selected issues belong to one project, and that project has component/s • This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues
Change Due Date	• This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues • Current user has 'schedule issue' permission for all the selected issues

Change Fix For Version/s	• Selected issues belong to one project, and that project has version/s • This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues
Change Issue Type	• Current user has 'edit issue' permission for all the selected issues
Change Priority	• This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues
Change Reporter	• This field is not hidden in any field configurations the selected issues belong to • Current user has 'edit issue' permission for all the selected issues • Current user has 'modify reporter' permission for all the selected issues
Change Security Level	• This field is not hidden in any field configurations the selected issues belong to • All the selected projects are assigned the same issue level security scheme • Current user has 'edit issue' permission for all the selected issues • Current user has 'set issue security' permission for all the selected issues
Change Custom Fields	The 'Change Custom Fields' operation is available only if: • a global custom field exists **OR** • an issue type custom field exists and the issues are all of this specific issue type **OR** • a project custom field exists and the issues are all of the same project

Unavailable Operations

The fields listed in this section have no operations for bulk editing. This is because there is an alternative method or it is not logical to perform bulk edit on them.

The following system fields are unavailable for bulk editing:

- Attachments
- Summary
- Description
- Environment
- Project — Please use 'Bulk Move' to move issues between projects
- Resolution — Please use 'Bulk Workflow Transitions' to modify the resolution of issues

The following custom fields are unavailable for bulk editing:

- Import Id
- Read Only Text

Searching for Issues

JIRA provides a powerful issue search facility. You can search for issues across projects, versions and components using a range of search criteria. If you need to find issues based on time-tracking details, then the standard search interface also allows you to search against the work logged on issues. JIRA also makes custom fields available as search criteria, allowing you to refine your searches even further.

You can also save your search as an issue filter in JIRA, allowing you to recall the same search and run it again or even share it with other users. Read more about issue filters.

On this page:

- Searching for issues
  - Searching particular projects or issue types
  - Searching custom fields
  - Using the 'Work Ratio' section

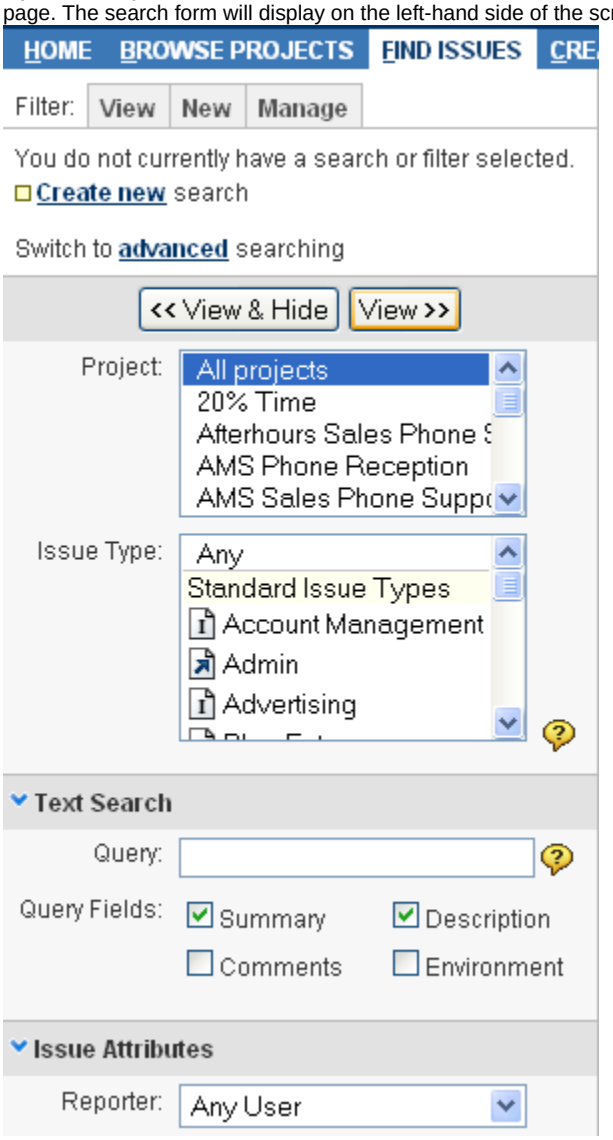
In this chapter:

- Using Quick Search

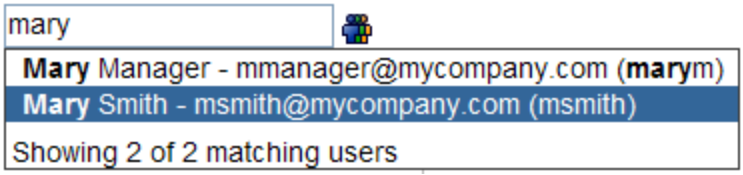
- Performing Text Searches
- Advanced Searching
- Saving Searches ('Issue Filters')
- Receiving Search Results via Email
- Using the Issue Navigator
    - Customising your Issue Navigator
    - Displaying Search Results in XML
    - Receiving Search Results as an RSS Feed
    - Exporting Search Results to Microsoft Word
    - Exporting Search Results to Microsoft Excel
    - Displaying Search Results as a Chart

Searching for issues

1. On the top navigation bar, click the '**Find Issues**' tab. This will display information on the issue filter or search you currently have selected, if any, on the left hand side of the page. If you have an issue filter or search currently selected, the results will be displayed in the 'Issue Navigator' on the right hand side of the page. Otherwise, no results will be shown.
2. If you currently have an issue filter or search selected and wish to run a new search, click the '**New**' link on the top left hand side of the page. The search form will display on the left-hand side of the screen.



   Several of the form items have contextual help available, denoted by the icon.
3. Type your search term(s) into the '**Query**' box (see Performing Text Searches for help with this), and/or select other criteria from the drop-down boxes and check-boxes. The drop-down boxes and check-boxes allow you to narrow your search, be it to issues in a certain project, only issues that are marked as 'stoppers', only issues marked as 'enhancements', only issues reported by or assigned to a particular person, and so on.
   For example, to find issues reported by a particular person, select 'Specify User' in the '**Reporter**' field. In the field that appears beneath, type a few letters of the person's name (first name or surname) to display a drop-down list:



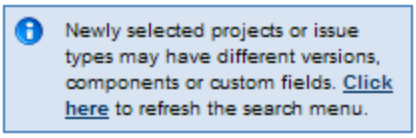
Please note: if the user drop-down does not display, your administrator may have disabled the 'User Picker Auto-complete' feature for your JIRA instance.

4. Click the '**View**' button to perform the search and keep the search form visible; or click the '**View & Hide**' button to perform the search and hide the search form.

While you are interacting with JIRA, your current search will be remembered, along with whether the search you are using is new, a saved search, or a modification of a saved search.

Searching particular projects or issue types

When using the search form, your selected Project(s) and Issue Type(s) determine what other fields are shown in the search form and what options you can see for these fields. For example, the 'Version' and 'Component' fields will only be available when searching for a single project, and will have different options for each project. When you change the project(s) or issue type(s), you may need to *refresh* the search filter in order to get the most up-to-date versions, components and custom fields. If a refresh is needed, a blue box will appear, containing a link for you to click.



If you search on a single project, JIRA will remember that as your selected project and will default to that project on the "Browse Project" and "Create Issue" screens.

Searching custom fields

Your administrator may have created custom fields for your JIRA system. Custom fields can be searched, but note that they will only appear in the search form on the left when appropriate. That is:

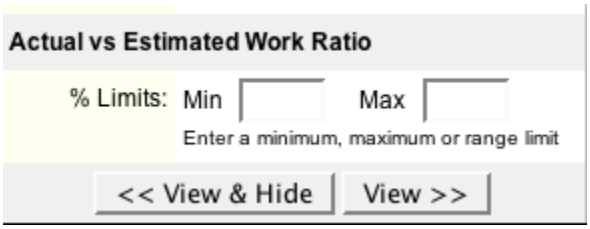
- Custom fields that relate to a particular *project* will only appear in the search form after you choose that project.
- Custom fields that relate to a particular *issue type* will only appear in the search form after you choose that issue type.

Using the 'Work Ratio' section

The search form contains a 'Work Ratio' section, enabling you to search JIRA issues based on time-tracking details.

The 'Work Ratio' search is based on the **Actual** work logged against an issue versus the original **Estimated** work duration.

- **Work Ratio Percentage = (Actual / Estimated Work) x 100**
  You can enter a minimum, maximum or percentage range; the search will respectively return all issues above, below or within the specified percentage range.



Using Quick Search

Sometimes you just want to be able to get to the particular issue that you are interested in. Other times you can't remember what the issue was,

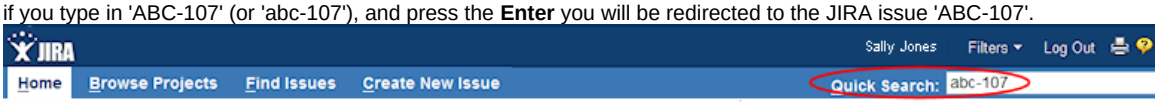
but you remember that it was an open issue, assigned to you. Quick Search can help you.

On this page:

- Jump to an Issue
- Smart Querying
- Free-text searching

Jump to an Issue

The Quick Search box is located at the top right of your screen. If you type in the **key** of an issue, you will jump straight to that issue. For example, if you type in 'ABC-107' (or 'abc-107'), and press the **Enter** you will be redirected to the JIRA issue 'ABC-107'.



In many cases, you do not even need to type in the full key, but just the numerical part. If you are currently working on the 'ABC' project, and you type in '123', you will be redirected to 'ABC-123'.

Smart Querying

Quick Search also enables you to perform 'smart' searches with minimal typing. For example, to find all the open bugs in the 'TEST' project, you could simply type 'test open bugs', and Quick Search would locate them all for you.

Your search results will be displayed in the Issue Navigator, where you can view them in a variety of useful formats (Excel, XML, etc).

The search terms that Quick Search recognises are:

- **Issues assigned to me** — You can use the keyword 'my', as in *'my* open bugs'.
- **Issues in a particular project** — To find all issues in a project, type the project name, e.g. *'test',* or the project key, e.g. *'TST'* (or *'tst').*
- **Overdue issues** — You can use the keyword 'overdue' to search for issues that were due before today.
- **Issues with a particular Created, Updated, or Due Date** — You can find issues with certain dates. You can use the prefix *created:*, *updated:*, or *due:*. For the date range, you can use *today*, *tomorrow*, *yesterday*, a single date range (e.g. '-1w'), or two date ranges (e.g. '-1w,1w'). Note that date ranges cannot spaces in them. Valid date/time abbreviations are: 'w' (week), 'd' (day), 'h' (hour), 'm' (minute). Some examples:
  - 'created:today'
  - 'created:yesterday'
  - 'updated:-1w' — issues updated in the last week
  - 'due:1w' — issues due in the next week.
  - 'due:-1d,1w' — all issues due from yesterday to next week.
  - 'created:-1w,-30m' — all issues created from one week ago, to 30 minutes ago.
  - 'created:-1d updated:-4h' — all issues created in the last day, updated in the last 4 hours.
- **Issues with a particular Status** — You can use the name of any status in your search, e.g. *'open'*, *'closed'*.
- **Issues with a particular Resolution** — You can use a resolution name to search for issues with a particular resolution, e.g. type *'duplicate'* to search for all issues that have a resolution of "Duplicate". You can also use the keyword *'unresolved'* to find all issues that do not yet have a resolution.
- **Issues with a particular Priority** — You can type a priority name to search for issues with a particular priority, e.g. *'blocker'*.
- **Issues of a particular Issue Type** — You can use the type of an issue in the search. Examples include *bug*, *task*. Note that you can also include plurals, such as *bugs.*
- **Issues with a particular Version** — You can use the prefix "v:" to search for issues with a particular version(s). *Note that there can be no spaces between "v:" and the version name.* "v:3.0" will match against the following versions (for example):
  - 3.0
  - 3.0 enterprise
  - 3.0 standard
    but will not match against the following versions (for example):
  - 3.0.1
  - 3.0.0.4
    That is, it will match against any version that contains the string you specify followed immediately by a space, but not against versions that do not contain a space immediately after the string you specify.
- **Issues with a particular Fix For Version** — You can use the prefix "ff:" to search for issues with a particular fix for version(s). The search mechanism is the same as the quick search for Issues with a particular Version.
- **Issues with a particular Component** — You can use the prefix "c:" to search for issues with a particular component(s). This allows you to search across multiple components. "c:security" will search for all issues with a component that contains the word "security". *Note that there can be no spaces between "c:" and the component name.*

In Mozilla-based browsers, try creating a bookmark with URL `http://jira.atlassian.com/secure/QuickSearch.jspa?searchString=%s` (substituting in your JIRA URL) and keyword *'j'*. Now typing *'j my open bugs'* in the browser URL bar will search JIRA for your open bugs.

Free-text searching

You can search for any word within the issue(s) you are looking for, provided the word is in one of the following fields:

- Summary
- Description
- Comments

Note that, unlike the keywords listed under 'Smart Querying' above, free-text search works in both the Quick Search box *and* the Issue Filter Text Search box.

Note that you can combine free-text and keywords together. For example, *'my closed tst tasks', 'open test bugs pear', 'closed test bugs '* are all valid search queries.

Performing Text Searches

This page provides information on how to perform text searches. It applies to both simple searches and advanced searches . (Note that this page does not apply to Quick Search).

Acknowledgements:
JIRA uses Lucene for text indexing. Lucene provides a rich query language; thanks to Jakarta and the Lucene team for such a great component. Most of the information on this page is derived from the Lucene document on Query Parser Syntax.

On this page:

- Query Terms
- Term Modifiers
  - Wildcard Searches: ? and *
  - Fuzzy Searches: ~
  - Proximity Searches
- Boosting a Term: ^
- Boolean Operators
  - OR
  - AND
  - Required term: +
  - NOT
  - Excluded term: -
- Grouping
- Escaping Special Characters: \
- Limitations
  - Whole words only

Query Terms

A query is broken up into **terms** and **operators.** There are two types of terms: **Single Terms** and **Phrases.**

A **Single Term** is a single word such as "`test`" or "`hello`".

A **Phrase** is a group of words surrounded by double quotes such as "`hello dolly`".

Multiple terms can be combined together with Boolean operators to form a more complex query (see below).

Note: All query terms in JIRA are case insensitive.

Term Modifiers

JIRA supports modifying query terms to provide a wide range of searching options.

*Wildcard Searches: ? and **

JIRA supports single and multiple character wildcard searches.

To perform a single character wildcard search use the "?" symbol.

To perform a multiple character wildcard search use the "`*`" symbol.

The single character wildcard search looks for terms that match that with the single character replaced. For example, to search for "`text`" or "`test`" you can use the search:

```
te?t
```

Multiple character wildcard searches looks for 0 or more characters. For example, to search for `Windows`, `Win95` or `WindowsNT` you can use the search:

```
win*
```

You can also use the wildcard searches in the middle of a term. For example, to search for `Win95` or `Windows95` you can use the search

```
wi*95
```

You cannot use a * or ? symbol as the first character of a search.

Fuzzy Searches: ~

JIRA supports fuzzy searches. To do a fuzzy search use the tilde, "~", symbol at the end of a Single word Term. For example to search for a term similar in spelling to "`roam`" use the fuzzy search:

```
roam~
```

This search will find terms like foam and roams.

Note: Terms found by the fuzzy search will automatically get a boost factor of 0.2

Proximity Searches

JIRA supports finding words are a within a specific distance away. To do a proximity search use the tilde, "~", symbol at the end of a Phrase. For example to search for a "`atlassian`" and "`jira`" within 10 words of each other in a document use the search:

```
"atlassian jira"~10
```

Boosting a Term: ^

JIRA provides the relevance level of matching documents based on the terms found. To boost a term use the caret, "^", symbol with a boost factor (a number) at the end of the term you are searching. The higher the boost factor, the more relevant the term will be.

Boosting allows you to control the relevance of a document by boosting its term. For example, if you are searching for

```
atlassian jira
```

and you want the term "`atlassian`" to be more relevant boost it using the ^ symbol along with the boost factor next to the term. You would type:

```
atlassian^4 jira
```

This will make documents with the term atlassian appear more relevant. You can also boost Phrase Terms as in the example:

```
"atlassian jira"^4 querying
```

By default, the boost factor is 1. Although, the boost factor must be positive, it can be less than 1 (i.e. .2)

Boolean Operators

Boolean operators allow terms to be combined through logic operators. JIRA supports AND, "+", OR, NOT and "-" as Boolean operators.

Boolean operators must be ALL CAPS.

OR

The OR operator is the default conjunction operator. This means that if there is no Boolean operator between two terms, the OR operator is used. The OR operator links two terms and finds a matching document if either of the terms exist in a document. This is equivalent to a union using sets. The symbol `||` can be used in place of the word OR.

To search for documents that contain either "`atlassian jira`" or just "`jira`" use the query:

```
"atlassian jira" || jira
```

or

```
"atlassian jira" OR jira
```

AND

The AND operator matches documents where both terms exist anywhere in the text of a single document. This is equivalent to an intersection using sets. The symbol `&&` can be used in place of the word AND.

To search for documents that contain "`atlassian jira`" and "`issue tracking`" use the query:

```
"atlassian jira" AND "issue tracking"
```

Required term: +

The "" or required operator requires that the term after the "" symbol exist somewhere in a the field of a single document.

To search for documents that must contain "`jira`" and may contain "`atlassian`" use the query:

```
+jira atlassian
```

NOT

The NOT operator excludes documents that contain the term after NOT. This is equivalent to a difference using sets. The symbol `!` can be used in place of the word NOT.

To search for documents that contain "`atlassian jira`" but not "`japan`" use the query:

```
"atlassian jira" NOT "japan"
```

Note: The NOT operator cannot be used with just one term. For example, the following search will return no results:

```
NOT "atlassian jira"
```

Usage of the **NOT** operator over multiple fields may return results that include the specified excluded term. This is due to the fact that the search query is executed over each field in turn and the result set for each field is combined to form the final result set. Hence, an issue that matches the search query based on one field, but fails based on another field, will be included in the search result set.

Excluded term: -

The "`-`" or prohibit operator excludes documents that contain the term after the "`-`" symbol.

To search for documents that contain "`atlassian jira`" but not "`japan`" use the query:

```
"atlassian jira" -japan
```

Grouping

JIRA supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

To search for either "`atlassian`" or "`jira`" and "`bugs`" use the query:

```
bugs AND (atlassian OR jira)
```

This eliminates any confusion and makes sure you that `bugs` must exist and either term `atlassian` or `jira` may exist.

Do not use the grouping character '(' at the start of a search query, as this will result in an error. For example, "(atlassian OR jira) AND bugs" will not work.

Escaping Special Characters: \

JIRA supports escaping special characters that are part of the query syntax. The current list of special characters is:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? \
```

To escape these characters, use the \ before the character. For example, to search for (1+1) use the query:

```
\(1\+1\)
```

Limitations

Please note that the following limitations of Lucene apply to JIRA:

Whole words only

You cannot search on word parts, only on whole words.

Advanced Searching

On this page:

- What is an Advanced Search?
- How to Perform an Advanced Search
    - JQL Language Reference
    - Fields and Functions Reference
    - Setting Precedence of Operators
    - Searching for Strings
    - Performing Text Searches
    - Using Auto-complete
    - JQL Reserved Words
    -

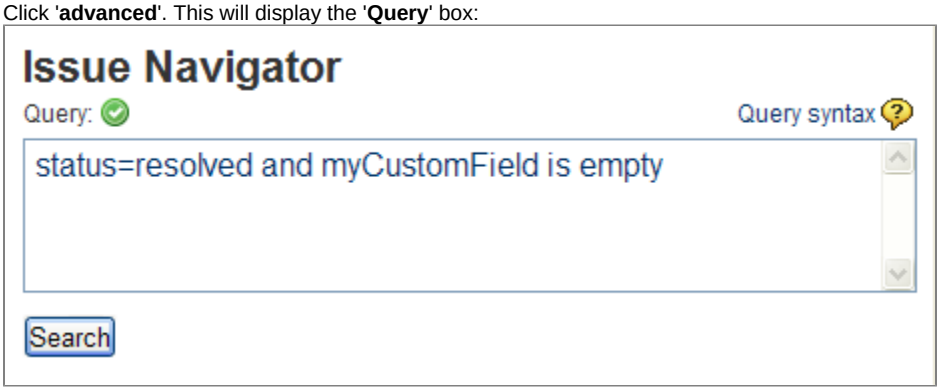
What is an Advanced Search?

An advanced search allows you to use structured queries to search for JIRA issues. Your query will consist of a *field or function* followed by an *operator* followed by a *value*, e.g.:

```
project = "TEST"
```

How to Perform an Advanced Search

1. On the top navigation bar, click the '**Issues**' tab. This will display the Search panel.
2. Click '**advanced**'. This will display the '**Query**' box:



3. Type your query using the JIRA Query Language (JQL) and the fields and functions listed below.
4. Click the '**Search**' button to run your query.
5. Your search results will be displayed in the Issue Navigator, where you can export them to MS Excel and many other formats. You can also save and subscribe to your advanced searches if you wish.

JQL Language Reference

- AND
- OR
- NOT
- IN
- EQUALS: = [1]
- GREATER THAN: > [1]
- GREATER THAN EQUALS: >= [1]
- LESS THAN: < [1]
- LESS THAN EQUALS: <= [1]
- MATCHES: ~ [1] [2]
- IS [1]
- IS NOT [1]
- ESCAPE CHARACTER: \
- WHITESPACE
- EMPTY
- NULL
- ORDER BY [1]

Statement	Example	Syntax
AND	Find all open issues in the "New office" project:	`project = "New office" and status = "open"`
	Find all open, urgent issues that are assigned to jsmith:	`status = open and priority = urgent and assignee = jsmith`
	Find all issues in a particular project that are not assigned to jsmith:	`project = JRA and assignee != jsmith`
	Find all issues for a specific release which consists of different version numbers across several projects:	`project in (JRA,CONF) and fixVersion = "3.14"`

OR	Find all issues that were created by either jsmith or jbrown: *(Note: also see IN)*	`reporter = jsmith or reporter = jbrown`
	Find all issues that are overdue or where no due date is set:	`duedate < now() or duedate is empty`

NOT	Find all issues that are assigned to any user except jsmith:	`not assignee = jsmith`
		or:
		`assignee != jsmith`

	Find all issues that were not reported by jsmith:	`reporter !=jsmith`
	Find all issues that were reported by me but are not assigned to me:	`reporter = currentUser() and assignee != currentUser()`
	Find all issues where the Reporter or Assignee is anyone except John Smith:	`assignee != "John Smith" or reporter != "John Smith"`
	Find all issues where the Assignee is not null	`assignee != empty`
IN	Find all issues that were created by either jsmith or jbrown or jjones:	`reporter in (jsmith,jbrown,jjones)`
	Find all issues where the Reporter or Assignee is either Jack or Jill:	`reporter in (Jack,Jill) or assignee in (Jack,Jill)`
	Find all issues where neither the Reporter nor the Assignee is Jack, Jill or John:	`reporter not in (Jack,Jill,John) and assignee not in (Jack,Jill,John)`
	Find all issues in version 3.14 or version 4.2:	`affectedVersion in ("3.14", "4.2")`

EQUALS: = [1]	Find all issues that were created by jsmith: [1] *Note: it is not possible to compare two fields.*	`reporter = jsmith`
	Find all issues that were created by John Smith:	`reporter = "John Smith"`
GREATER THAN: > [1]	Find all issues with more than 4 votes: [1] *Note: it is not possible to compare two fields.*	`votes > 4`
	Find all overdue issues:	`duedate < now()  and resolution is empty`
	Find all issues where priority is higher than "Normal":	`priority > normal`
GREATER THAN EQUALS: >= [1]	Find all issues with 4 or more votes: [1] *Note: it is not possible to compare two fields.*	`votes >= 4`
	Find all issues due on or after 31/12/2008:	`duedate >= "2008/12/31"`
	Find all issues created in the last five days:	`created >= "-5d"`
LESS THAN: < [1]	Find all issues with less than 4 votes: [1] *Note: it is not possible to compare two fields.*	`votes < 4`
LESS THAN EQUALS: <= [1]	Find all issues with 4 or fewer votes: [1] *Note: it is not possible to compare two fields.*	`votes <= 4`
	Find all issues that have not been updated in the past month (30 days):	`updated <= "-4w 2d"`
MATCHES: ~ [1] [2]	Find all issues whose Summary matches the word "run" (or derivatives of that word, such as "running" or "ran"). [1] *For use with text fields only.* [2] *Can be used with* JIRA text-search syntax.	`summary ~ run`

IS [1]	Find all issues that have no Fix Version:
	[1] *Can only be used with EMPTY and NULL.*
		`fixVersion is empty`
		or
		`fixVersion is null`

IS NOT [1]	Find all issues that have one or more votes:
	[1] *Can only be used with EMPTY and NULL.*
		`votes is not empty`
		or
		`votes is not null`

ESCAPE CHARACTER: \	Find all issues whose Resolution is "Harry's":	`resolution = Harry\'s`
	Find all issues whose Summary contains "Sales\Marketing":	`summary ~ Sales\\Marketing`
WHITESPACE	Find all issues whose Summary contains "internet browser":	`summary ~ "\"internet browser\""`

EMPTY	Find all issues without a due date:	`duedate = empty`
		or
		`duedate is empty`

NULL	Find all issues without a due date:	`duedate = null`
		or
		`duedate is null`

ORDER BY [1]	Find all issues without a due date, sorted by creation date: [1] *Uses the field's default sorting order*.	`duedate = empty order by created`
	Find all issues without a due date, sorted by creation date, then by priority (highest to lowest):	`duedate = empty order by created, priority desc`

	Find all issues without a due date, sorted by creation date, then by priority (lowest to highest):	`duedate = empty order by created, priority asc`

Fields and Functions Reference

- Project
- Category
- Issue Key
- Affected Versions
- Fix Versions
- Components
- Issue Type
- Priority
- Status
- Resolution
- Issue Security Level
- Summary
- Description
- Environment
- Comment
- Assignee
- Reporter
- Parent [1]
- Original Estimate [1]
- Remaining Estimate [1]
- Time Spent [1]
- Work Ratio [1]
- Votes
- Created
- Updated
- Due
- Resolved
- Custom Field
- cascadeOption(*parentOption*, *childOption*)
- currentUser()
- membersOf(*GroupName*)
- now()
- releasedVersions(*Project)*
- unreleasedVersions(*Project)*
- standardIssueTypes()
- subtaskIssueTypes()
- linkedIssues(*IssueKey,LinkType*)
- votedIssues()
- watchedIssues()
- issueHistory()
- text
- filter

Field	Alias	Usage notes	Syntax
Project		Search by Project Name:	`project = "ABC Project"`
		Search by Project Key:	`project = "ABC"`
		Search by ID (i.e. the number that JIRA automatically allocates to a project):	`project = 1234`

Category		Search by Project Category:	`category = "Alphabet Projects"`
Issue Key	`issue` or `key`	Search by Issue Key:	`issueKey = "ABC-123"`
Affected Versions		Search by Version Name:	`affectedVersion = "3.14"` or `affectedVersion = "Big Ted"`
		Search by ID (i.e. the number that JIRA automatically allocates to a Version):	`affectedVersion = 10350`
Fix Versions		Search by Version Name:	`fixVersion in ("3.14", "4.2")` or `fixVersion = "Little Ted"`
		Search by ID (i.e. the number that JIRA automatically allocates to a version):	`fixVersion = 10000`
Components		Search by Component Name:	`component in (Comp1, Comp2)`
		Search by ID (i.e. the number that JIRA automatically allocates to a Component):	`component = 20500`
Issue Type	`type`	Search by Issue Type Name:	`issueType in (Bug,Improvement)` or `issueType = Bug`

		Search by ID (i.e. the number that JIRA automatically allocates to an Issue Type):	`issueType = 2`
Priority		Search by Priority Name:	`priority = High`
		Search by ID (i.e. the number that JIRA automatically allocates to a Priority):	`priority = 10000`
Status		Search by Status Name:	`status = Open`
		Search by ID (i.e. the number that JIRA automatically allocates to a Status):	`status = 1`
Resolution		Search by Resolution Name:	`resolution in ("Cannot Reproduce", "Won't Fix")`
		Search by ID (i.e. the number that JIRA automatically allocates to a Resolution):	`resolution = 1`
		Search for issues that do not have a Resolution:	`resolution = unresolved`
Issue Security Level		Search by Issue Security Level Name:	`level in ("Really High", level1)`
		Search by ID (i.e. the number that JIRA automatically allocates to an Issue Security Level):	`level = 123`
Summary		Specify search text, using "LIKE" ("~"): *(Note: JIRA text-search syntax can be used.)*	`summary ~ "Error saving file"`
Description		Specify search text, using "LIKE" ("~"): *(Note: JIRA text-search syntax can be used.)*	`description ~ "Please see screenshot for details."`

Environment		Specify search text, using "LIKE" ("~"): *(Note: JIRA text-search syntax can be used.)*	`environment ~ "Third floor"`
Comment		Specify search text, using "LIKE" ("~"): *(Note: JIRA text-search syntax can be used.)*	`comment ~ "My PC is quite old"`

Assignee		Search for issues that are assigned to a particular user(s), ie. any user(s) whose ID or Full Name matches your search:	`assignee = "John Smith"`
			or
			`assignee = jsmith`

Reporter		Search for issues that were created by a particular user(s), ie. any user(s) whose ID or Full Name matches your search:	`reporter = "Jill Jones"`
			or
			`reporter = jjones`

Parent [1]		Search for all sub-tasks of a particular issue: *[1]Only available if sub-tasks have been enabled by your JIRA administrator.*	`parent = "JRA-1234"`

Original Estimate [1]		Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes:
		[1]Only available if time-tracking has been enabled by your JIRA administrator.
			`originalEstimate = 1h`
			or
			`originalEstimate > 2d`

Remaining Estimate [1]		Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes: *[1]Only available if time-tracking has been enabled by your JIRA administrator.*	`remainingEstimate > 4h`
Time Spent [1]		Use 'w', 'd', 'h' and 'm' to specify weeks, days, hours or minutes: *[1]Only available if time-tracking has been enabled by your JIRA administrator.*	`timeSpent > 5d`
Work Ratio [1]		Search for issues with a particular Work Ratio[2], e.g. find issues on which more than 75% of the Original Estimate has been spent: *[1]Only available if time-tracking has been enabled by your JIRA administrator.* *[2]Work Ratio = (Time Spent / Original Estimate) x 100*	`workratio > 75`

Votes		Search by number of votes, e.g. find all issues with 12 or more votes:	`votes >= 12`

Created	`createdDate`	Search for issues that were created on, before or after a particular date[1] (or date range), e.g. find all issues created on or before 12th December 2008 00:00:
		
		[1]Use one of the following date formats:
		
		`yyyy/MM/dd HH:mm`
		`yyyy-MM-dd HH:mm`
		`yyyy/MM/dd`
		`yyyy-MM-dd`
		
		Or use 'w', 'd', 'h' and 'm' to specify a Created Date relative to the current time.
			`created <= "2008/12/12"`

		Find issues created less than one day ago:	`created > "-1d"`

Updated	`updatedDate`	Search for issues that were last updated on, before or after a particular date[1] (or date range), e.g. find all issues updated on or before 12th December 2008 00:00:
		
		[1]Use one of the following date formats:
		
		`yyyy/MM/dd HH:mm`
		`yyyy-MM-dd HH:mm`
		`yyyy/MM/dd`
		`yyyy-MM-dd`
		
		Or use 'w' (weeks), 'd' (days), 'h' (hours) and 'm' (minutes) to specify an Updated Date relative to the current time.
			`updated <= "2008/12/12"`

		Find issues that updated more than two weeks ago:	`updated < "-2w"`

Due	`dueDate`	Search for issues that were due on, before or after a particular date[1] (or date range), e.g. find all issues due on or before 31st December 2008[2]:
		
		[1]Use one of the following date formats:
		
		`yyyy/MM/dd HH:mm`
		`yyyy-MM-dd HH:mm`
		`yyyy/MM/dd`
		`yyyy-MM-dd`
		
		Or use 'w' (weeks), 'd' (days), 'h' (hours) and 'm' (minutes)to specify a Due Date relative to the current time.
		*[2]Note that Due Date relates to the **date** only (not to the time).*
			`due <= "2008/12/31"`

		Find issues that are due tomorrow:	`due = "1d"`

Resolved	`resolutionDate`	Search for issues resolved on, before or after a particular date[1] (or date range), e.g. find all issues due on or before 31st December 2008 00:00; or issues that were resolved in January 2009:
		
		[1]Use one of the following date formats:
		
		`yyyy/MM/dd HH:mm`
		`yyyy-MM-dd HH:mm`
		`yyyy/MM/dd`
		`yyyy-MM-dd`
		
		Or use 'w' (weeks), 'd' (days), 'h' (hours) and 'm' (minutes) to specify a Resolution Date relative to the current time.
			`resolved <= "2008/12/31"`
			
			or
			
			`resolved > "2008/12/31" and resolved < "2009/02/01"`

		Find issues that were resolved in the last hour:	`resolved > -1h`

Custom Field		Search for issues where a particular custom field (e.g. "Location") isn't blank, or where it has a particular value:
		
		(Note: JIRA text-search syntax can be used with custom fields of type 'Text'.)
			`location != empty`
			
			or
			
			`location = "New York"`
			
			or
			
			`location in ("London", "Milan","Paris")`

		Search by ID[1] (i.e. the number that JIRA automatically allocates to a custom field), e.g.: *[1]This is useful if you have a custom field with the same name as a built-in JIRA system field.*	`cf[10003] = "New York"`
cascadeOption(*parentOption*, *childOption*)		Find issues that match the selected values of a 'cascading select' custom field. The *parentoption* parameter matches against the first tier of options in cascading select. The *childoption* parameter matches against the second tier of options in the cascading select and is optional. E.g. find issues where a custom field ("Location") has the value "USA" for the first tier and "New York" for the second tier:	`location in cascadeOption("USA","New York")`
		E.g. find issues where a custom field ("Location") has the value "USA" for the first tier and any value (or no value) for the second tier:	`location in cascadeOption("USA")`
		E.g. find issues where a custom field ("Location") has the value "USA" for the first tier and no value for the second tier:	`location in cascadeOption("USA" ,none)`

		E.g. find issues where a custom field ("Location") has no value for the first tier and no value for the second tier:	`location in cascadeOption(none)`
		E.g. find issues where a custom field ("Referrer") has the value "none" for the first tier and "none" for the second tier:	`referrer in cascadeOption("\"none"\",\"none"\")`
		E.g. find issues where a custom field ("Referrer") has the value "none" for the first tier and no value for the second tier:	`referrer in cascadeOption("\"none"\",none)`
currentUser()		Perform searches based on the currently logged-in user, e.g. find issues that are assigned to me:	`assignee = currentUser()`
		Find issues that were reported to me but are not assigned to me:	`reporter = currentUser() and assignee != currentUser()`
membersOf(*GroupName*)		Perform searches based on the members of a particular group, e.g. search for issues where the Assignee is a member of the group "jira-developers":	`assignee in membersOf("jira-developers")`
		Search through multiple groups and a specific user, e.g:	`reporter in membersOf("jira-developers") or reporter in membersOf("jira-administrators") or reporter=jsmith`
		Search for a particular group, but exclude a particular member or members, e.g.:	`assignee in membersOf(QA) and assignee not in ("John Smith","Jill Jones")`
		Exclude members of a particular group:	`assignee not in membersOf(QA)`
now()		Perform searches based on the current time[1], e.g. find issues that were logged in the last eight hours: *[1]You can use `'w'` (week), `'d'` (day), `'h'` (hours) and `'m'` (minutes) to refine your search.*	`created > "-8h"`

		Find issues that are overdue:	`duedate < now() and status not in (closed, resolved)`
releasedVersions(*Project)*		Perform searches based on the released versions of a specified project, e.g. find issues whose Fix For Version is a released version of the ABC project:	`fixVersion in releasedVersions(ABC)`
		Find issues that relate to released versions of the JIRA project:	`affectedVersion in releasedVersions(ABC) or fixVersion in releasedVersions(ABC)`
unreleasedVersions(*Project)*		Perform searches based on the unreleased versions of a specified project, e.g. find issues whose Fix For Version is an unreleased version of the ABC project:	`fixVersion in unreleasedVersions(ABC)`
		Find issues that relate to unreleased versions of the JIRA project:	`affectedVersion in unreleasedVersions(ABC) or fixVersion in unreleasedVersions(ABC)`
standardIssueTypes()		Find issues that are not subtasks (i.e. issues whose Issue Type is a standard issue type, not a subtask issue type):	`issuetype in standardIssueTypes()`
subtaskIssueTypes()		Find issues that are subtasks (i.e. issues whose Issue Type is a subtask issue type):	`issuetype in subtaskIssueTypes()`
linkedIssues(*IssueKey,LinkType*)		Find issues that are linked to a particular issue:	`issue in linkedIssues("ABC-123")`
		Find issues that are linked to a particular issue via a particular type of link: *(Note that LinkType is case-sensitive.)*	`issue in linkedIssues("ABC-123","is duplicated by")`
votedIssues()		Find issues that you have voted for:	`issue in votedIssues()`

watchedIssues()		Find issues that you are watching:	`issue in watchedIssues()`
issueHistory()		Find issues that you have recently viewed, i.e. issues that are in your History drop-down: *Note: issueHistory() returns up to 50 issues, whereas the History drop-down returns only 20.* *Note: if you are not logged in to JIRA, your History will only include issues from your current browser session.*	`issue in issueHistory()`
text		Find issues where a text field (i.e. Summary, Description, Environment, Comments or a custom field of type 'text') matches your specified text, e.g. "Fred": *Note: The `text` clause can only be used with the `MATCHES` operator `"~"`*	`text ~ "Fred"`

filter		Use a saved filter to narrow your search, e.g. search the results of the filter "My Saved Filter" (which has an ID of 12000) for issues assigned to the user jsmith:
		Note: An ORDER BY statement in your specified query will override an ORDER BY statement in the saved filter.
			`filter = "My Saved Filter" and assignee = jsmith`
			or
			`filter = 12000 and assignee = jsmith`

Setting Precedence of Operators

You can use parentheses in complex JQL statements to enforce the precedence of operators.

For example, if you want to find all resolved issues in the SysAdmin project as well as all issues (any status, any project) currently assigned to the system administrator (bobsmith), you can use parentheses to enforce the precedence of the boolean operators in your query, i.e.:

```
(status=resolved AND project=SysAdmin) OR assignee=bobsmith
```

Searching for Strings

You will need to use quote-marks if your string contains any of the following:

- space (e.g. "office move")
- plus-sign (e.g. "A+B")
- full-stop (e.g. "3.13")
- comma (e.g. "A,B,C")
- hyphen (e.g. "ABC-123")
- single quote-mark (e.g. "Harry's")

If your string contains a double quote-mark, you will need to precede it with the escape character (back-slash), e.g.: "Type your name in the \"Login\" box"

Performing Text Searches

You can use Lucene's text searching features when performing searches on the following fields:

- Summary
- Description
- Environment
- Comment
- Custom fields of type 'Text'

For details, please see the page on Performing Text Searches, which includes the following sections:

- Query Terms

- Term Modifiers
    - Wildcard Searches: ? and *
    - Fuzzy Searches: ~
    - Proximity Searches
- Boosting a Term: ^
- Boolean Operators
    - OR
    - AND
    - Required term: +
    - NOT
    - Excluded term: -
- Grouping
- Escaping Special Characters: \
- Limitations
    - Whole words only

Using Auto-complete

As you type your query, JIRA will recognise the context and offer a list of "auto-complete" suggestions as follows:

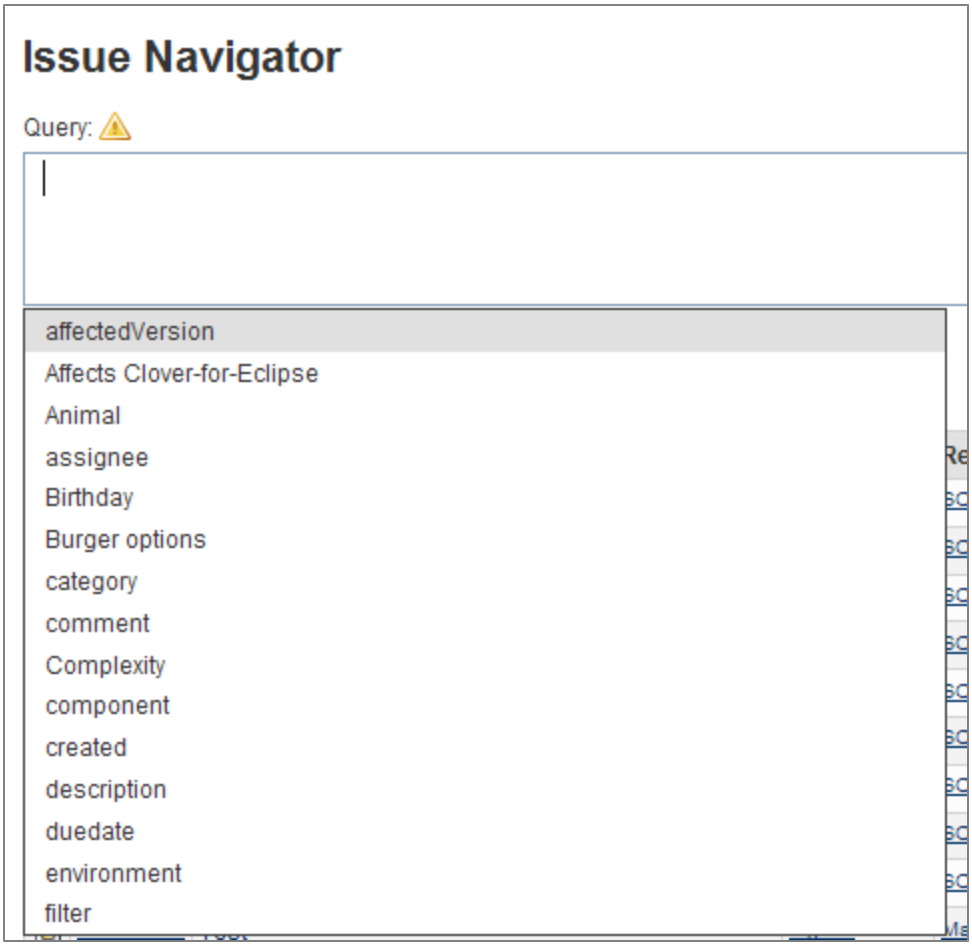
- If you type a space at the start of your query...
- If you type one or more characters...
- If you type a field or function, then a space...
- If you type a field[1], then an operator, then a space...
- If you type a field[1], then an operator, then one or more characters...

The list of auto-complete suggestions is displayed alphabetically and includes the first 15 matches. Note that auto-complete suggestions are not offered for statements within parentheses.

> **Please note:**
> If no auto-complete suggestions are offered, your administrator may have disabled the 'JQL Auto-complete' feature for your JIRA instance.

If you type a space at the start of your query...

...JIRA will offer a list of all available fields and functions, e.g.:

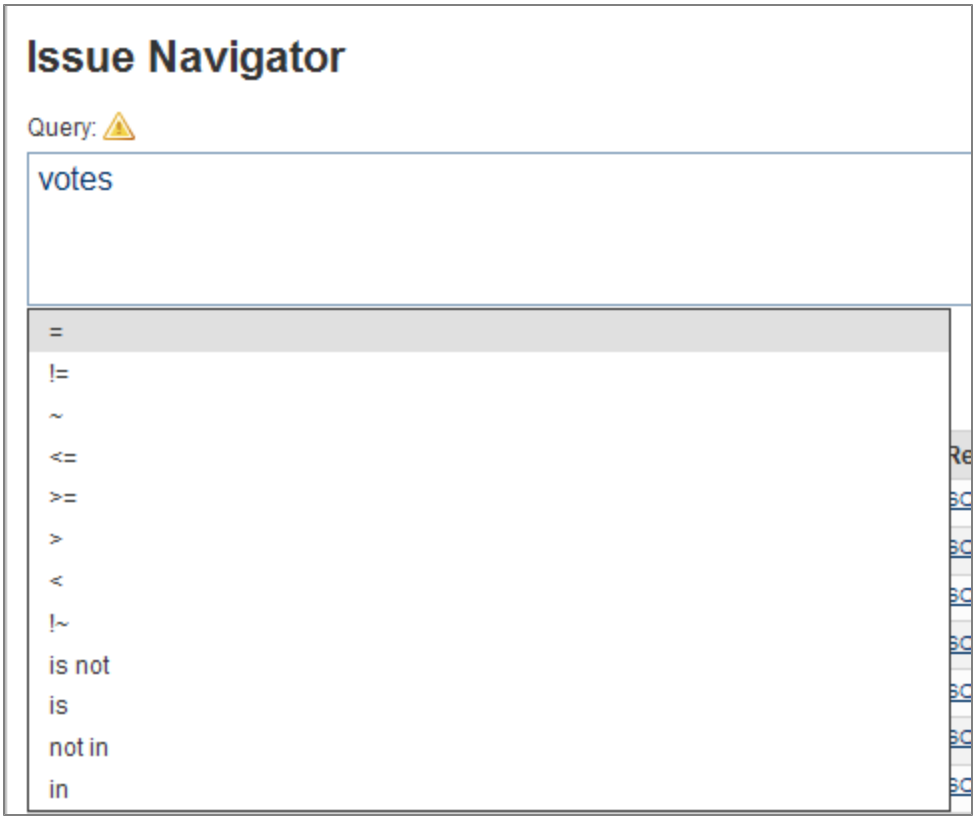


If you type one or more characters...

...JIRA will offer a list of matching fields and functions, e.g.:

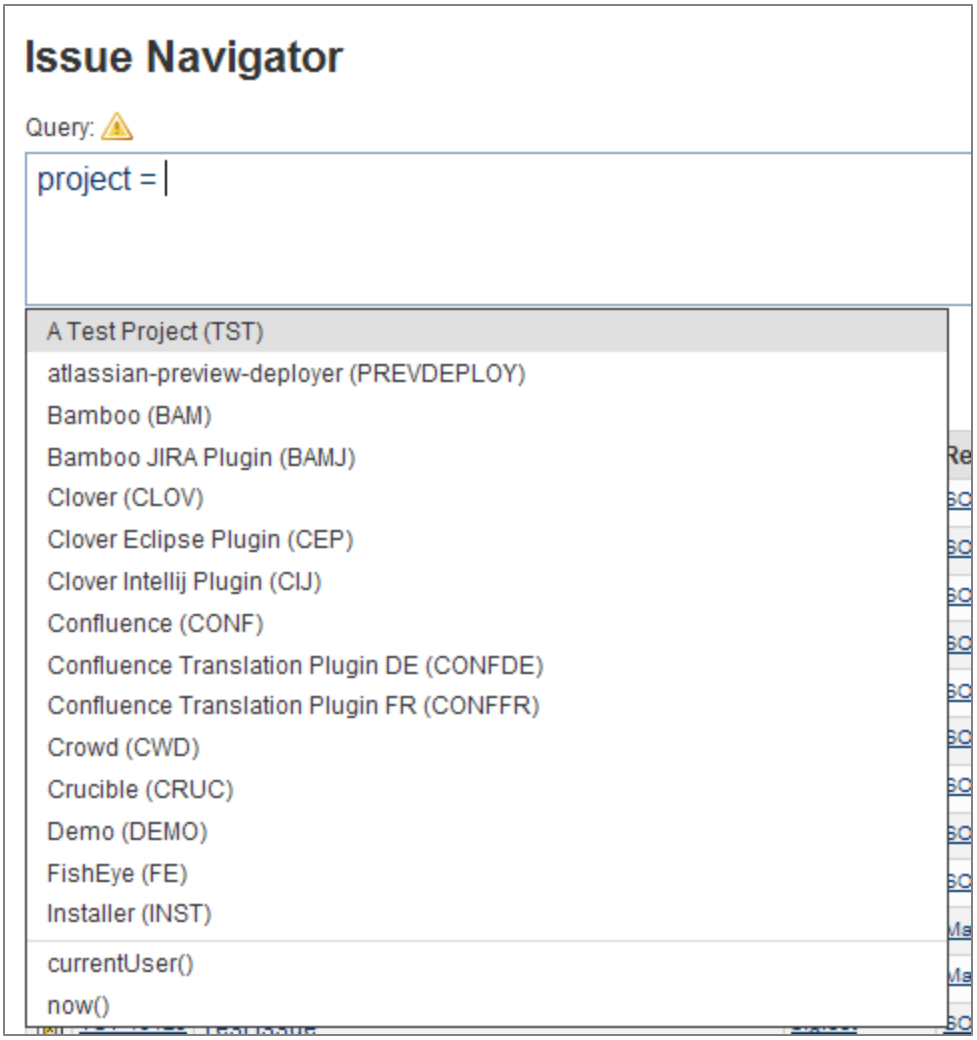
If you type a field or function, then a space...

...JIRA will offer a list of valid operators, e.g.:



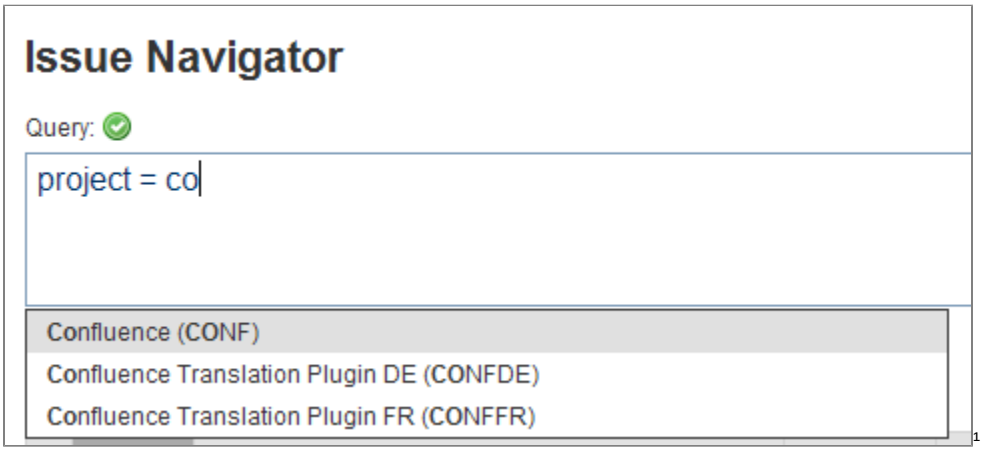
If you type a field[1], then an operator, then a space...

...JIRA will offer a list of valid values, e.g.:



If you type a field[1], then an operator, then one or more characters...

...JIRA will offer a list of matching values, e.g.:



[1] applies only to 'Project', 'Assignee', 'Reporter' and custom fields.

JQL Reserved Words

JQL has a list of reserved words. These words need to be surrounded by quotation marks if you wish to use them in queries:

"abort", "access", "add", "after", "alias", "all", "alter", "and", "any", "as", "asc",
"audit", "avg", "before", "begin", "between", "boolean", "break", "by", "byte", "catch", "cf",
"char", "character", "check", "checkpoint", "collate", "collation", "column", "commit", "connect", "continue",

"count", "create", "current", "date", "decimal", "declare", "decrement", "default", "defaults", "define", "delete",
"delimiter", "desc", "difference", "distinct", "divide", "do", "double", "drop", "else", "empty", "encoding",
"end", "equals", "escape", "exclusive", "exec", "execute", "exists", "explain", "false", "fetch", "file", "field",
"first", "float", "for", "from", "function", "go", "goto", "grant", "greater", "group", "having",
"identified", "if", "immediate", "in", "increment", "index", "initial", "inner", "inout", "input", "insert",
"int", "integer", "intersect", "intersection", "into", "is", "isempty", "isnull", "join", "last", "left",
"less", "like", "limit", "lock", "long", "max", "min", "minus", "mode", "modify",
"modulo", "more", "multiply", "next", "noaudit", "not", "notin", "nowait", "null", "number", "object",
"of", "on", "option", "or", "order", "outer", "output", "power", "previous", "prior", "privileges",
"public", "raise", "raw", "remainder", "rename", "resource", "return", "returns", "revoke", "right", "row",
"rowid", "rownum", "rows", "select", "session", "set", "share", "size", "sqrt", "start", "strict",
"string", "subtract", "sum", "synonym", "table", "then", "to", "trans", "transaction", "trigger", "true",
"uid", "union", "unique", "update", "user", "validate", "values", "view", "when", "whenever", "where",
"while", "with"

(The list is hard coded in the JqlStringSupportImpl.java file.)

Saving Searches ('Issue Filters')

The utility of JIRA's powerful issue search functionality is further enhanced by the ability to *save* a search for use at a later time. A saved search is called an **issue filter.**

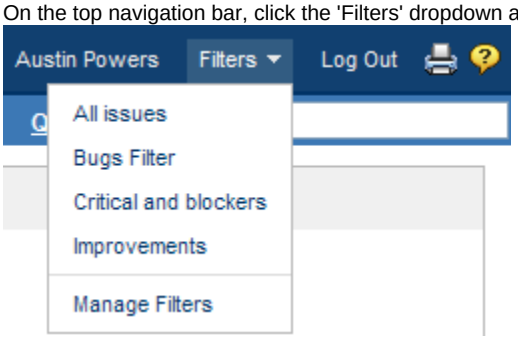
With an issue filter you can:

- display the search results in the Issue Navigator, where you can view and export them in various formats (RSS, Excel, etc)
- display the search results in a report format
- display the search results in a dashboard portlet
- share the search with colleagues (see below)
- add another user's shared filter as a favourite (see below)
- have the search results emailed to you according to your preferred schedule

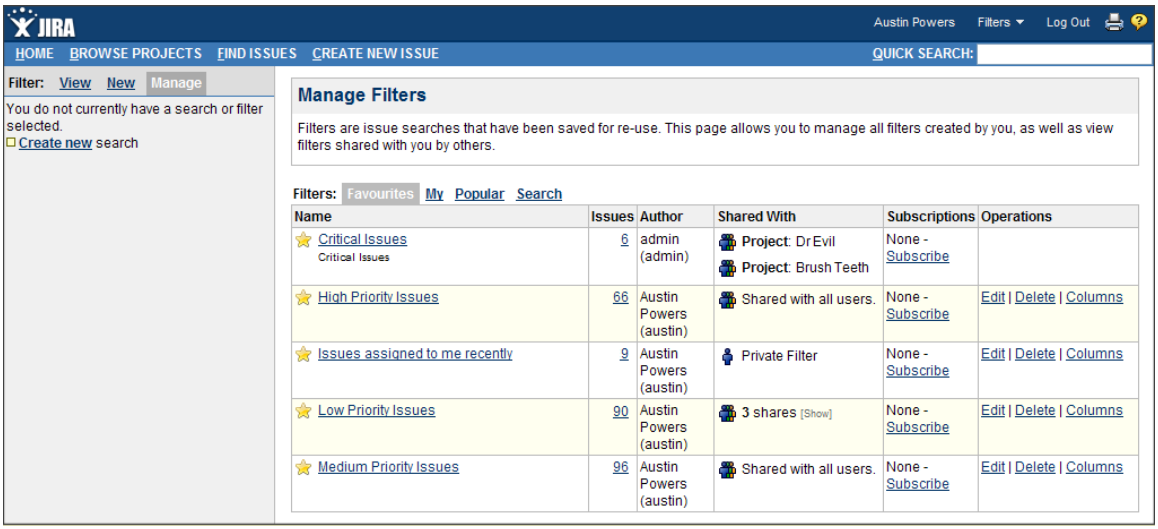
Managing your Issue Filters

The Manage Filters page allows you to view and configure filters that you have created, as well as work with filters that other users have shared with you.

1. On the top navigation bar, click the 'Filters' dropdown and select 'Manage Filters' from the list.



2. The 'Manage Filters' page will display. From this page, you can perform the functions listed below:
    - Create a new search to be saved as a filter.
    - Add a filter as a favourite.
    - Share a filter that you have created with other users.
    - Search for filters that has been created by you or shared with you by other users.
    - Update an existing filter's details or edit a filter's search criteria for a filter that you have created.



Saving a search to an Issue Filter

1. Refine and run your search as described in 'Searching for Issues'.
2. Click the '**Save it as a filter**' link at the top of the search form.

3. The 'Save Current Filter' page will display. Provide a name for the new issue filter and optionally enter a short description.

4. Your new filter will be added as a favourite filter by default upon creation. If you do not wish this filter to be added as a favourite, deselect the star icon. You can add the filter as a favourite after it has been created. Read more about adding an existing filter as a Favourite.
5. The sharing of your new filter is defaulted, depending on your sharing preference in your user profile. If you have not specified a personal preference, then the global default for sharing will apply (i.e. 'Private', unless changed by your JIRA Administrator under 'User Defaults' in the Administration menu). If you wish to change the sharing of your filter, refer to the instructions on sharing filters below.

> Please note, you need the 'Create Shared Object' global permission to be able to share your filter. If you cannot see any issue sharing functionality, contact your JIRA Administrator to have this permission added to your profile.

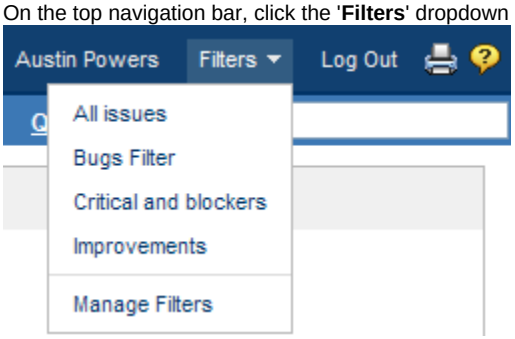
Adding an Issue Filter as a Favourite

Issue filters that you created or that have been shared by other people can be added as a favourite filter. This means that the filter will appear in the 'Filters' dropdown in the top menu, as well as, display in the 'Favourite Filters' portlet on your dashboard (if you have the portlet added to your

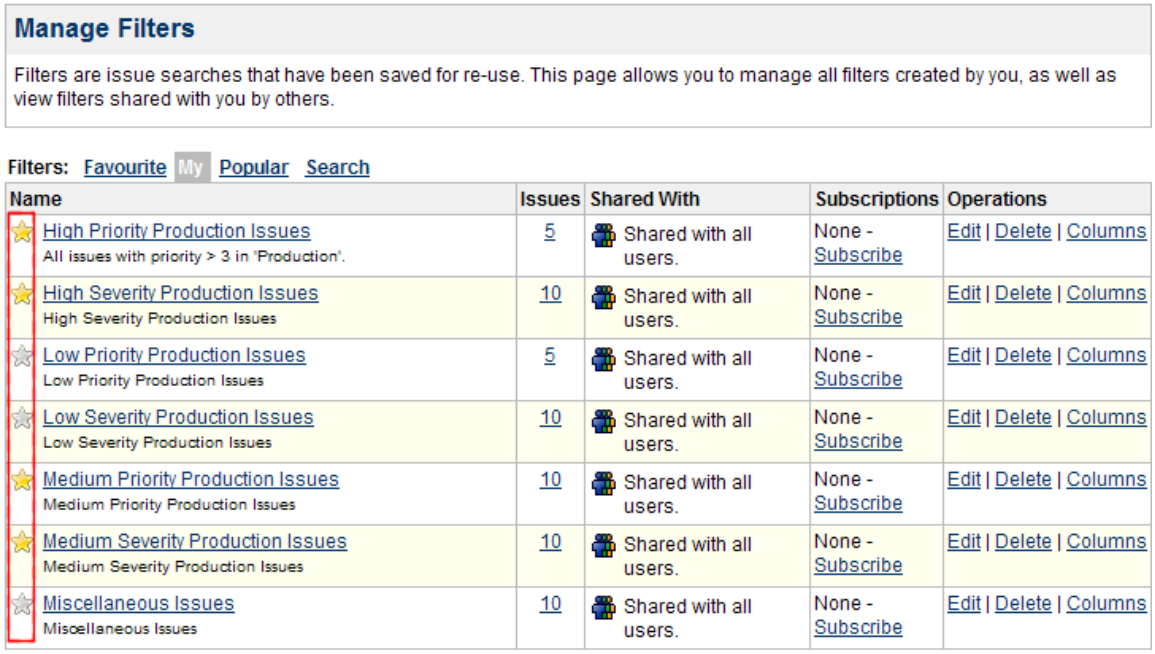
dashboard).

Follow the steps below to add an existing shared filter as a favourite:

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. Locate the filter you wish to add as a favourite. If you created the filter, it will be listed under the 'My' tab, otherwise you can search for filters shared by other users via the 'Search' tab.
3. Click the star icon next to the filter name to select it as a favourite.

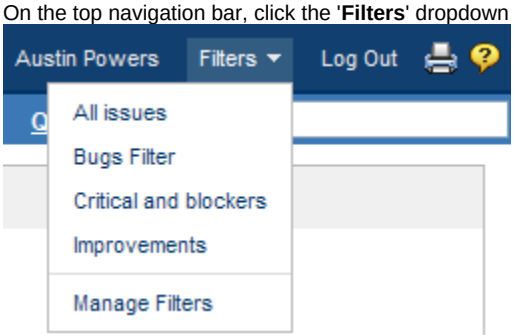


Sharing an existing Issue Filter

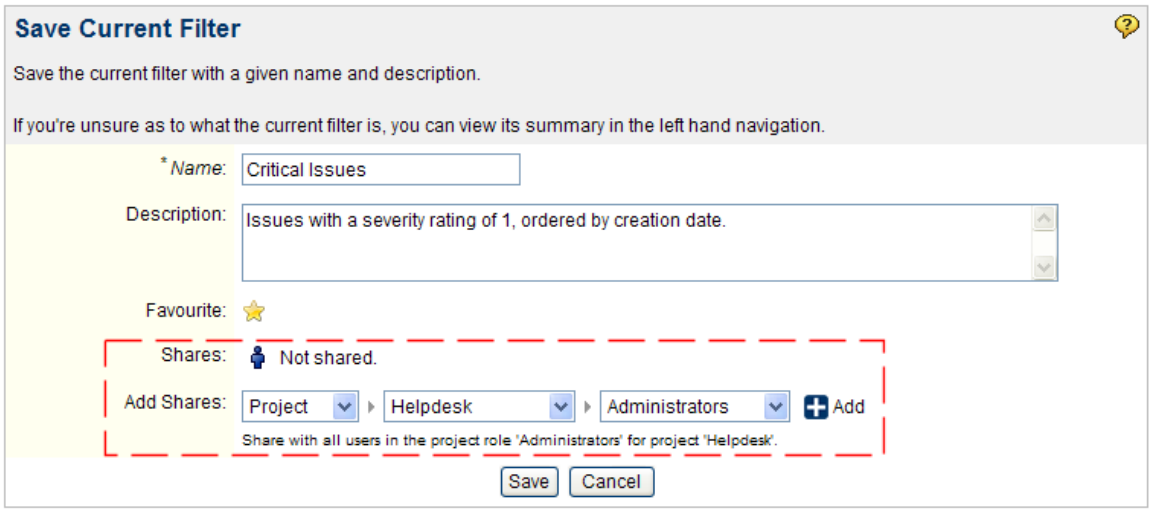
Issue filters that you have created can be shared with other users via user groups, projects and project roles. Issue filters can also be shared globally.

Follow the steps below to share an existing issue filter:

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. Locate the filter you wish to share under the '**My**' tab, and click the '**Edit**' link in the '**Operations**' column.
3. Select the group, project or project role that you want to share the filter with, or share it with all users, if you wish. Click the '**Add**' link to add the share. You can add further share permissions if you wish.



4. Click '**Save**' to save your changes.

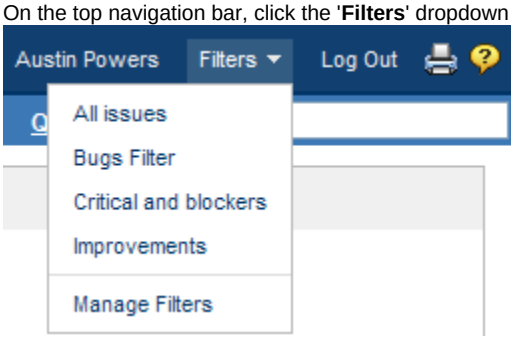
> The ability to share filters is only available for the Professional and Enterprise editions of JIRA. If you are using the Professional or Enterprise edition of JIRA and the filter sharing functions described above are not available to you, you probably do not have the 'Create Shared Object' global permission assigned to you. Please contact your JIRA administrator to obtain this permission.

Finding an existing Issue Filter

Issue filters that you have created or have been shared by other users can be found via the issue filter search function of the 'Manage Filters' page. If the filter has been added as a favourite by many users, you also may be able locate it on the 'Popular' tab of the 'Manage Filters' page. This tab lists the top twenty most popular filters, counted by the number of users that have selected the filter as a favourite.

Follow the steps below to search for an existing issue filter:

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. Click the '**Search**' tab. The issue filter Search will display. Enter your search criteria and click '**Search**' to run the search.

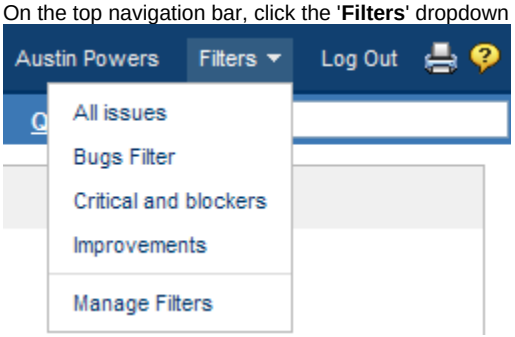
3. Your search results will be displayed on the same page. Click the name of any issue filter to run it and select it as your current filter. You can also sort the search results by any of the columns, by clicking the column headers.

Updating an existing Issue Filter's details

You can always update the details, i.e. Name, Description, Sharing, Favourite, of an existing Issue Filter after its creation. Please note that you can only update the details of Issue Filters which you have created.

Follow the steps below to update the details of one of your existing issue filters:

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. Click the '**My**' tab. This tab displays all the filters that have been created by you.
3. Locate the filter you wish to update, and click the '**Edit**' link in the 'Operations' column.
4. The 'Save Current Filter' page will display. Update the filter details as required. If you wish to change the sharing or favourite settings for the filter, refer to the relevant instructions above.

5. Click the '**Save**' button to save your changes.

> The ability to share filters is only available for the Professional and Enterprise editions of JIRA. If you are using the Professional or Enterprise edition of JIRA and the filter sharing functions described above are not available to you, you probably do not have the 'Create Shared Object' global permission assigned to you. Please contact your JIRA administrator to obtain this permission.

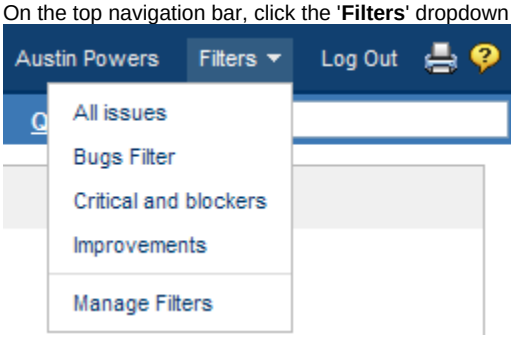
Editing/Cloning an existing Issue Filter

The search criteria of an existing issue filter can always be changed after creation by editing the issue filter. You can also clone an existing issue filter via the edit function.

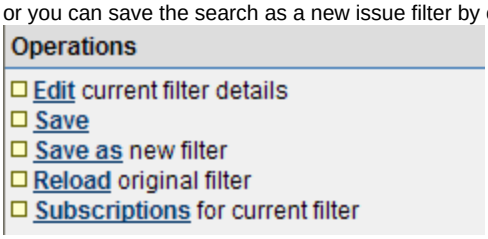
Issue filters that have been created by you or shared with you by other users can be edited and/or cloned.

Follow the steps below to edit and/or clone an existing issue filter:

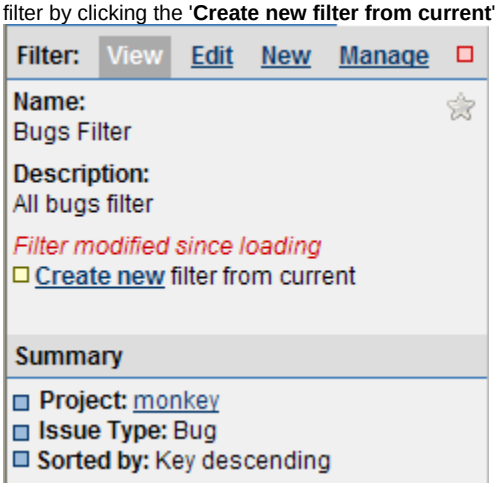
1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. At the top of the search form, click '**Manage**'.
3. Locate the filter you wish to edit/clone, and click the name of the issue filter to run it and select it as your current filter.
4. Click the '**Edit**' link at the top of the left hand menu. The left hand menu will refresh and the search criteria of the filter will display.
5. If you wish to clone the issue filter without modifying the search criteria, click the '**Create new filter from current**' link. The 'Save Current Filter' screen will display and you can save the filter as described above.
6. If you wish to change the search criteria, update the search terms as desired and click the '**View**' button to run the filter. The search results will display in the Issue Navigator on the right hand side of the page.
7. If you wish to save the modified search, click '**View & Hide**' button. The filter operations will be displayed on the right hand side of the page, as follows:
    - If the filter was created by you, you can choose to overwrite the filter with the modified search criteria by clicking the '**Save**' link, or you can save the search as a new issue filter by clicking the '**Save as new filter**' link.



    - If the filter was not created by you (i.e. shared by another user), you will only have the option to save the search as a new issue filter by clicking the '**Create new filter from current**' link.



Defining filter-specific Issue Navigator Column Order

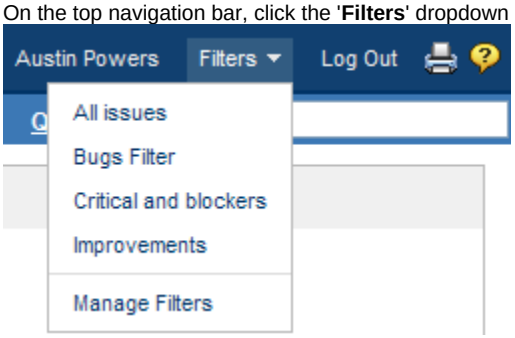
In JIRA Enterprise edition it is possible to add an Issue Navigator Column Order to a saved filter. The results of a filter are displayed using the saved column order, if the filter has one. Otherwise the results are shown using the user's personal column order or the system default.

> To display your configured column order in a filter subscription, you must select 'HTML' for the 'Outgoing email format' in your User Profile. If you are receiving text emails from JIRA, you will not be able to see your configured column order.

Adding a Column Order

To add a column layout to a saved filter,

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.

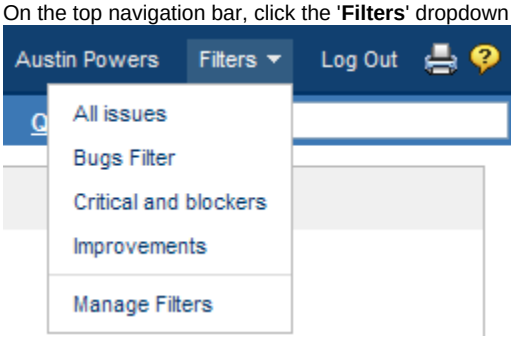


2. Locate the filter you wish to share under the '**My**' tab, and click the '**Columns**' link in the '**Operations**' column.
3. Configure the column order as desired. You can configure the column order the same way you would configure your personal Issue Navigator column order.

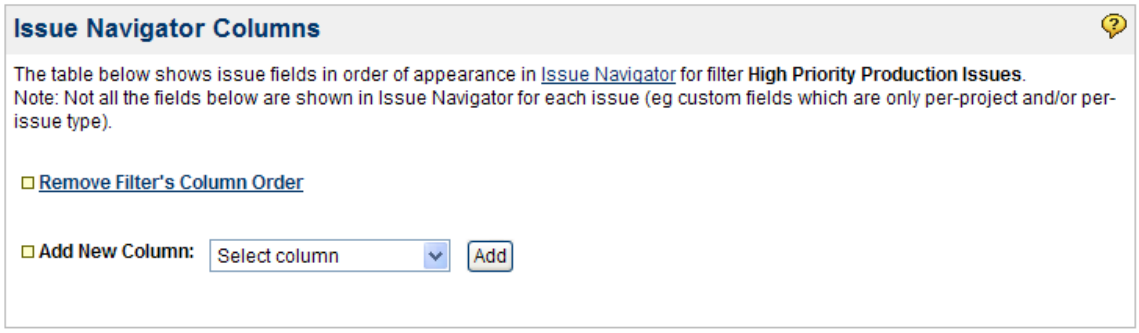
Removing Column Order

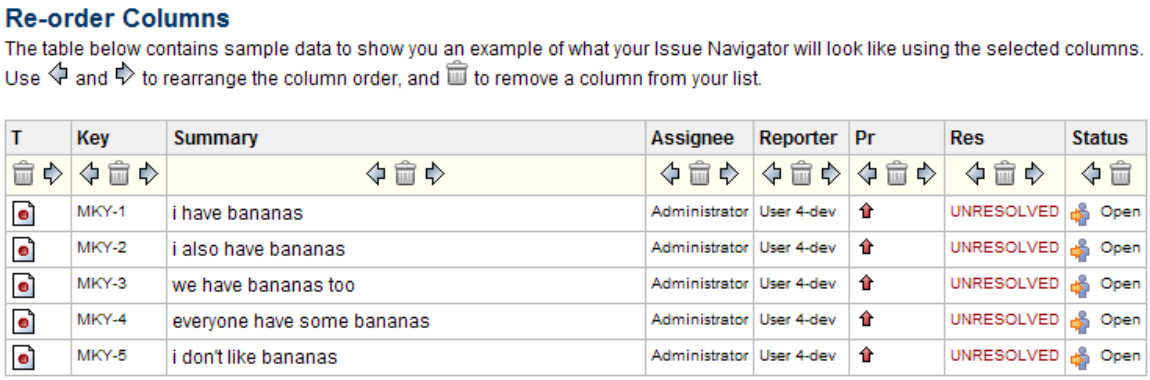
To remove a filter's saved column layout,

1. On the top navigation bar, click the '**Filters**' dropdown and select '**Manage Filters**' from the list.



2. Locate the filter you wish to share under the '**My**' tab, and click the '**Columns**' link in the '**Operations**' column.
3. Click the '**Remove Filter's Column Order**' link. The default column order will be restored.





Overriding Column Order

If a filter has a saved column order, the results will be presented using that column order when the filter is run. You can, however, choose to use your own column order (or the system default column order, if you do not have a personal one configured) to view the results. To do this, click the '**Use your default Column Order**' link on the right of the Issue Navigator search results screen.

To go back to using the filter's own column order, select the '**Use filter's Column Order**' link.

Additional Resources

- Creating SLA issue filters tutorial video — Watch this short tutorial video to see how to create an issue filter to support an SLA (service level agreement). Please note the JIRA version and JIRA edition of the tutorial video before watching.

Receiving Search Results via Email

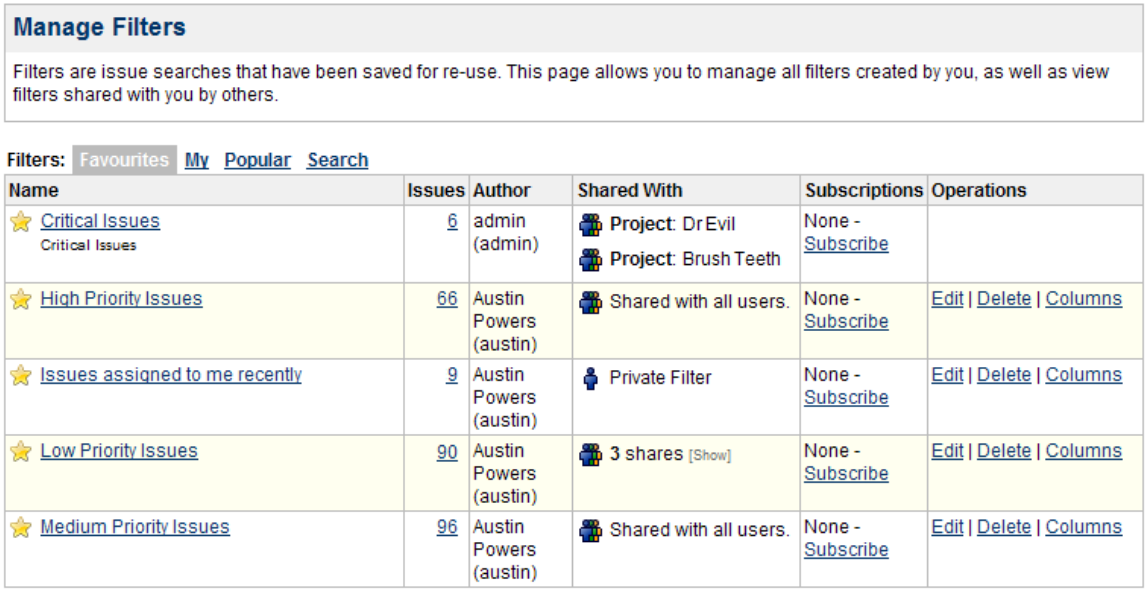
JIRA enables you to *subscribe* to an issue filter (a saved search). JIRA will then run the search according to your specified schedule, and will email the results to you.

You can specify when and how often you would like to receive the search results, e.g. 'Every hour between 9.00AM-5.00PM, Monday-Friday', or 'The last Friday of every month at 7.00AM'.

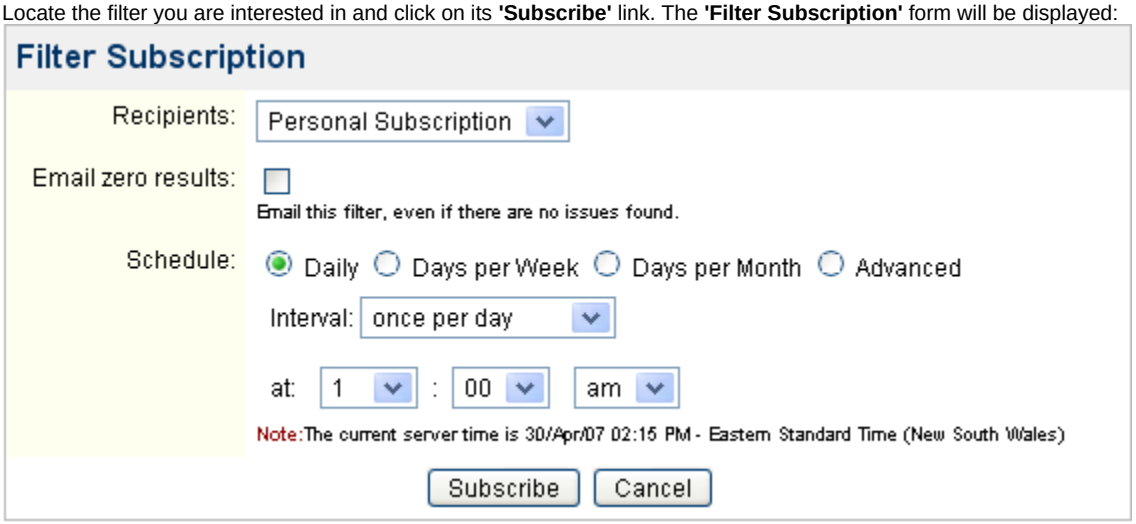
> Emails can only be sent if your administrator has configured an SMTP mail server.

Subscribing to an Issue Filter

1. On the top navigation bar, click on the **'Find Issues'** tab.
2. At the top of the search form, at the left of the screen, click **'Manage'.**
3. A list of available filters will be displayed:



4. Locate the filter you are interested in and click on its **'Subscribe'** link. The **'Filter Subscription'** form will be displayed:



5. In the **'Recipients'** box, either choose 'Personal Subscription' (if you only wish to subscribe yourself), or select a group of recipients from the drop-down list. Note that you cannot select a *group* unless your JIRA administrator has granted you the 'Manage Group Filter Subscriptions' global permission.
6. Select **'Email zero results'** if you would like the email to always be sent, even if there are no search results at that time.
7. Select one of the following types of schedule:
    - **'Daily'** — choose this if you want to receive one or more emails *every day.*
    - **'Days per Week'** — choose this if you want to receive one or more emails on *particular days of every week.*
    - **'Days per Month'** — choose this if you want to receive an email on *a particular day of every month.*
    - **'Advanced'** — see *Advanced scheduling ('cron')* below.
8. Click the '**Subscribe**' button.
9. You will now be shown a subscription summary page. If you wish, you can click **'Run now'** to test your subscription.

Advanced scheduling ('cron')

You can use a **'Cron Expression'** to specify a custom schedule to suit your particular requirements.

Cron expressions consist of the following fields, separated by spaces:

Field	Allowed values	Allowed special characters
Second	0-59	, - * /
Minute	0-59	, - * /
Hour	0-23	, - * /
Day-of-month	1-31	, - * / ? L W C

Month	1-12 or JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * / ? L C #
Year *(optional)*	1970-2099	, - * /

The special characters operate as follows:

Special character	Usage
,	Specifies a list of values. For example, in the **Day-of-week** field, 'MON,WED,FRI' means 'every Monday, Wednesday, and Friday'.
-	Specifies a range of values. For example, in the **Day-of-week** field, 'MON-FRI' means 'every Monday, Tuesday, Wednesday, Thursday and Friday'.
*	Specifies all possible values. For example, in the **Hour** field, '*' means 'every hour of the day'.
/	Specifies increments to the given value. For example, in the **Minute** field, '0/15' means 'every 15 minutes during the hour, starting at minute zero'.
?	Specifies no particular value. This is useful when you need to specify a value for one of the two fields **Day-of-month** or **Day-of-week,** but not the other.
L	Specifies the last possible value; this has different meanings depending on context. In the **Day-of-week** field, 'L' on its own means 'the last day of every week' (i.e. 'every Saturday'), or if used after another value, means 'the last xxx day of the month' (e.g. 'SATL' and '7L' both mean 'the last Saturday of the month). In the **Day-of-month** field, 'L' on its own means 'the last day of the month', or 'LW' means 'the last weekday of the month'.
W	Specifies the weekday (Monday-Friday) nearest the given day of the month. For example, '1W' means 'the nearest weekday to the 1st of the month' (note that if the 1st is a Saturday, the email will be sent on the nearest weekday *within the same month,* i.e. on Monday 3rd). 'W' can only be used when the day-of-month is a single day, not a range or list of days.
#	Specifies the nth occurrence of a given day of the week. For example, 'TUES#2' (or '3#2') means 'the second Tuesday of the month'.

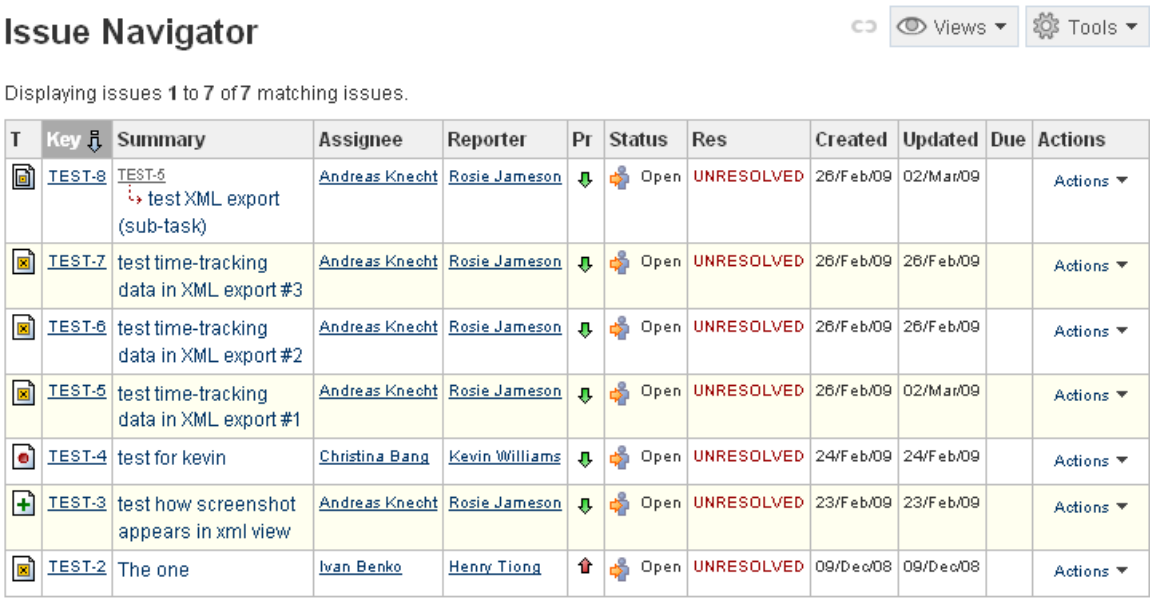
Here are some sample cron expressions:

```
*0 15 8 ? * ** Every day at 8.15 pm.
 *0 15 8 * * ?* Every day at 8.15 am.
 *0 * 14 * * ?* Every minute starting at 2.00 pm and ending at 2:59 pm, every day.
 *0 0/5 14 * * ?* Every 5 minutes starting at 2.00 pm and ending at 2:55 pm, every day.
 *0 0/5 14,18 * * ?* Every 5 minutes starting at 2.00 pm and ending at 2:55 pm, AND every 5 minutes
starting at 6.00 pm and ending at 6:55 pm, every day.
 *0 0-5 14 * * ?* Every minute starting at 2.00 pm and ending at 2:05 pm, every day.
 *0 10,44 14 ? 3 WED* 2:10 pm and 2:44 pm every Wednesday in the month of March.
 *0 15 8 ? * MON-FRI* 8:15 am every Monday, Tuesday, Wednesday, Thursday and Friday.
 *0 15 8 15 * ?* 8:15 am on the 15th day of every month.
 *0 15 8 L * ?* 8:15 am on the last day of every month.
 *0 15 8 LW * ?* 8:15 am on the last weekday of every month.
 *0 15 8 ? * 6L* 8:15 am on the last Friday of every month.
 *0 15 8 ? * 6#2* 8:15 am on the second Friday of every month.
 *0 15 8 ? * 6#2 2007-2009* 8:15 am on the second Friday of every month during the years 2007, 2008
and 2009.
```

Note: cron expressions are not case-sensitive.

Using the Issue Navigator

The Issue Navigator displays the search results from an issue filter, a Quick Search or an Advanced Search, e.g:



Issue Navigator

Views Tools

Displaying issues 1 to 7 of 7 matching issues.

T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due	Actions
	TEST-8	TEST-6 test XML export (sub-task)	Andreas Knecht	Rosie Jameson		Open	UNRESOLVED	26/Feb/09	02/Mar/09		Actions
	TEST-7	test time-tracking data in XML export #3	Andreas Knecht	Rosie Jameson		Open	UNRESOLVED	26/Feb/09	26/Feb/09		Actions
	TEST-6	test time-tracking data in XML export #2	Andreas Knecht	Rosie Jameson		Open	UNRESOLVED	26/Feb/09	26/Feb/09		Actions
	TEST-5	test time-tracking data in XML export #1	Andreas Knecht	Rosie Jameson		Open	UNRESOLVED	26/Feb/09	02/Mar/09		Actions
	TEST-4	test for kevin	Christina Bang	Kevin Williams		Open	UNRESOLVED	24/Feb/09	24/Feb/09		Actions
	TEST-3	test how screenshot appears in xml view	Andreas Knecht	Rosie Jameson		Open	UNRESOLVED	23/Feb/09	23/Feb/09		Actions
	TEST-2	The one	Ivan Benko	Henry Tiong		Open	UNRESOLVED	09/Dec/08	09/Dec/08		Actions

The Issue Navigator provides convenient ways to perform many of JIRA's most useful functions. You can:

- Use the **Actions** menu for a particular issue to quickly:
  - **Resolve issue** — Set the issue's Status to **'Resolved'** and select the appropriate Resolution.
  - **Close issue** — Set a resolved issue's Status to **'Closed'.**
  - **Reopen issue** — Set a resolved or closed issue's Status to **'Reopened'.**
  - **Assign this issue** — Select an asignee for the issue.
  - **Assign this issue to me** — Assign the issue to yourself.
  - **Attach file to this issue** — Select a file, upload it and attach it to the issue.
  - **Clone this issue** — Create a new issue which is a copy of the issue.
  - **Comment on this issue** — Add a comment to the issue.
  - **Delete this issue** — Permanently remove the issue.
  - **Edit this issue** — Edit the issue's details (**Summary, Description,** etc).
  - **Link this issue** — Create a link between the issue and another issue.
  - **Move this issue** — Move the issue to a different project.
  - **Create sub-task for this issue** — Create a new issue which is a sub-task of the issue.
  - **Convert sub-task to issue** — If the issue is a sub-task, convert it to a standalone issue.
  - **Start watching this issue** — Become a watcher of the issue.
  - **Log work for this issue** — Record the work done and time spent on the issue.

> Note that the options in the **Actions** menu will only be available if you have the necessary permissions, and that options which change an issue's Status (**Resolve issue, Close issue, Reopen issue**) may differ from this list, depending on your organisation's workflow.

- Use the **Views** menu to view/export your search results in various formats:
  - **Printable** — All search results on one page, with one row of data per issue. Includes the issue fields that are currently configured in your Issue Navigator.
  - **Full Content** — All search results on one page. Includes **Description, Comments** and all other issue data, not just the issue fields that are currently configured in your Issue Navigator.
  - **XML** — An XML view of issue data, suitable for use with the Confluence JIRA Issues Macro. (Also suitable for use as an RSS 0.9.2 feed). For details, see 'Displaying Search Results in XML'.
  - **RSS (Issues)** — An RSS 2.0 feed of issue data, suitable for displaying in an RSS reader. For details, see 'Receiving Search Results via RSS'.
  - **RSS (Comments)** — An RSS 2.0 feed of comments on the issues, suitable for displaying in an RSS reader. For details, see ' Receiving Search Results via RSS'.
  - **Word** — An MS Word document. Includes **Description, Comments** and all other issue data, not just the issue fields that are currently configured in your Issue Navigator. Can be saved and/or edited. For details, see 'Exporting Search Results to Microsoft Word'.
  - **Excel (All fields)** — An MS Excel spreadsheet. Includes **Description** and all other issue data (excluding comments). Can be saved, or edited to produce graphs and charts; also useful for basic reporting and statistics. For details, see 'Exporting Search Results to Microsoft Excel'.
  - **Excel (Current fields)** — An MS Excel spreadsheet containing the issue fields that are currently configured in your Issue Navigator. Can be saved, or edited to produce graphs and charts; also useful for basic reporting and statistics. For details, see ' Exporting Search Results to Microsoft Excel'.
  - **Charts** — A graphical summary of the search results, which can be saved as a dashboard portlet. For details, see 'Displaying Search Results as a Chart'.
- Use the **Tools** menu to:
  - **Bulk Change** — You can move, edit or delete multiple issues at once as described in 'Modifying Multiple ('Bulk') Issues'. (Note

that this option will only be available if you have been given the necessary permission.)
    - **Configure Columns** — You can add, remove and re-order columns as described in 'Customising your Issue Navigator'.
- Click the **'permalink'** icon:

  to create a permanent URL link to your search results.

> The Issue Navigator only displays one set of search results at any one time, even if you have multiple browser windows open. However, you can easily save your searches (see 'issue filters'), then display them as needed.

Re-ordering the search results

When viewing search results in the Issue Navigator, you can re-order the issues by clicking on the column header*. For example, if you click the **'Reporter'** column header, the Issue Navigator will re-display the issues in ascending order of reporter's name. If you click the **'Reporter'** column header a second time, the Issue Navigator will re-display the issues in *descending* order of reporter's name.

* With some exceptions, e.g. the 'Images' column and the sub-task aggregate columns (i.e. all columns beginning with '') are non-orderable.

To choose different fields to display in your Issue Navigator, see Customising your Issue Navigator

Viewing individual issues

When an issue from a search result set is selected and displayed, a mini-navigator is shown at the right of the issue's title bar:

This mini-navigator displays the current issue's position within the result set. It also provides a link to the next and previous issue in the result set, along with a link to return to the search results. You can navigate through the search results by using the shortcut keys `[JIRADOC:ALT]+P` (Previous) and `[JIRADOC:ALT]+N` (Next).

Accessing protected data

When accessing data generated from JIRA, you may find that access to some resources requires user authentication (i.e. requires you to login). There are three options for this:

1. To enable access to data without logging in, your JIRA administrator may specify the 'Browse' permission for **Anyone.**
2. You can provide the parameters **os_username** and **os_password** in the request URL (e.g. http://jira.atlassian.com/browse/TST-1?os_username=tester&os_password=tstpassword). The problem with this method is that it transmits your username and password across the wire in clear text, which may not be an option for some users.
3. You can provide the request parameter **os_authType=basic** (e.g. http://mycompany.com/anypage?os_authType=basic). This will force the server to issue a challenge for user credentials (i.e. a login prompt) via the basic http authentication protocol. If you are running over SSL, you still need to specify the **os_authType=basic** parameter if you require the user to authenticate.

Customising your Issue Navigator

The Issue Navigator displays the search results from an issue filter, a Quick Search or an Advanced Search.

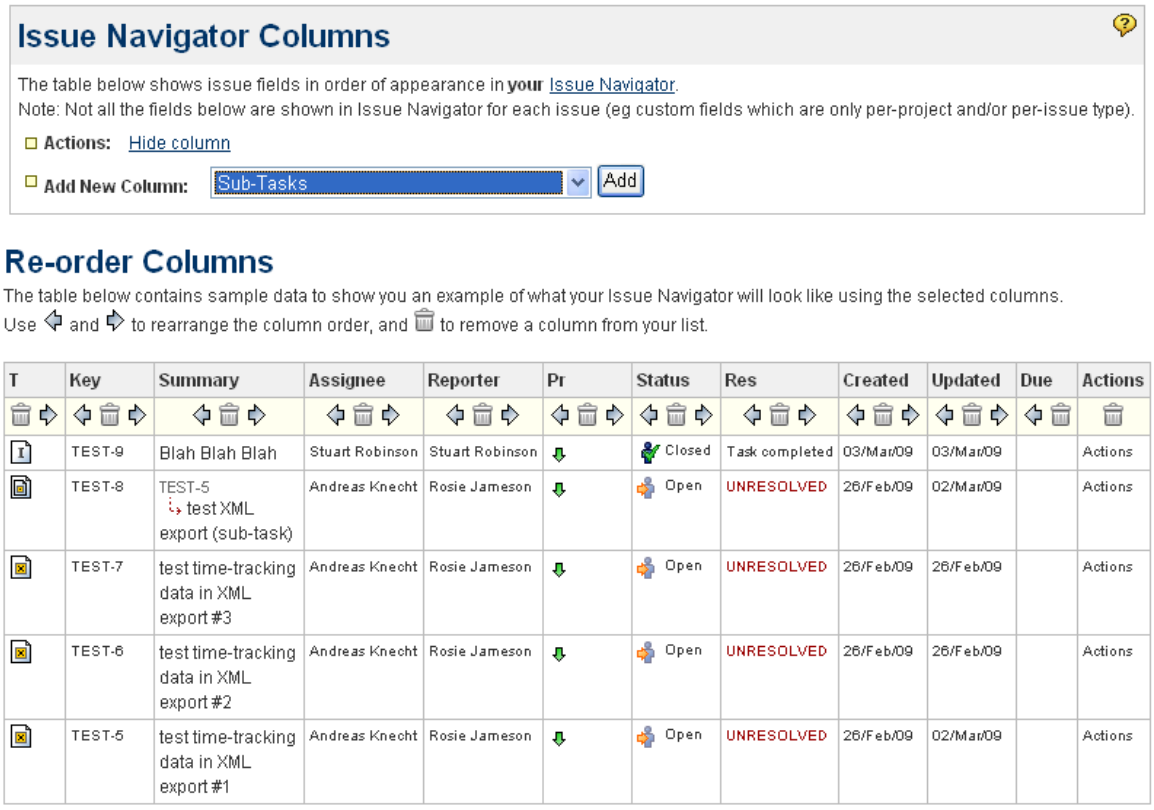
You can customise your Issue Navigator by choosing:

- which **columns** (i.e. issue fields) to display
- how many **rows** (i.e. issues) to display

Customising your Issue Navigator columns

To choose which columns (i.e. issue fields) to display in your Issue Navigator,

1. Click the **'Find Issues'** link at the top of the screen. The Issue Navigator will be displayed.
2. From the **'Tools'** menu at the right of the screen (above the search results), select **'Configure Columns'.** The following will be displayed:



- **To move a column left or right,** click on the left-arrow or right-arrow icon that appears under the column's heading.
- **To remove a column from the list,** click the bin icon which appears under the column's heading.
- **To add a column to the list,** select the issue field name from the drop-down box titled **'Add New Column'** and click the **'Add'** button. The column will appear as the right-most column in the list. You can then position the column where desired by using the arrow icons.
- **To hide the *'Actions'** column,* click the **'Hide Column'** link.

> In JIRA Enterprise edition it is also possible to configure filter-specific column order for saved filters.

Customising your Issue Navigator rows

To choose how many rows (i.e. issues) to display on each page of your Issue Navigator:

1. Click your user name at the top-right of the page to open your User Profile.
2. Click **'Edit Preferences',** under **'Operations'** in the left-hand column of the screen.
3. Enter your preferred **'Number of Issues displayed per Issue Navigator page'**, (The default is 50).
4. Click the **'Update'** button.

See also

Using the Issue Navigator

Displaying Search Results in XML

The JIRA Issue Navigator enables you to display your search results in XML format, suitable for use with the Confluence JIRA Issues Macro, for example. (Also suitable for use as an RSS 0.9.2 feed. Note: for an RSS 2.0 feed, please see ' Receiving Search Results via RSS '.)

You can choose which fields to include in the XML output, as described below.

Exporting to XML

To display your search results in XML format,

1. On the top navigation bar, click the **'Find Issues'** tab.
2. Refine your search, as described in *'Searching for Issues',* until the required results are displayed in the Issue Navigator.
3. Click the **'Views'** menu, and select **'XML'.**

4. Your search results will be displayed in XML format. Note:
    - If you wish to restrict which fields are included in the XML output, use the * `field` * parameter as described below.
    - To choose how many issues are included in the XML output, change the value of the * `tempMax` * parameter. The default is 1,000 issues (i.e. `tempMax=1000`).
5. Copy the URL that is currently displayed on your screen.
6. If you are using the Confluence **JIRA Issues Macro,** paste the URL into your Confluence document as described in the Confluence documentation.

Choosing which fields to include

To restrict which issue fields are returned in the XML export, specify the `field` parameter in your URL. For example, to include only the **Issue key** and **Summary,** add `&field=key&field=summary` to the URL.

Note:

- If the `field` parameter is not specified, the XML output will include *all* the issue fields.
- If one or more `field` parameters are specified, the XML output will contain only the **Issue key** plus your chosen field(s).

The complete list of available values for the `field` parameter is as follows:

Value	Sample XML output
`title`	`<title>[TEST-4] This is a test</title>`
`link`	`<link>https://extranet.atlassian.com:443/jira/browse/TEST-4</link>`
`project` (or **`pid`**)	`<project id="10330" key="TST">Test</project>`
`description`	`<description>This is a detailed description of the issue.<description>`
`environment`	`<environment>Sydney network<environment>`
`key`	`<key id="22574">TEST-4</key>`
`summary`	`<summary>This is a test<summary>`

type (or **issuetype**)	<type id="3" iconUrl="https://extranet.atlassian.com:443/jira/images/icons/task.gif">Task</type>
parent	<parent id="22620">TEST-5</parent>
priority	<priority id="4" iconUrl="https://extranet.atlassian.com:443/jira/images/icons/priority_minor.gif">Minor<priority>
status	<status id="5" iconUrl="https://extranet.atlassian.com:443/jira/images/icons/status_resolved.gif">Resolved</status>
resolution	<resolution id="1">Fixed</resolution>
assignee	<assignee username="jsmith">John Smith</assignee>
reporter	<assignee username="jsmith">John Smith</assignee>
security	<security id="10021">Private</security>
created	<created>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)>/created>
updated	<updated>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)>/updated>

resolved (or **resolutiondate**)	`<resolved>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)>/resolved>`
due (or **duedate**)	`<due>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)>/due>`
version (or **versions**)	`<version>2.4.7</version>`
fixfor (or **fixVersions**)	`<fixVersion>2.6</fixVersion>`
component (or **components**)	`<component>Documentation</component>`
votes	`<votes>1</votes>`

comments (or **comment**)	`<comments>`
	`  <comment id="39270" author="jsmith" created="Tue, 24 Feb 2009 16:45:02 -0600 (CST)">this looks familiar</comment>`
	`  <comment id="39273" author="jbrown" created="Tue, 24 Feb 2009 16:48:16 -0600 (CST)">to me too</comment>`
	`</comments>`

attachments (or **attachment**)	`<attachments>`
	`  <attachment id="30318" name="Issue Navigator - Atlassian JIRA-2.png" size="16161" author="yoz" created="Mon, 9 Feb 2009 13:32:58 -0600 (CST)"/>`
	`  <attachment id="30323" name="Windows XP (with Firefox 3.0).jpg" size="5802" author="vbharara" created="Tue, 10 Feb 2009 00:30:11 -0600 (CST)"/>`
	`</attachments>`

timeoriginalestimate	`<timeoriginalestimate seconds="600">10 minutes</timeoriginalestimate>`

timeestimate	<timeestimate seconds="300">5 minutes</timeestimate>
timespent	<timespent seconds="300">5 minutes</timespent>
aggregatetimeoriginalestimate	<aggregatetimeoriginalestimate seconds="36000">10 hours</aggregatetimeoriginalestimate>
aggregatetimeestimate	<aggregatetimeremainingestimate seconds="18000">5 hours</aggregatetimeremainingestimate>
aggregatetimespent	<aggregatetimespent seconds="18000">5 hours</aggregatetimespent>
timetracking	<timeoriginalestimate seconds="600">10 minutes</timeoriginalestimate> <timeestimate seconds="300">5 minutes</timeestimate> <timespent seconds="300">5 minutes</timespent> <aggregatetimeoriginalestimate seconds="36000">10 hours</aggregatetimeoriginalestimate> <aggregatetimeremainingestimate seconds="18000">5 hours</aggregatetimeremainingestimate> <aggregatetimespent seconds="18000">5 hours</aggregatetimespent>
issuelinks	<issuelinks> <issuelinktype id="10020"> <name>Duplicate</name> <inwardlinks description="is duplicated by"> <issuelink> <issuekey id="22477">INTSYS-1009</issuekey> </issuelink> </inwardlinks> </issuelinktype> </issuelinks>
subtasks (or **subtask**)	<subtasks> <subtask id="22623">TEST-8</subtask> </subtasks>

customfield_xxxxx	<customfields> <customfield id="customfield_10112" key="com.atlassian.jira.plugin.system.customfieldtypes:select"> <customfieldname>Department</customfieldname> <customfieldvalues> <customfieldvalue>Adminstration</customfieldvalue> </customfieldvalues> </customfield> </customfields>
allcustom	<customfields> <customfield id="customfield_10112" key="com.atlassian.jira.plugin.system.customfieldtypes:select"> <customfieldname>Department</customfieldname> <customfieldvalues> <customfieldvalue>Adminstration</customfieldvalue> </customfieldvalues> </customfield> <customfield id="customfield_10111" key="com.atlassian.jira.plugin.system.customfieldtypes:select"> <customfieldname>Expenditure Type</customfieldname> <customfieldvalues> <customfieldvalue>Operating</customfieldvalue> </customfieldvalues> </customfield> </customfields>

Receiving Search Results as an RSS Feed

JIRA enables you to subscribe to an RSS 2.0 feed that is based on any issue filter (saved search), or on your chosen search results, as displayed in the Issue Navigator.

You can choose either an RSS feed that contains issue data **(Summary, Description,** etc), or one that contains comments.

Note that the **tempMax** parameter can be used to control the maximum number of issues that are returned, e.g. sample RSS feed of the latest 15 issues reported on jira.atlassian.com.

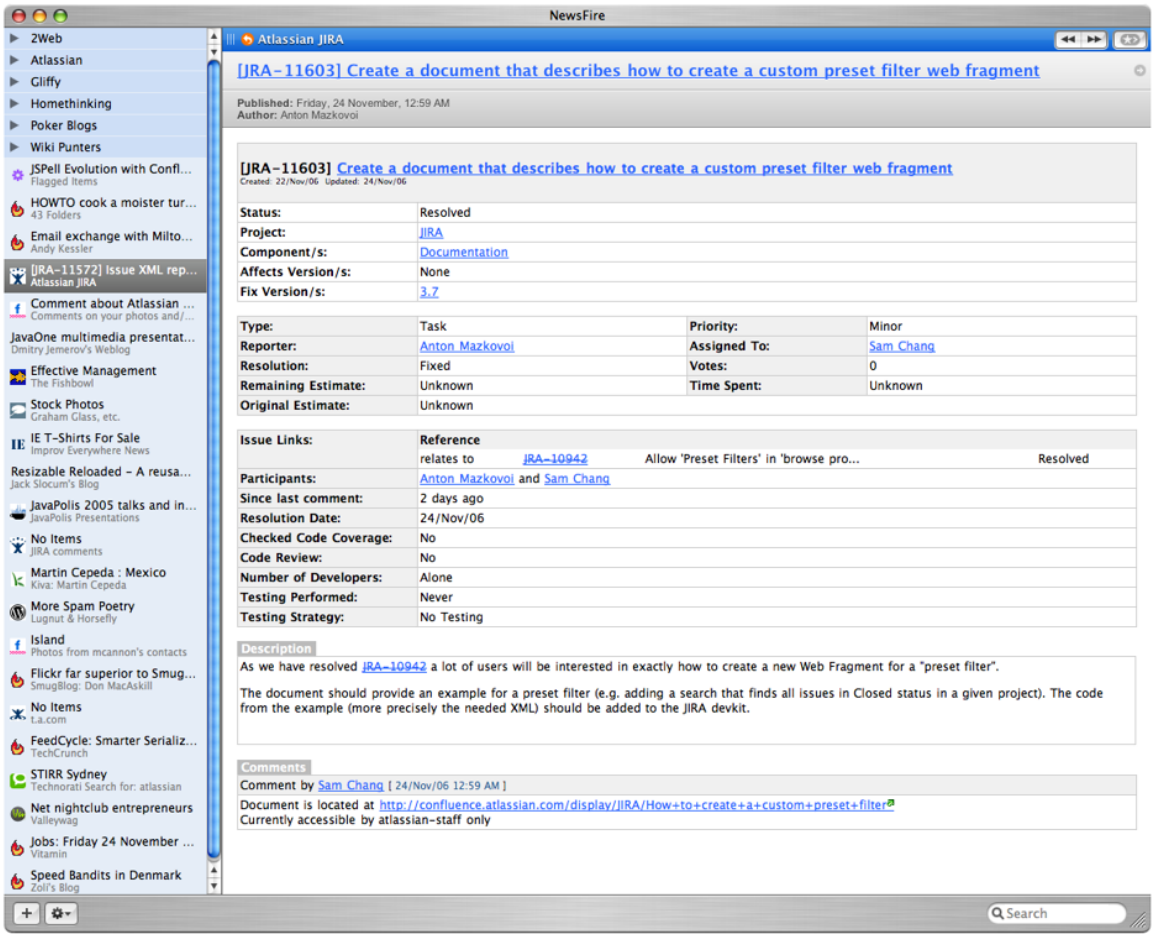
Subscribing to an RSS Feed

To subscribe to an RSS feed,

1. On the top navigation bar, click on the **'Find Issues'** tab.
2. Refine your search, as described in *'Searching for Issues',* until the required results are displayed in the Issue Navigator.
3. Click the **'Views'** menu, and select one of the following:
    - **'RSS (Issues)'** - this will create an RSS feed that contains just issue data.
    - **'RSS (Comments)'** - this will create an RSS feed that contains comments. HINT: To only receive current comments, use the **Updated, Updated After** and **Updated Before** fields in the **'Dates and Times'** section of the search form. E.g. to only receive comments created in the last week, type `-1w` in the **From** field.
4. Copy the URL that is currently displayed on your **Issue Navigator** screen.
    - NOTE: The **tempMax** parameter can be used to control the maximum number of issues returned in your RSS feed.
5. Paste the URL into your RSS reader.
    - **NOTE:** If you are getting an empty RSS feed, make sure you have logged in to JIRA - see *'Accessing protected data'* (below).

Example

Here is a sample RSS feed:



Accessing protected data

When accessing data generated from JIRA, you may find that access to some resources requires user authentication (i.e. requires you to login). There are three options for this:

1. To enable access to data without logging in, your JIRA administrator may specify the 'Browse' permission for **Anyone.**
2. You can provide the parameters **os_username** and **os_password** in the request URL (e.g. http://jira.atlassian.com/browse/TST-1?os_username=tester&os_password=tstpassword). The problem with this method is that it transmits your username and password across the wire in clear text, which may not be an option for some users.
3. You can provide the request parameter **os_authType=basic** (e.g. http://mycompany.com/anypage?os_authType=basic). This will force the server to issue a challenge for user credentials (i.e. a login prompt) via the basic http authentication protocol. If you are running over SSL, you still need to specify the **os_authType=basic** parameter if you require the user to authenticate.

See also

Exporting Search Results to XML

Adding the Activity Stream Gadget

Exporting Search Results to Microsoft Word

JIRA enables you to easily export your search results from the Issue Navigator to Microsoft Word. This can be a useful way to create reports in your own customised format.

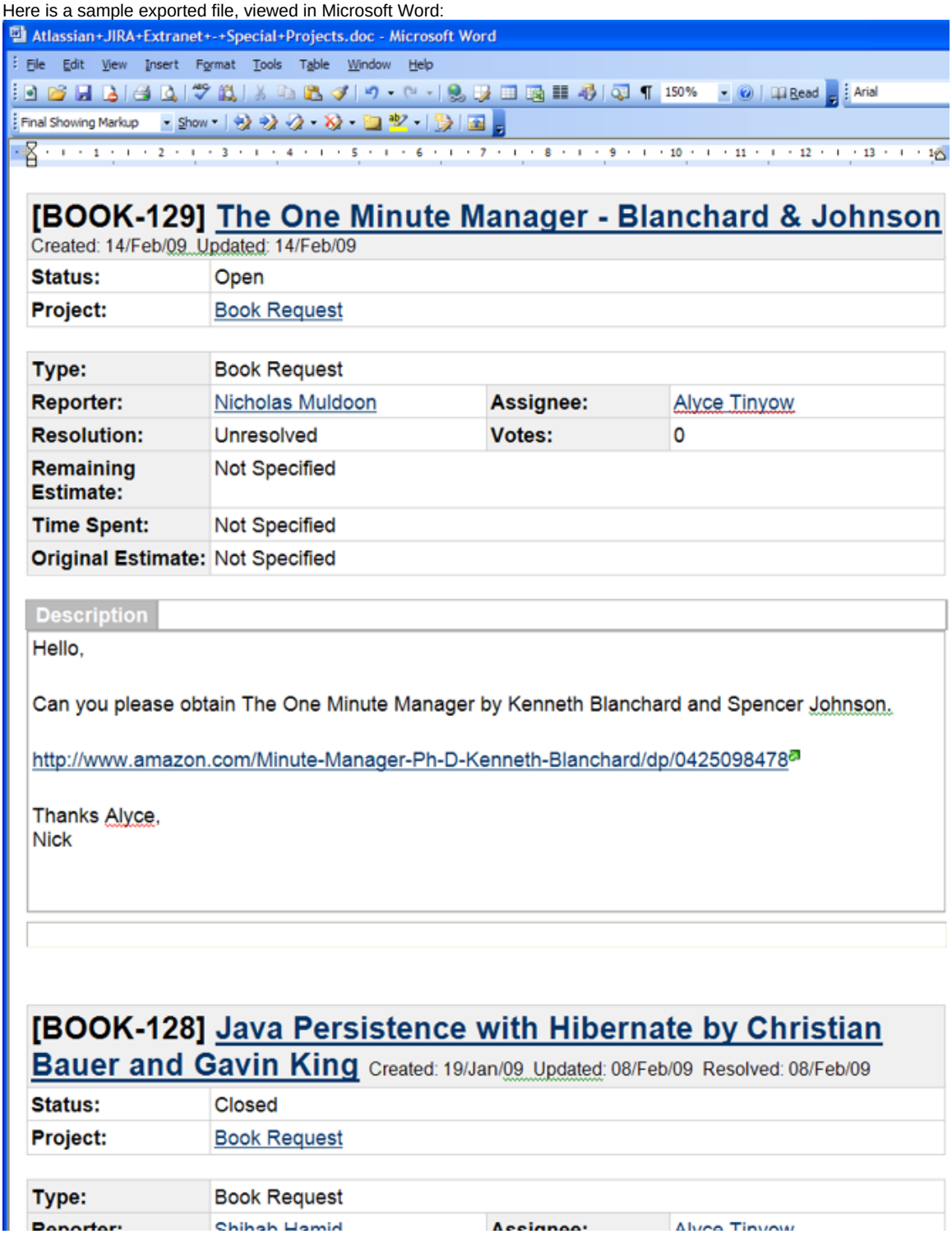
Exporting to Microsoft Word

1. On the top navigation bar, click on the **'Find Issues'** tab.
2. Refine your search, as described in *'Searching for Issues',* until the required results are displayed in the Issue Navigator.
   Please note:
   - The export will include Description, Comments and all other issue data, not just the issue fields that are currently configured in your Issue Navigator.
   - Large exports (e.g. hundreds of issues) are not recommended.
3. Click the **'Views'** menu, and select **'Word'.**

4. A file called **'<My company's JIRA> - <My project>.doc'** will be created. Edit this file using Microsoft Word and/or save it as required.

Example

Here is a sample exported file, viewed in Microsoft Word:



See also

Exporting Search Results to Microsoft Excel

Exporting Search Results to Microsoft Excel

JIRA enables you to easily export your search results from the Issue Navigator to Microsoft Excel. This can be a useful way to format data and

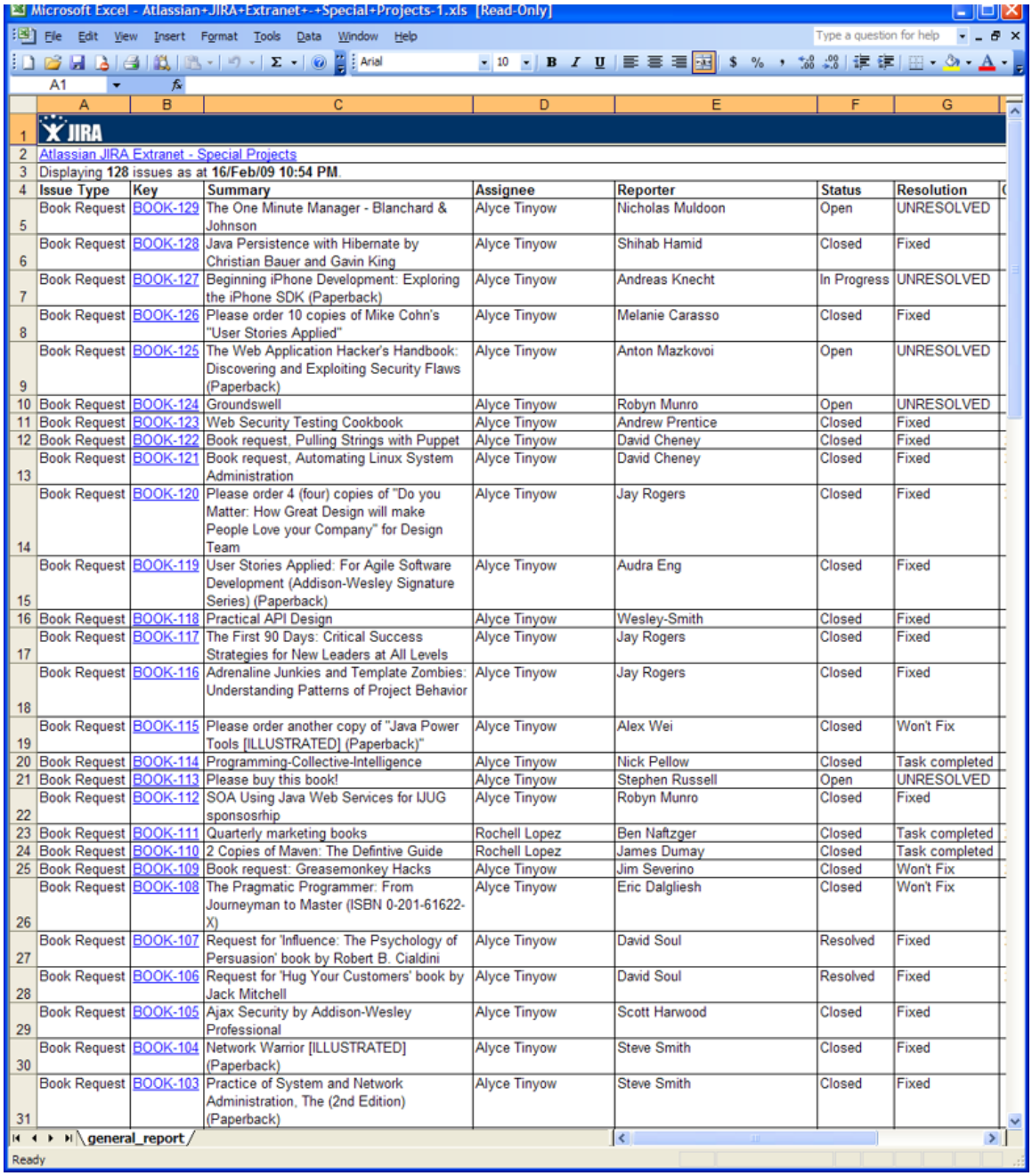
create your own customised reports, graphs and charts.

Exporting to Microsoft Excel

1. On the top navigation bar, click on the **'Find Issues'** tab.
2. Refine your search, as described in *'Searching for Issues',* until the required results are displayed in the Issue Navigator.
    - Note: large exports (e.g. many hundreds of issues) are not recommended.
    - Note: to change the number of issues that are exported, change the value of the **tempMax** parameter in the URL.
3. Click the **'Views'** menu, and select one of the following:
    - **'Excel (All fields)'** — this will create a spreadsheet column for every issue field (excluding comments).
    - **'Excel (Current fields)'** — this will create a spreadsheet column for the issue fields that are currently displayed in your Issue Navigator.
4. A file called **'<My company's JIRA> - <My project>.xls'** will be created. Edit this file using Microsoft Excel and/or save it as required.

Example

Here is a sample exported file, viewed in Microsoft Excel:



Microsoft Excel - Atlassian+JIRA+Extranet+-+Special+Projects-1.xls [Read-Only]

JIRA

Atlassian JIRA Extranet - Special Projects

Displaying **128** issues as at **16/Feb/09 10:54 PM**.

Issue Type	Key	Summary	Assignee	Reporter	Status	Resolution
Book Request	BOOK-129	The One Minute Manager - Blanchard & Johnson	Alyce Tinyow	Nicholas Muldoon	Open	UNRESOLVED
Book Request	BOOK-128	Java Persistence with Hibernate by Christian Bauer and Gavin King	Alyce Tinyow	Shihab Hamid	Closed	Fixed
Book Request	BOOK-127	Beginning iPhone Development: Exploring the iPhone SDK (Paperback)	Alyce Tinyow	Andreas Knecht	In Progress	UNRESOLVED
Book Request	BOOK-126	Please order 10 copies of Mike Cohn's "User Stories Applied"	Alyce Tinyow	Melanie Carasso	Closed	Fixed
Book Request	BOOK-125	The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws (Paperback)	Alyce Tinyow	Anton Mazkovoi	Open	UNRESOLVED
Book Request	BOOK-124	Groundswell	Alyce Tinyow	Robyn Munro	Open	UNRESOLVED
Book Request	BOOK-123	Web Security Testing Cookbook	Alyce Tinyow	Andrew Prentice	Closed	Fixed
Book Request	BOOK-122	Book request, Pulling Strings with Puppet	Alyce Tinyow	David Cheney	Closed	Fixed
Book Request	BOOK-121	Book request, Automating Linux System Administration	Alyce Tinyow	David Cheney	Closed	Fixed
Book Request	BOOK-120	Please order 4 (four) copies of "Do you Matter: How Great Design will make People Love your Company" for Design Team	Alyce Tinyow	Jay Rogers	Closed	Fixed
Book Request	BOOK-119	User Stories Applied: For Agile Software Development (Addison-Wesley Signature Series) (Paperback)	Alyce Tinyow	Audra Eng	Closed	Fixed
Book Request	BOOK-118	Practical API Design	Alyce Tinyow	Wesley-Smith	Closed	Fixed
Book Request	BOOK-117	The First 90 Days: Critical Success Strategies for New Leaders at All Levels	Alyce Tinyow	Jay Rogers	Closed	Fixed
Book Request	BOOK-116	Adrenaline Junkies and Template Zombies: Understanding Patterns of Project Behavior	Alyce Tinyow	Jay Rogers	Closed	Fixed
Book Request	BOOK-115	Please order another copy of "Java Power Tools [ILLUSTRATED] (Paperback)"	Alyce Tinyow	Alex Wei	Closed	Won't Fix
Book Request	BOOK-114	Programming-Collective-Intelligence	Alyce Tinyow	Nick Pellow	Closed	Task completed
Book Request	BOOK-113	Please buy this book!	Alyce Tinyow	Stephen Russell	Open	UNRESOLVED
Book Request	BOOK-112	SOA Using Java Web Services for IJUG sponsosrhip	Alyce Tinyow	Robyn Munro	Closed	Fixed
Book Request	BOOK-111	Quarterly marketing books	Rochell Lopez	Ben Naftzger	Closed	Task completed
Book Request	BOOK-110	2 Copies of Maven: The Defintive Guide	Rochell Lopez	James Dumay	Closed	Task completed
Book Request	BOOK-109	Book request: Greasemonkey Hacks	Alyce Tinyow	Jim Severino	Closed	Won't Fix
Book Request	BOOK-108	The Pragmatic Programmer: From Journeyman to Master (ISBN 0-201-61622-X)	Alyce Tinyow	Eric Dalgliesh	Closed	Won't Fix
Book Request	BOOK-107	Request for 'Influence: The Psychology of Persuasion' book by Robert B. Cialdini	Alyce Tinyow	David Soul	Resolved	Fixed
Book Request	BOOK-106	Request for 'Hug Your Customers' book by Jack Mitchell	Alyce Tinyow	David Soul	Resolved	Fixed
Book Request	BOOK-105	Ajax Security by Addison-Wesley Professional	Alyce Tinyow	Scott Harwood	Closed	Fixed
Book Request	BOOK-104	Network Warrior [ILLUSTRATED] (Paperback)	Alyce Tinyow	Steve Smith	Closed	Fixed
Book Request	BOOK-103	Practice of System and Network Administration, The (2nd Edition) (Paperback)	Alyce Tinyow	Steve Smith	Closed	Fixed

general_report

Ready

See also

Exporting Search Results to Microsoft Word

Displaying Search Results as a Chart

Overview

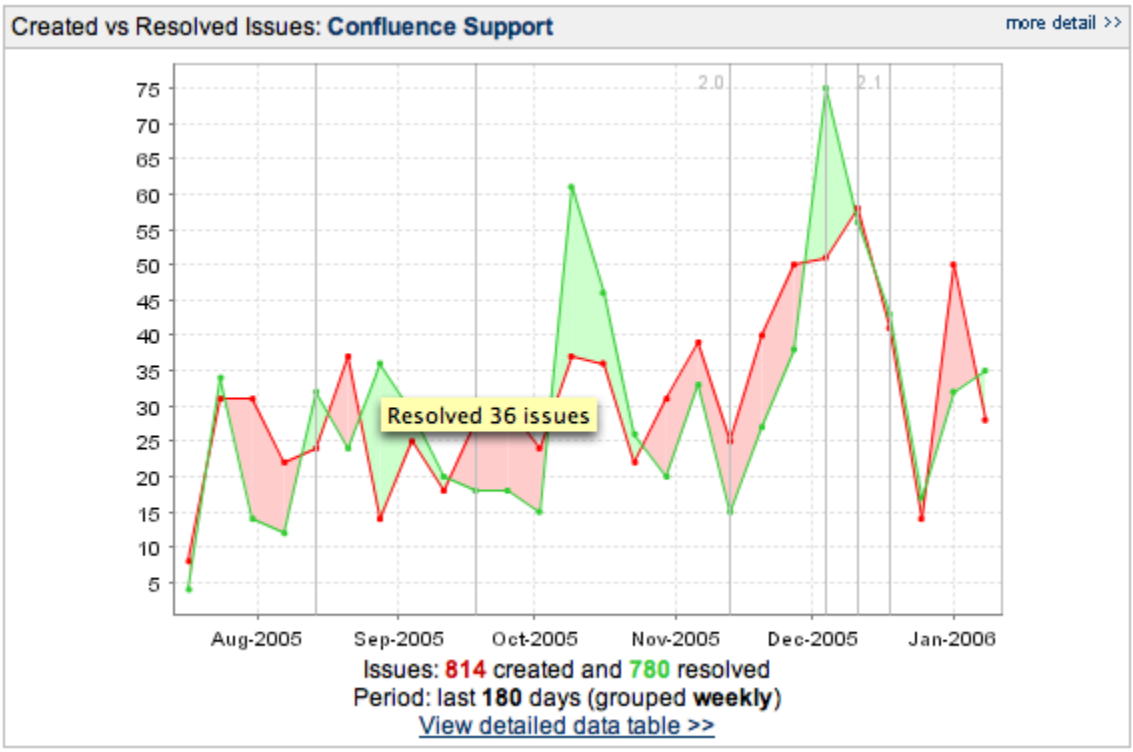
You can view any search results from the Issue Navigator in a variety of charting formats. You can also save them as a **Charting gadget** on your dashboard, as described below.

The **Charts** display visual representations of a filter in a variety of ways. In general, charts are:

- for a period of X days previous to the current date.
- broken down into incremental periods of hours, days, weeks, months, quarters or years.
- hyperlinked to relevant issues in the Issue Navigator.

What do they look like?

The **Created vs Resolved Issues** chart, for example, should appear as follows:



This is just one of the available charting gadgets, which include:

- **Created vs Resolved Issues** — a difference chart showing the issues created vs resolved over a given period.
  - This chart can either be cumulative or not.
  - Areas in red show periods where more issues were created than resolved, areas in green show periods where more were resolved than created.
  - Versions can also be added to this chart, showing you how issue creation and resolution related to version releases.
- **Resolution Time** — a bar chart showing the average resolution time (in days) of resolved issues.
  - This is useful to show you over time whether your team is getting better or worse at resolving issues in a timely fashion.
- **Pie Chart** — displays issues grouped by a statistic type in pie-chart format
  - The issues can be grouped by any statistic type (e.g. Status, Priority, etc.)
- **Average-Age Open Issues** — a bar chart showing the average number of days that issues have been unresolved
  - This chart displays the average of how long issues remain open on a specified interval (e.g. daily, weekly, etc.)
- **Recently Created Issues** — a bar chart showing the issues recently created.
  - The green portion of the bar shows the created issues which are resolved, the red portion shows created but as yet unresolved issues.
  - This visually shows both how quickly you're creating issues, as well as how many of those created issues are resolved.
- **Time Since Issues** — displays a bar chart showing the number of issues for which your chosen date field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a date custom field) was set on a given date.
- **Average Time in Status*** — displays the average number of days issues have spent in a status.
- **Average Number of Times in Status*** — displays the average number of times an issues has been in a status.

- **Time to First Response*** — displays the average number of times an issues has been in a status.

> ***This particular chart will only be available if your JIRA administrator has installed the Charting plugin.**.

Viewing a Chart

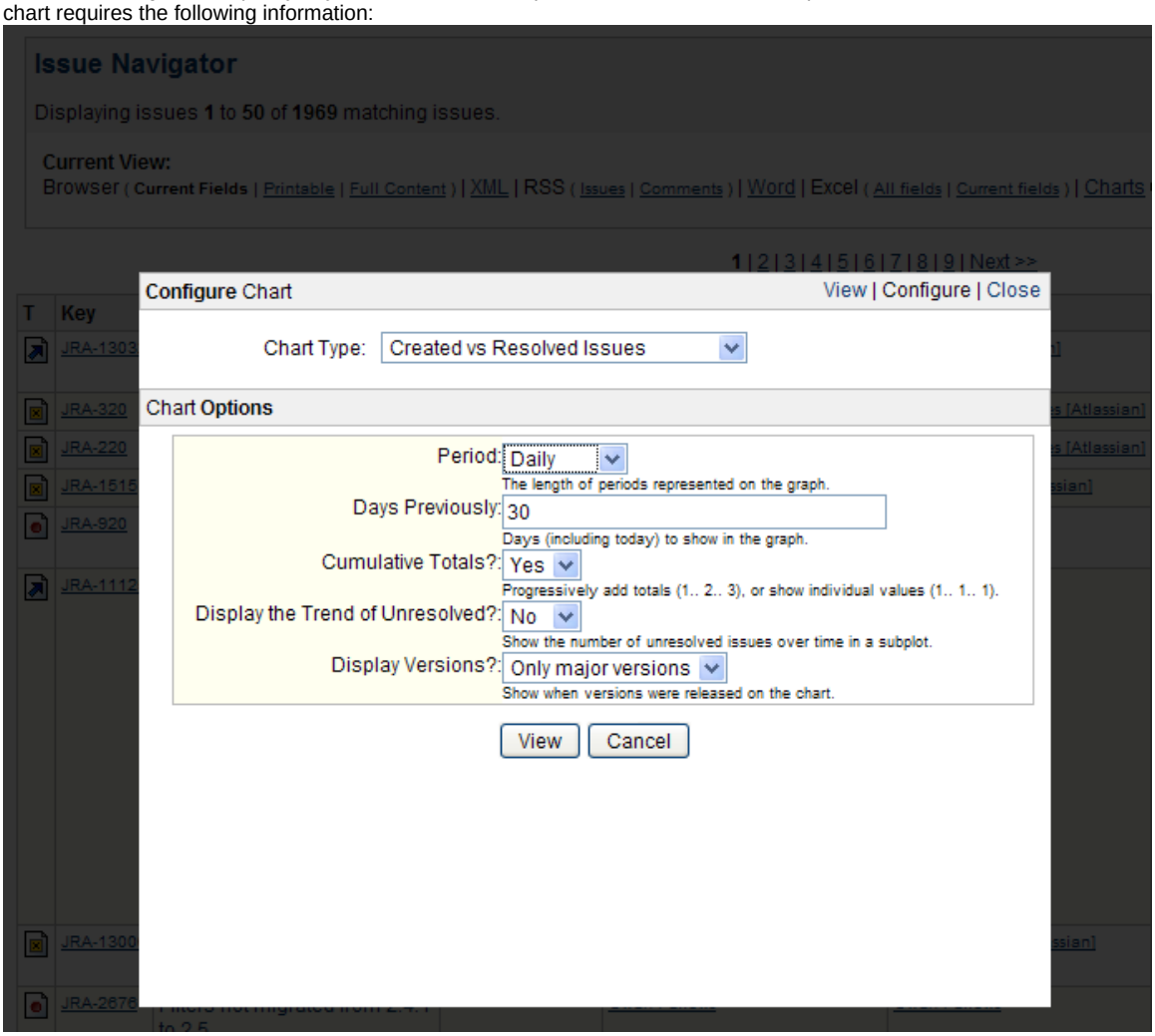
To view your search results as a chart,

1. On the top navigation bar, click on the **'Find Issues'** tab.
2. Refine your search, as described in *'Searching for Issues',* until the required results are displayed in the Issue Navigator.
3. Click the **'Views'** menu, and select **'Charts'.**
4. Your search results will be displayed as a chart. If you wish to select a different type of chart,
    - Click **'Configure'.**
    - Select the appropriate **'Chart Type'** as listed above, and enter any additional parameters (if required for your chosen type of chart).
    - Click **'View'.**

Adding Gadgets to your Dashboard

To create a chart based on your search results and display it on your dashboard (note that this process will also create a saved filter):

1. View the desired search results in your Issue Navigator.
2. Click the **'Views'** menu, and select **'Charts'.**
3. The default chart will be displayed.
4. If you wish to choose a different type of chart,
    a. Click **'Configure'.** The **'Configure Chart'** form will be displayed.
    b. In the **'Chart Type'** field, select one of the other Charting portlets as listed above.
    c. Under **'Chart Options',** specify any other information required for the chart. For example, the **'Created vs Resolved Issues'** chart requires the following information:



    d. Click **'View'** to preview your chart. The **'Save'** button will reappear.

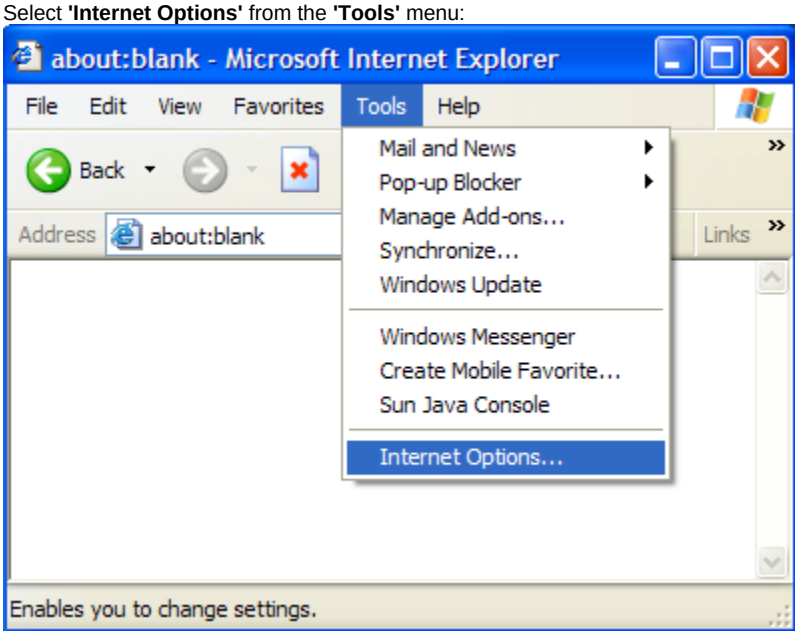
5. Type a name for your search results in the **'Filter Name'** field, then click the **'Save'** button.
6. The chart will now appear as a gadget on your dashboard.
7. If you wish, the gadget can be repositioned on the dashboard through the dashboard configuration page.

Further information on all JIRA dashboard gadgets is available here.

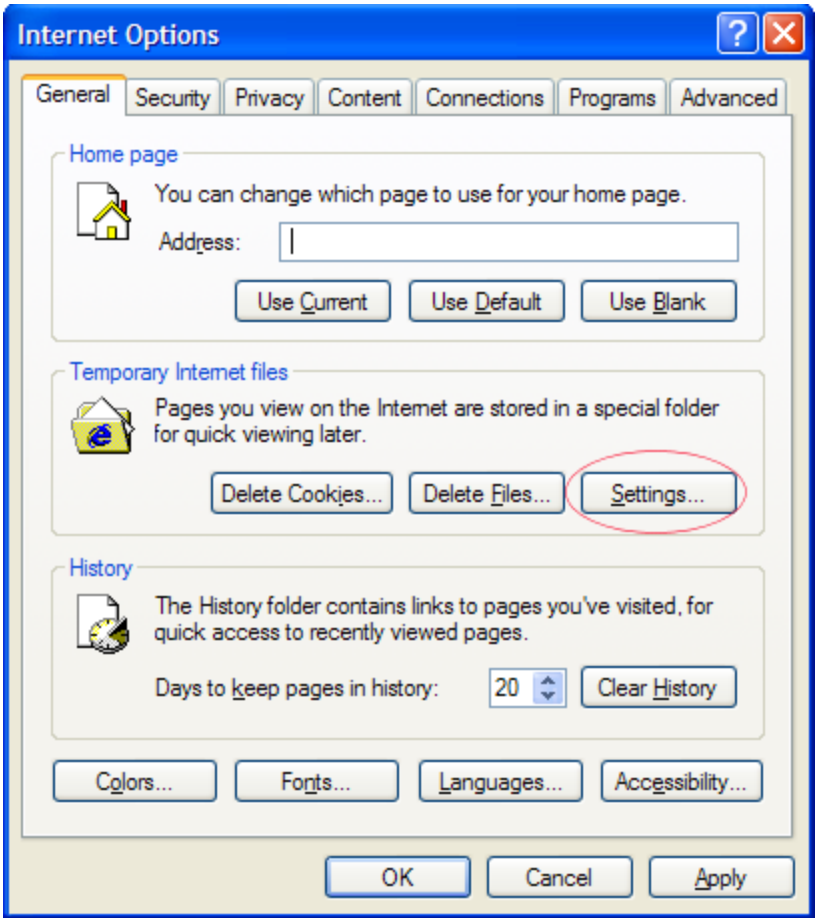
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

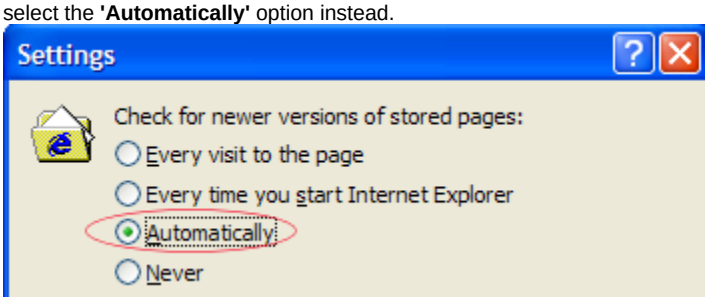
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Generating Reports

JIRA provides reports that show statistics for particular people, projects, versions, or other fields within issues. The following reports are included with JIRA:

- User Workload Report * — Shows how much work a user has been allocated, and how long it should take.
- Version Workload Report * — Shows how much outstanding work there is (per user and per issue) before a given version is complete.
- Version Time Tracking Report * — Shows progress towards completing a given version, based on issues' work logs and time estimates.
- Single Level Group By Report — Shows the search results from an issue filter, grouped by a field of your choice.
- Created vs Resolved Issues Report — Shows the number of issues created vs number of issues resolved over a given period of time.
- Resolution Time Report — Shows the average time taken to resolve issues.
- Pie Chart Report — Shows the search results from a specified issue filter (or project) in a pie-chart, based on a statistic of your choice.
- Average Age Report — Shows the average age (in days) of unresolved issues.
- Recently Created Issues Report — Shows the rate at which issues are being created.
- Time Since Issues Report — Shows the number of issues for which your chosen date field (e.g. 'Created') was set on a given date.

** Only available if your JIRA administrator has enabled Time Tracking.*

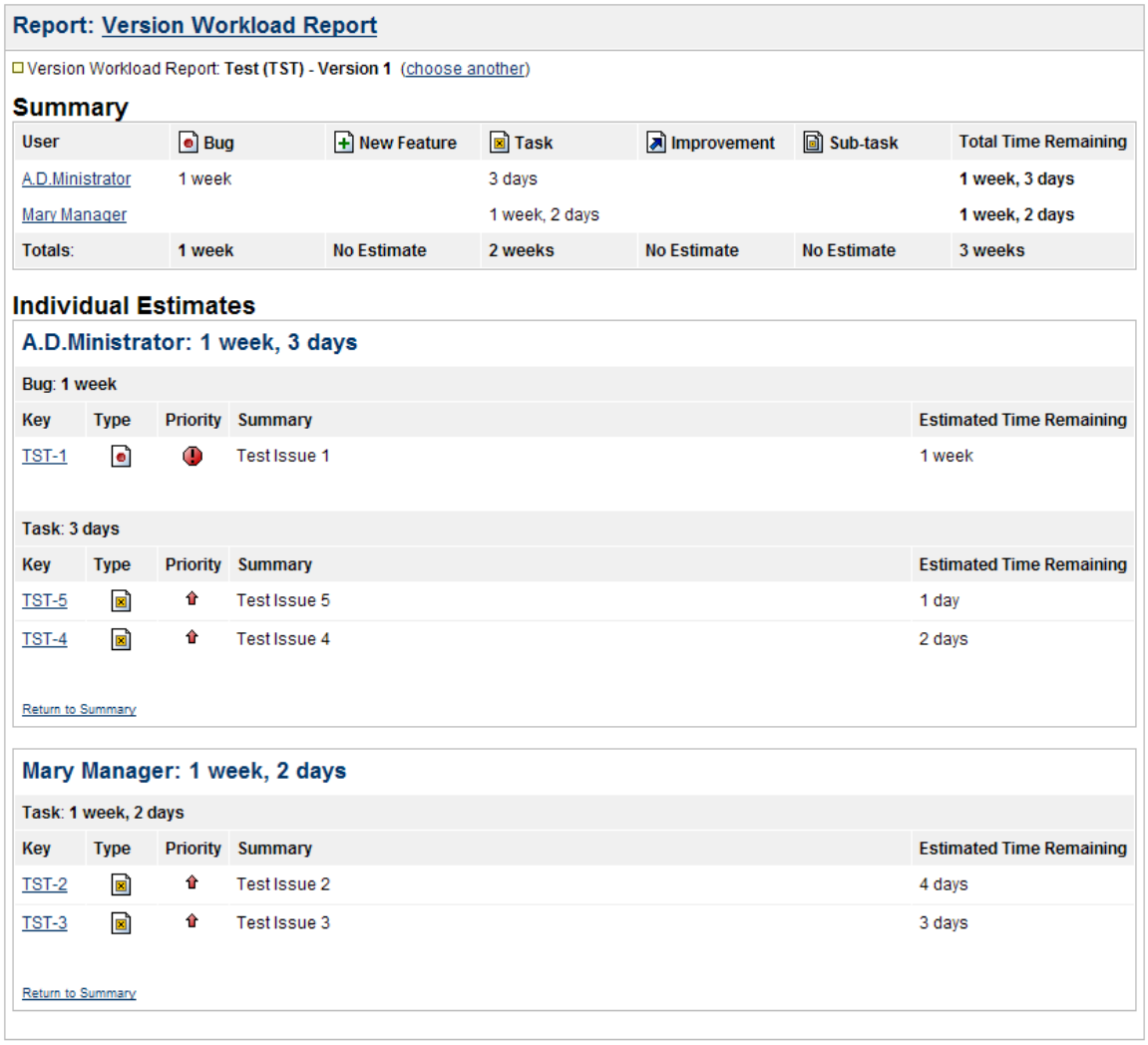
In addition to the built-in reports, other reports (e.g. Gantt Chart Report, Timesheet Report/Portlet) are available for download from the Atlassian Plugin Exchange and the JIRA Extensions site. JIRA administrators can also create new reports with the plugin API.

Version Workload Report

The Version Workload Report displays useful time tracking information on the current workload for a specific version within a specific project. For the specified version, it shows a list of unresolved issues assigned to each user, each user's workload, and a summary of the total workload remaining for the version. (Note: this report is only available if time tracking has been enabled by your JIRA administrator).

What does the 'Version Workload' report look like?

The report generated will look something like this:



Report: Version Workload Report

Version Workload Report: **Test (TST) - Version 1** (choose another)

Summary

User	Bug	New Feature	Task	Improvement	Sub-task	Total Time Remaining
A.D.Ministrator	1 week		3 days			**1 week, 3 days**
Mary Manager			1 week, 2 days			**1 week, 2 days**
Totals:	**1 week**	**No Estimate**	**2 weeks**	**No Estimate**	**No Estimate**	**3 weeks**

Individual Estimates

A.D.Ministrator: 1 week, 3 days

Bug: 1 week

Key	Type	Priority	Summary	Estimated Time Remaining
TST-1			Test Issue 1	1 week

Task: 3 days

Key	Type	Priority	Summary	Estimated Time Remaining
TST-5			Test Issue 5	1 day
TST-4			Test Issue 4	2 days

Return to Summary

Mary Manager: 1 week, 2 days

Task: 1 week, 2 days

Key	Type	Priority	Summary	Estimated Time Remaining
TST-2			Test Issue 2	4 days
TST-3			Test Issue 3	3 days

Return to Summary

The first table summarises the workload for each user, broken down by issue type, for the specified version.

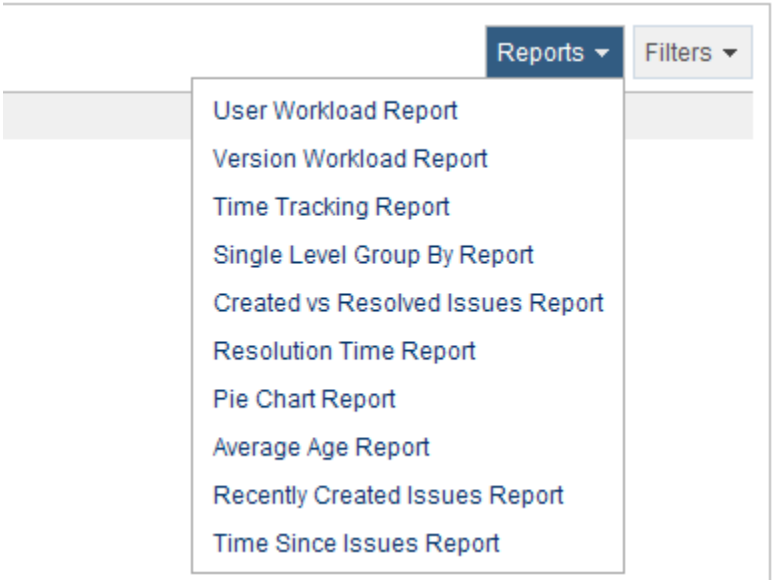
Following the summary, the report is composed of individual sections for each user --- with workload broken down by issue type. Each individual section begins with the workload total for the specific user. Finally, all unassigned issues (if any exist) are displayed.

Generating a Version Workload Report

To generate a version workload report:

1. Click **'Browse Project'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.

3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Version Workload Report'.**
5. The following form will appear:
6. In the **'Display unestimated issues'** drop-down list, choose which issues will be included in the report:
    - Select **'Yes'** to show all unresolved issues, regardless of the value of their **'Estimated Time Remaining'** or **'Original Estimate'** fields.
    - Select **'No'** to exclude issues which are not time-tracked (i.e. do not have an **'Original Estimate'** specified).
7. In the **'Version'** drop-down list, select the version on which you wish to report. The report will include all issues that belong to this version, that is, all issues whose **'Fix Version'** is this version.
8. In the **'Sub-task Inclusion'** drop-down list (note: this will only appear if sub-tasks are enabled), choose which sub-tasks will be included in the report, for all parent issues that belong to this version:
    - Select **'Only include sub-tasks with the selected version'** to only include an issue's sub-tasks if the sub-tasks belong to the same version as the issue; or
    - Select **'Also include sub-tasks without a version set'** to include an issue's sub-tasks if the sub-tasks belong to either the same version as the issue or to no version; or
    - Select **'Include all sub-tasks'** to include all of an issue's sub-tasks, regardless of whether the sub-tasks belong to the same version, some other version or no version.

Note: sub-tasks which belong to this version, but whose parent issues donotbelong to this version, will always be included in the report.

Time Tracking Report

The Time Tracking Report displays useful time tracking information on issues for a particular version of a project. This report shows original and current time estimates for all the issues, and whether they are ahead of or behind the original schedule. (Note: this report is only available if time tracking has been enabled by your JIRA administrator).

What does the 'Time Tracking' report look like?

The report generated will look something like this:

Report: Time Tracking Report

Description:
This report shows the time tracking details for a specific project.

Excel View

Time Tracking Report for Test (Version 1)

Progress: 47%

Accuracy: 8%

2w 1d completed from current total estimate of **4w 3d**

Issues in this version are **ahead** of the original estimate of **5w** by **2 days**.

Key	Summary	Original Estimate	Est. Time Remaining	Time Spent	Accuracy
TST-1	Test Issue 1	1w	1w	-	on track
TST-3	Test Issue 3	1w	3d	2d	on track
TST-4	Test Issue 4	1w	2d	3d	on track
TST-5	Test Issue 5	1w	1d	1w	-1d
TST-2	Test Issue 2	1w	1d	1d	3d
Total		**5w**	**2w 2d**	**2w 1d**	**2d**

Or, if sub-tasks are enabled, the report will look something like this:

Report: Time Tracking Report

Description:
This report shows the time tracking details for a specific project.

Excel View

Time Tracking Report for Dove

Only including sub-tasks with the selected version

Progress: 54%

Accuracy: 0%

2w 2d completed from current total estimate of **4w 2d**

Issues in this version are on schedule with original estimate of 4w 2d.

Key	Summary	Original Estimate	Σ	Est. Time Remaining	Σ	Time Spent	Σ	Accuracy	Σ
DOVE-1	Win 'homing dove' contest	-	4w 2d	-	2w	-	2w 2d	-	on track
DOVE-6	Spend prize money	-	-	-	-	-	-	-	-
DOVE-5	Win	-	-	-	-	-	-	-	-
DOVE-3	Teach dove to fly	4w	-	2w	-	2w	-	on track	-
DOVE-4	Enter competition	1d	-	0m	-	1d	-	on track	-
DOVE-2	Buy a dove	1d	-	0m	-	1d	-	on track	-
Total		**4w 2d**		**2w**		**2w 2d**		**on track**	

The table in the report shows the issues within the version:

- There are four time tracking fields as follows:
  - **Original Estimate** - The original estimate of the total amount of time it would take to complete this issue.
  - **Estimated Time Remaining** - The current estimate of the remaining amount of time it would take to complete this issue.
  - **Time Spent** - The amount of time spent on the issue. This is the aggregate amount of time which has been logged against this issue.
  - **Accuracy**- The accuracy of the original estimate compared to the current estimate for the issue. It is the difference between the sum of the **Time Spent** and **Estimated Time Remaining** fields, and the **Original Estimate** field.
- If sub-tasks are enabled, the *'*column at the right of the field shows the aggregate time tracking information for each 'parent' issue (i.e. the sum of the issue's own values plus those of its sub-tasks).
- The last line of the table shows the aggregate time tracking information for the whole version.

The report also includes two bar-graphs (above the table) which represent the aggregate time tracking information for the version:

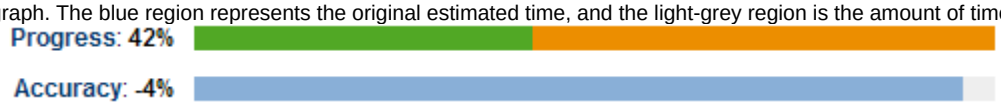
- The first bar-graph **('Progress')** shows the percentage of completed issues (green) and incomplete issues (orange) in this version:

- The second bar-graph **('Accuracy'** -blue) shows the accuracy of the original estimates.
  The length of the **Accuracy** bar compared to the **Progress** bar indicates whether the issues in this version are ahead of or behind schedule. There are three cases:

1. *The issues are on schedule with the original estimate.* The **Accuracy** bar is completely blue and is the same length as the **Progress** bar above it.

2. *The issues are behind the original estimate (i.e. will take longer than originally estimated).* The **Progress** graph is longer than the

Accuracygraph. The blue region represents the original estimated time, and the light-grey region is the amount of time by which issues are behind.

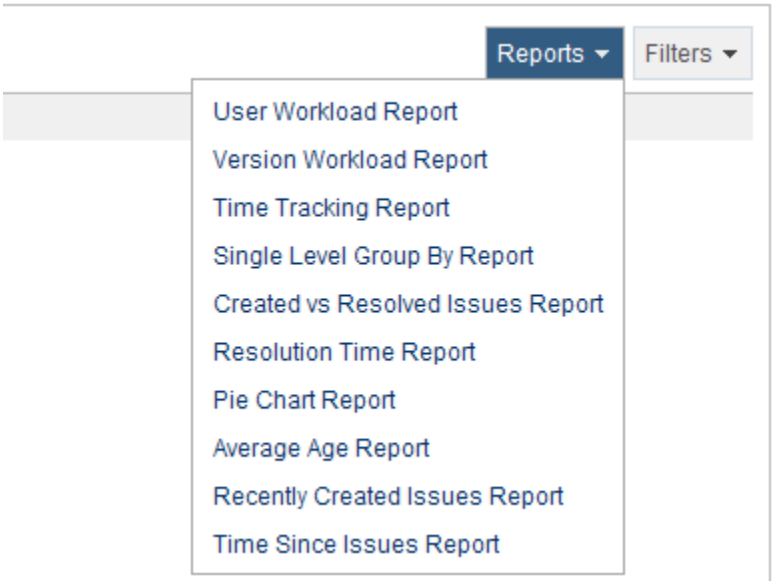


3. *The issues are ahead of the original estimate (i.e. will take less time than originally estimated).* The **Accuracy** graph is longer than the **Progress**graph. The blue bar represents the original estimated time, and the light-grey region represents the amount of time by which the original estimates were overestimated.

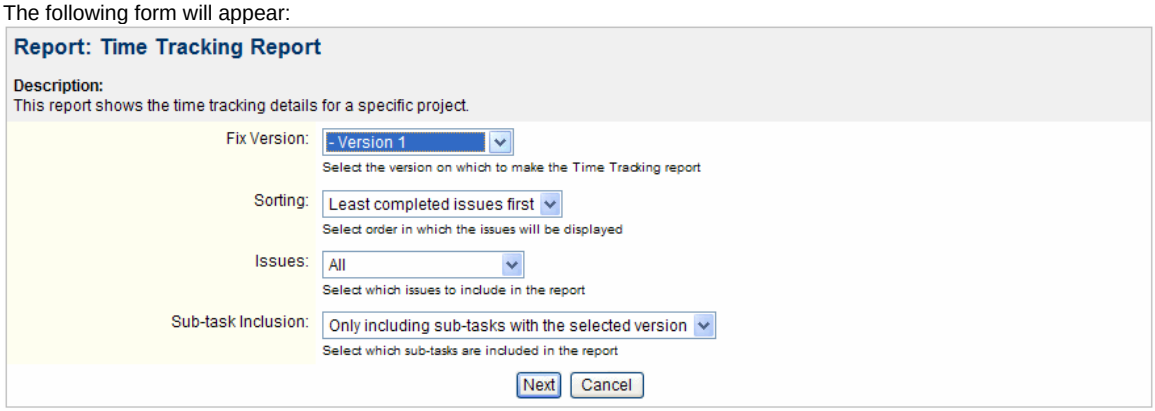
Generating a 'Time Tracking' report

To generate a Time Tracking Report:

1. Click **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Time Tracking Report'.**
5. The following form will appear:



6. In the **'Version'** drop-down list, select the version on which you wish to report. The report will include all issues that belong to this version, that is, all issues whose **'Fix Version'** is this version.
7. In the **'Sorting'** drop-down list, choose how the issues in the report will be sorted:
   - Select **'Least completed issues first'** to show issues with the highest **'Estimated Time Remaining** first; or
   - Select **'Most completed issues first'** to show issues with the lowest **'Estimated Time Remaining** first.
8. In the **'Issues'** drop-down list, choose which issues will be included in the report:
   - Select **'All'** to include all issues assigned to this version; or
   - Select **'Incomplete issues only'** to exclude issues which are either completed (i.e. have an **'Estimated Time Remaining'** of zero), or are not time-tracked (i.e. do not have an **'Original Estimate').** Note that issue status does not affect which issues are displayed.
9. In the **'Sub-task Inclusion'** drop-down list *(note: this will only appear if sub-tasks are enabled),* choose which sub-tasks will be included in the report, for all parent issues that belong to this version:
   - Select **'Only include sub-tasks with the selected version'** to only include an issue's sub-tasks if the sub-tasks belong to the

same version as the issue; or
- Select **'Also include sub-tasks without a version set'** to include an issue's sub-tasks if the sub-tasks belong to either the same version as the issue or to no version; or
- Select **'Include all sub-tasks'** to include all of an issue's sub-tasks, regardless of whether the sub-tasks belong to the same version, some other version or no version.Note: sub-tasks which belong to this version, but whose parent issues do *not* belong to this version, will always be included in the report.

User Workload Report

The 'User Workload' report displays useful time tracking information on issues assigned to a particular user. It shows the number of unresolved issues assigned to the specified user, and the workload remaining, on a per-project basis. (Note: this report is only available if time tracking has been enabled by your JIRA administrator).

What does the 'User Workload' report look like?

The report generated will look something like this:

Report: User Workload Report

Workload report for user A.D.Ministrator (choose another)

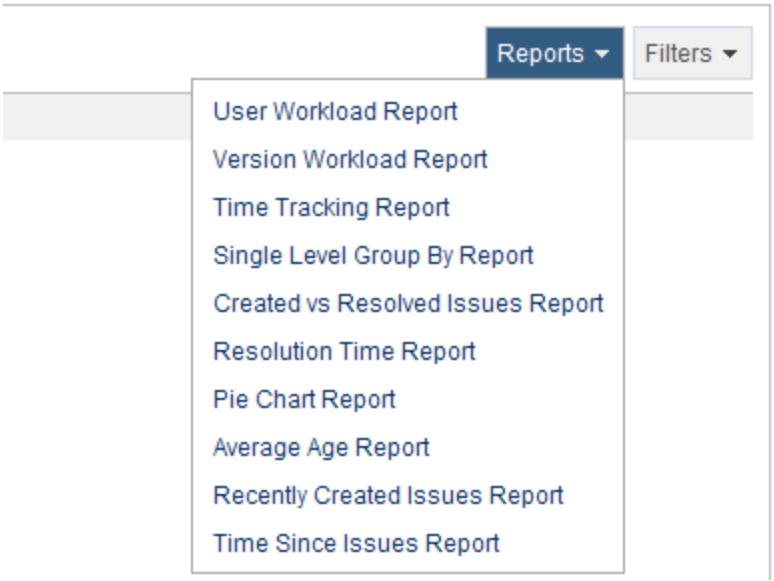
Projects	Assigned Issues	Workload
Test	3	1 week, 3 days
Dove	1	2 weeks
Total	**4**	**3 weeks, 3 days**

The table shows the number of unresolved issues assigned to the specified user, and the workload remaining, on a per-project basis. The last line in the table shows the total the number of issues and the total workload remaining for this user.

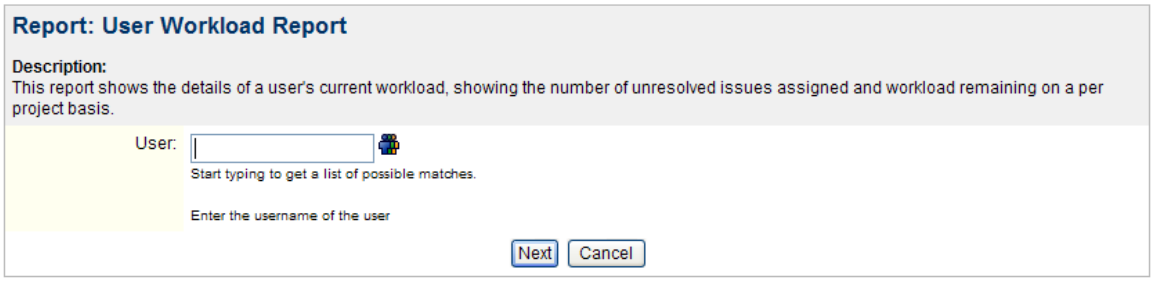
Generating a 'User Workload' report

To generate a user workload report:

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list. (The choice of project will not affect the report.)
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'User Workload Report'.**
5. The following form will appear:



6. Select or type the name of the user for whom you wish to generate a User Workload report.

Single Level Group By Report

The 'Single Level Group By' report displays issues returned from a specified issue filter, grouped by a specified field. For example, an issue filter can be created to retrieve all open issues for a particular version of a particular project. The 'Single Level Group By' report can then be used to display these issues grouped by a specified field (e.g. Assignee).

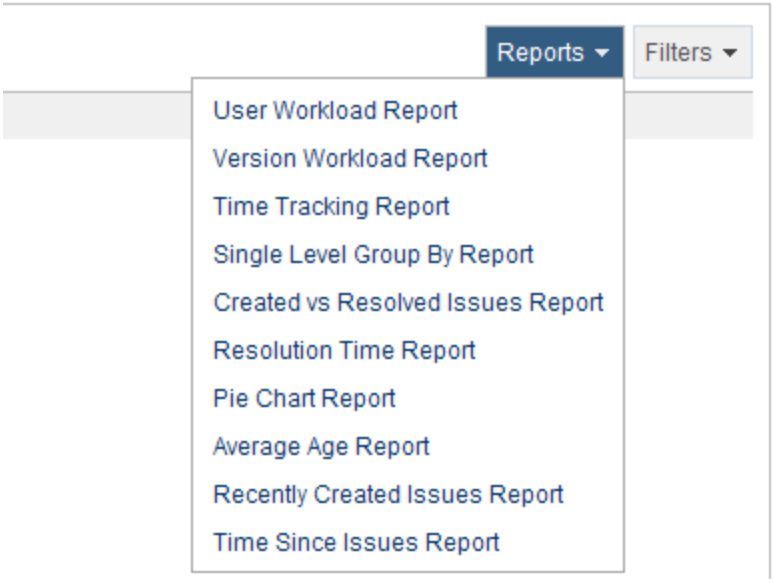
What does the 'Single Level Group By' report look like?

The report generated will look something like this:

The report displays the issues returned by the specified filter, grouped by the specified field.

Generating a 'Single Level Group By' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list. (The choice of project will not affect the report.)
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Single Level Group By Report'.**
5. The following form will appear:

6. Click **'Select Filter...'.**
7. The popup **Filter Picker** will appear. Select the issue filter. You will then be returned to the form.
8. In the **'Statistic Type'** field, select the field by which the report will group the issues returned from your chosen issue filter.

Created vs Resolved Issues Report

The 'Created vs Resolved Issues' report is a difference chart showing the number of issues created vs number of issues resolved over a given period of time. The report is based on your choice of project or issue filter, and the chart can either be cumulative or not.

What does the 'Created vs Resolved Issues' report look like?

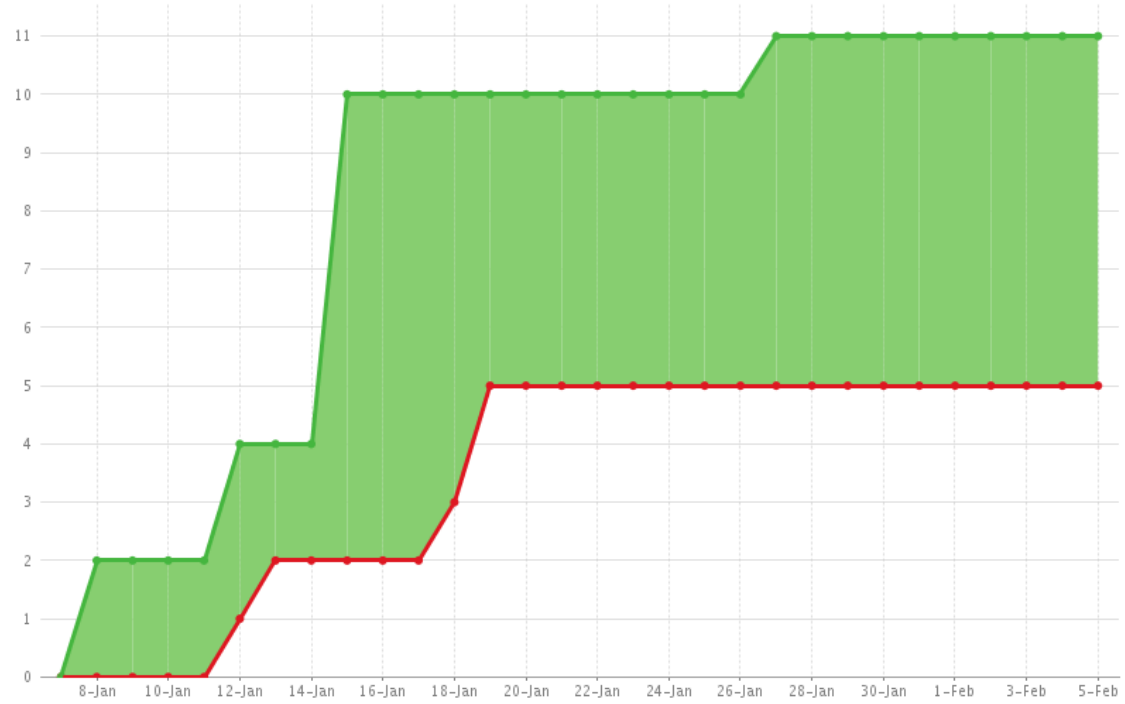
The report generated will look something like this:

Report: Created vs Resolved Issues Report

Project: Book Request

Chart

This chart shows the number of issues created vs the number of issues resolved in the last 30 days.



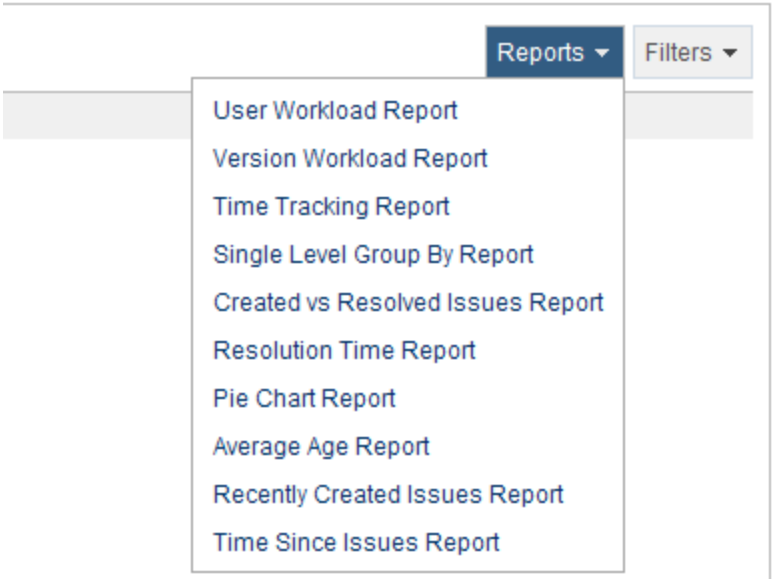
Data Table

Period	Created	Resolved
7-January-2009	0	0
8-January-2009	0	2
9-January-2009	0	0
10-January-2009	0	0
11-January-2009	0	0
12-January-2009	1	2
13-January-2009	1	0
14-January-2009	0	0
15-January-2009	0	6

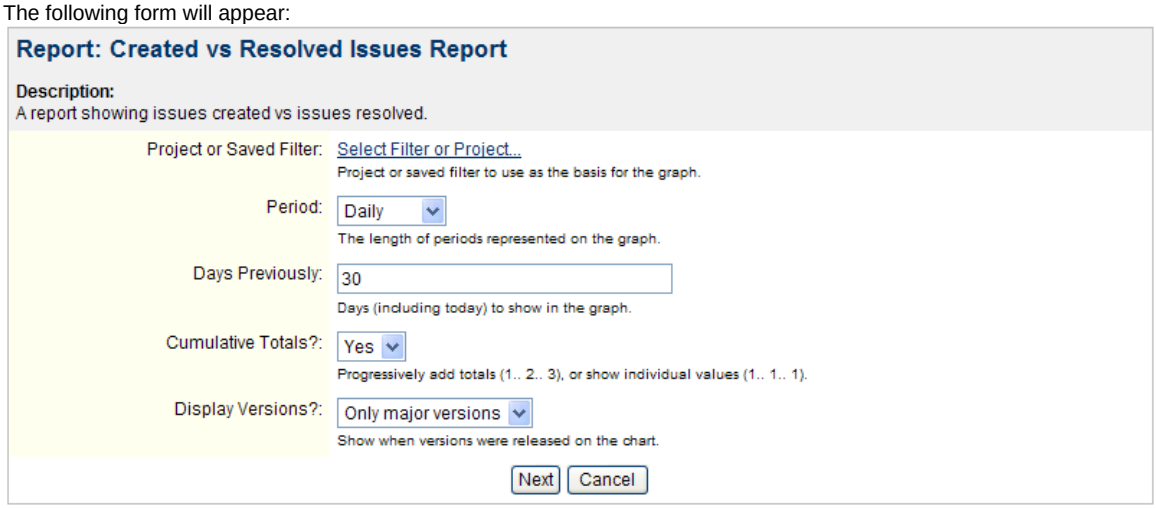
Areas in red show periods where more issues were created than resolved. Areas in green show periods where more were resolved than created.

Generating a 'Created vs Resolved Issues' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Created vs Resolved Issues Report'.**
5. The following form will appear:

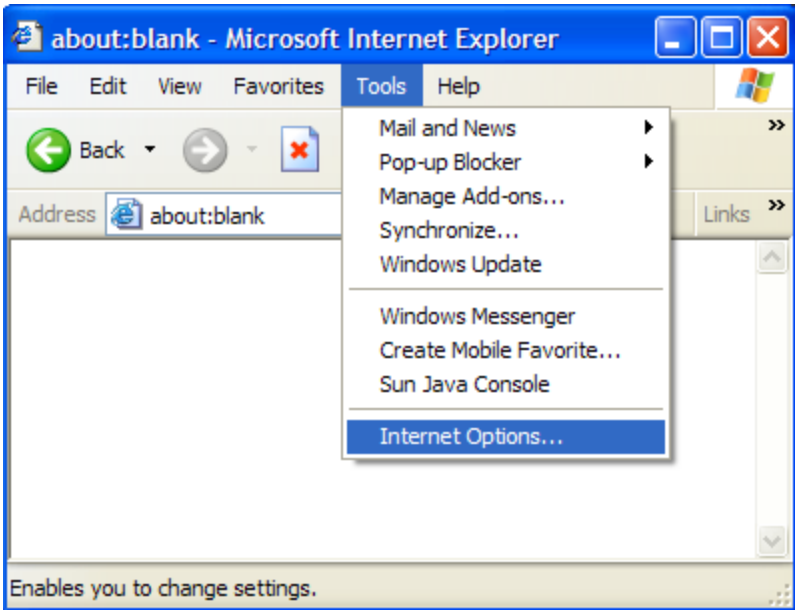


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Period'** field, select the timeframe on which the report will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
9. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.
10. In the **'Cumulative Totals?'** field, choose either:
    - **'Yes'** to progressively add data to the preceding column; or
    - **'No'** to show just a single value in each column.
11. In the **'Display Versions?'** field, choose either:
    - **'All versions'** to show version release dates on the chart, for all released versions; or
    - **'Only major versions'** to show version release dates on the chart, for released versions that are named 'x.x' only; or
    - **'None'** to not show version release dates on the chart.

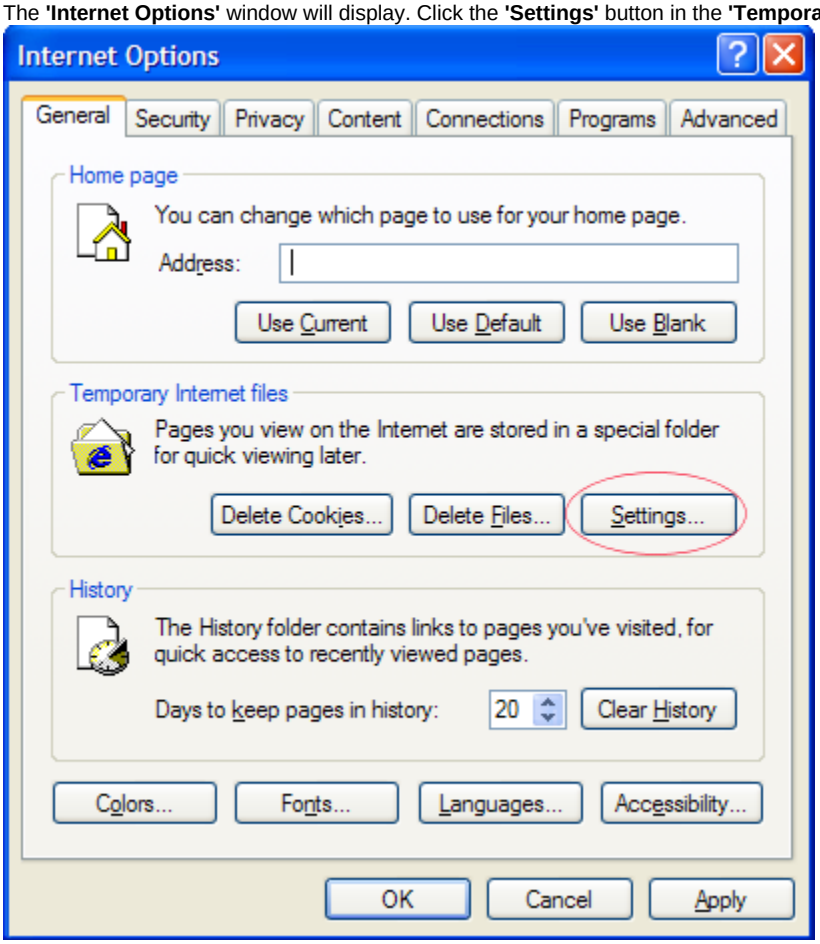
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

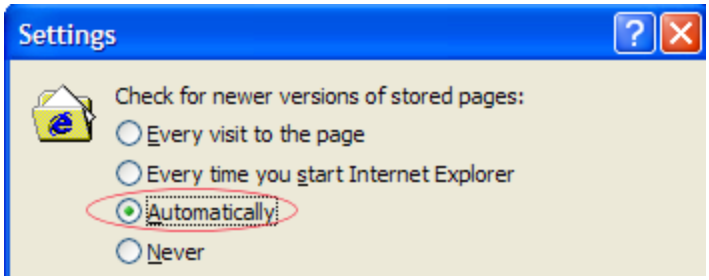
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Resolution Time Report

The 'Resolution Time' report is a bar chart showing the average time taken to resolve issues. This is useful to show you the trends in resolution time. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does the 'Resolution Time' report look like?

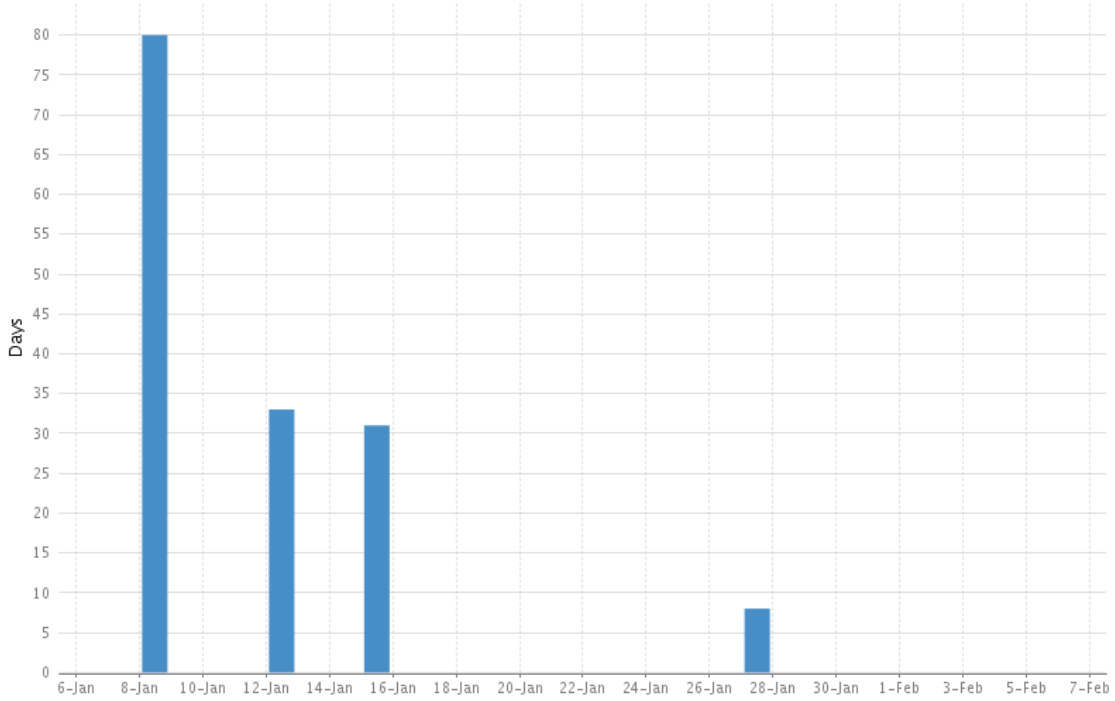
The report generated will look something like this:

Report: Resolution Time Report

Project: Book Request

Chart:

This chart shows the average number of days issues took to be resolved.
Issues were resolved in the last **30** days.

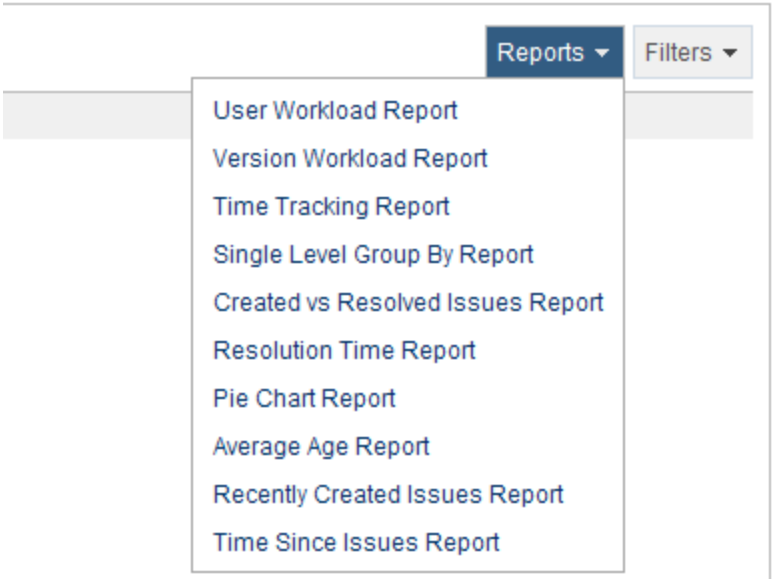


Data Table

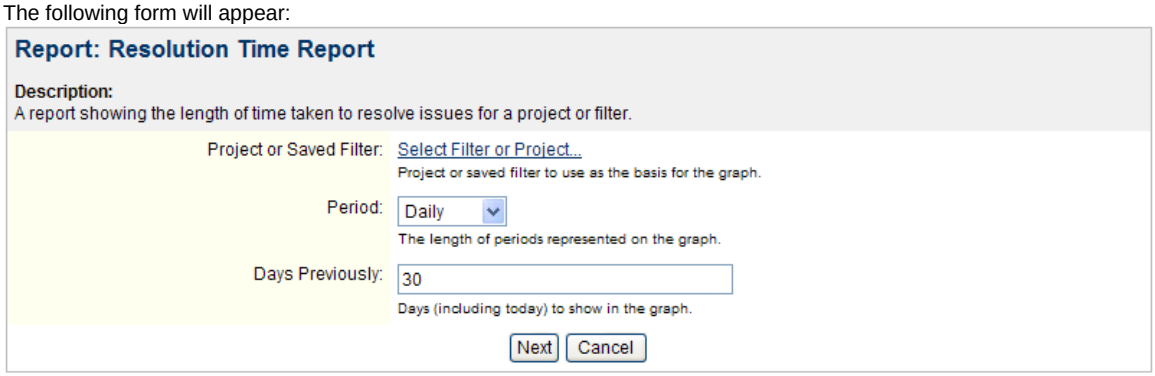
Period	Issues Resolved	Total Resolution Time	Avg. Resolution Time
7-January-2009	0	0	0
8-January-2009	2	160	80
9-January-2009	0	0	0
10-January-2009	0	0	0
11-January-2009	0	0	0
12-January-2009	2	66	33
13-January-2009	0	0	0

Generating a 'Resolution Time' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Resolution Time Report'.**
5. The following form will appear:

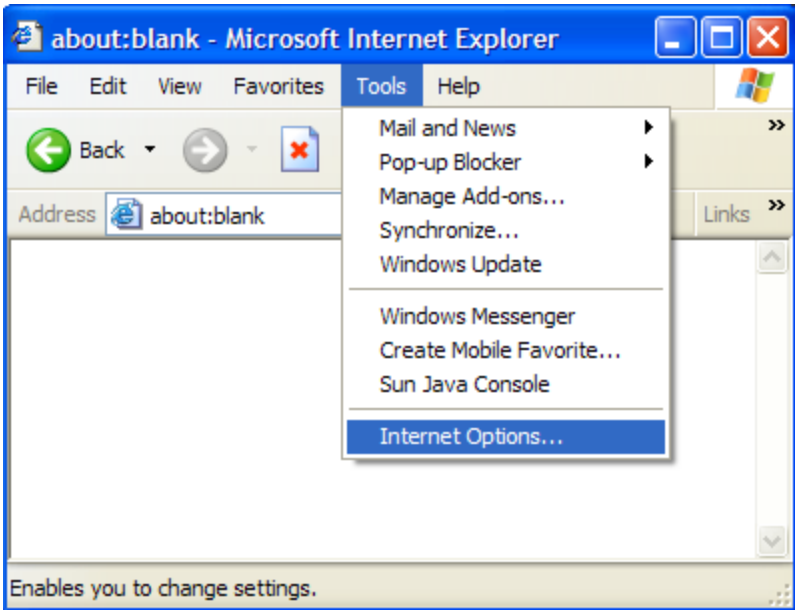


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Period'** field, select the timeframe on which the report will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
9. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.

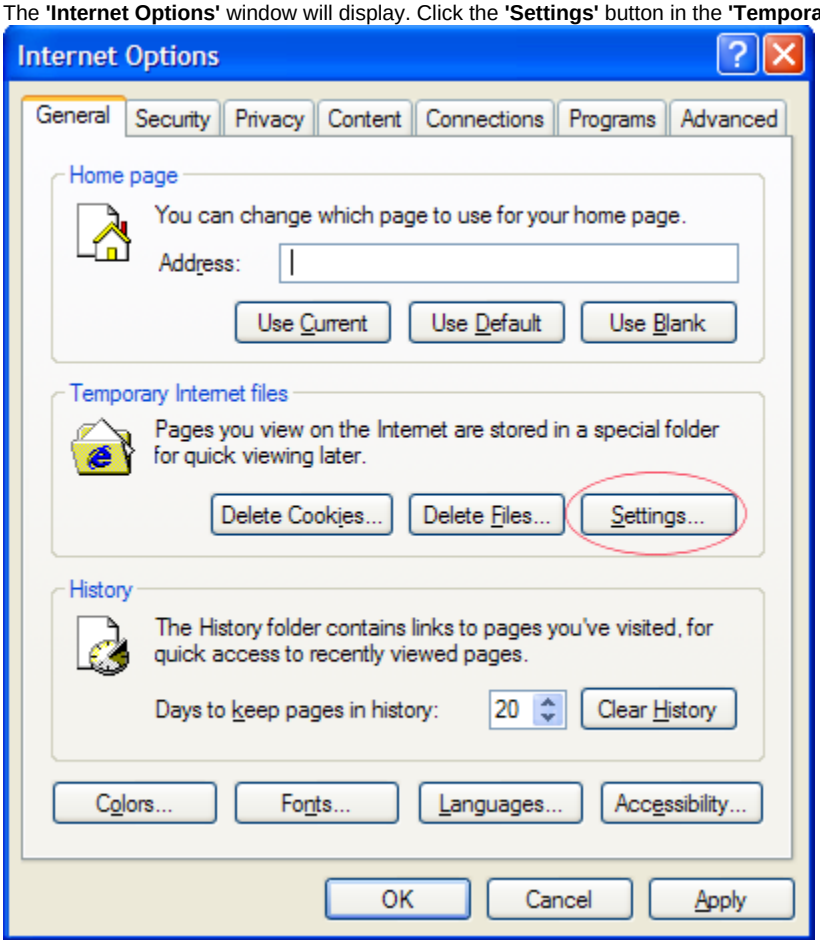
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

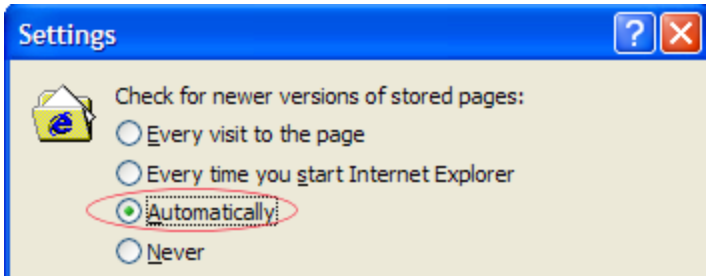
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Pie Chart Report

The 'Pie Chart' report displays issues returned from a specified project or issue filter, grouped by a specified field. For example, an issue filter can be created to retrieve all open issues for a particular version of a particular project. The 'Pie Chart' report can then be used to display these issues grouped by a specified field (e.g. Assignee).

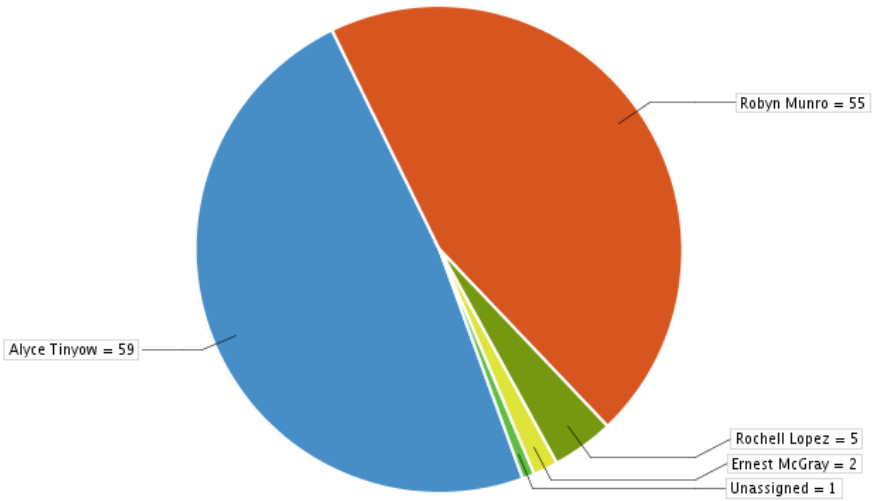
What does the 'Pie Chart' report look like?

The report generated will look something like this:

Report: Pie Chart Report

Project: Book Request (Assignee)

Chart

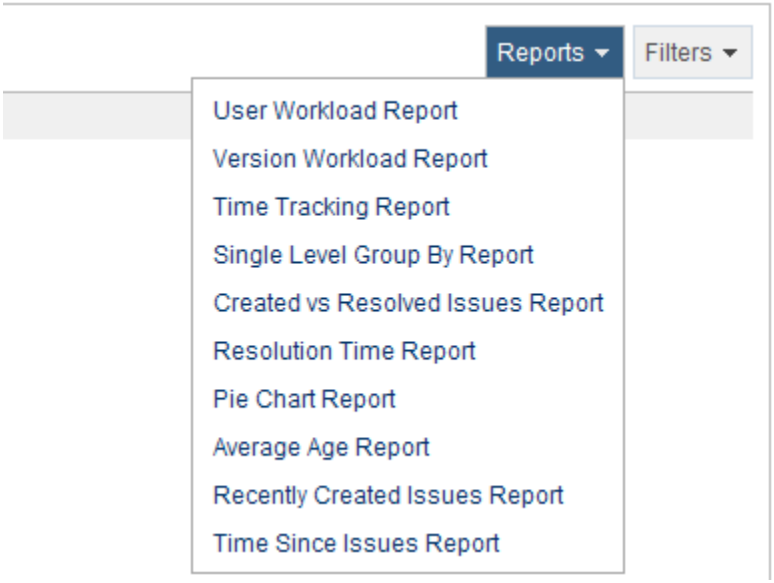


Data Table

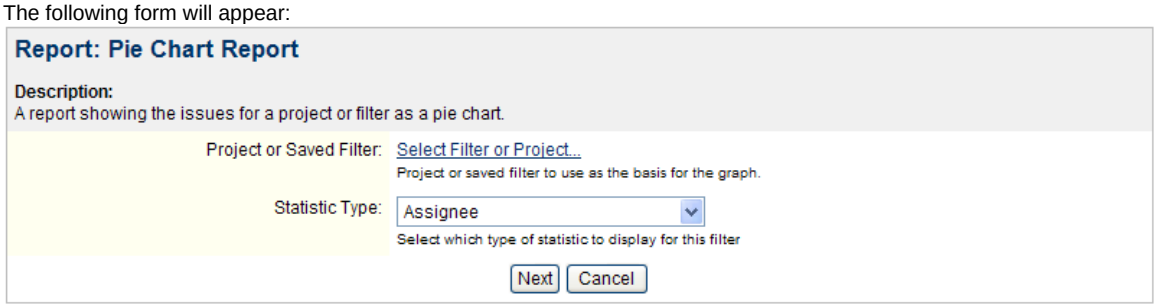
	Issues	%
Alyce Tinyow	59	48%
Robyn Munro	55	45%
Rochell Lopez	5	4%
Ernest McGray	2	1%
Unassigned	1	0%

Generating a 'Pie Chart' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Pie Chart Report'.**
5. The following form will appear:

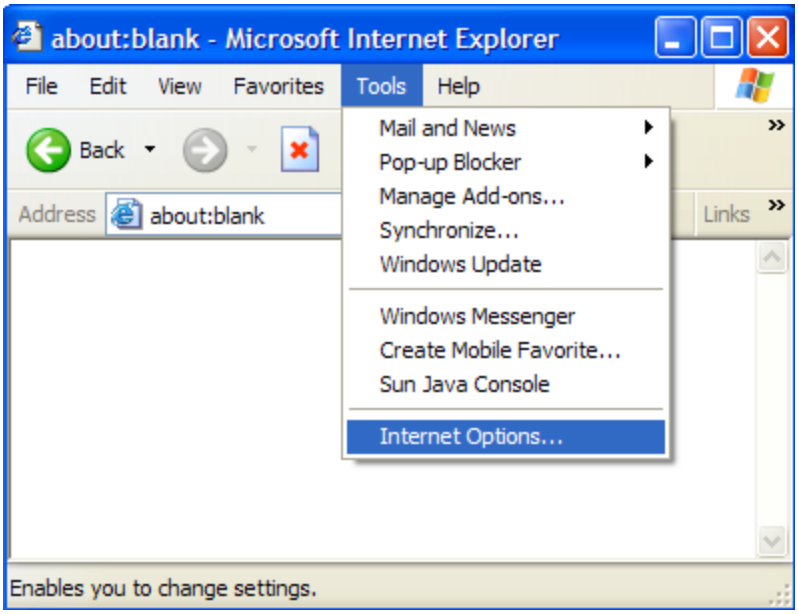


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Statistic Type'** field, select the field on which the pie chart will be based. (Note that you can choose only fields which have finite values).

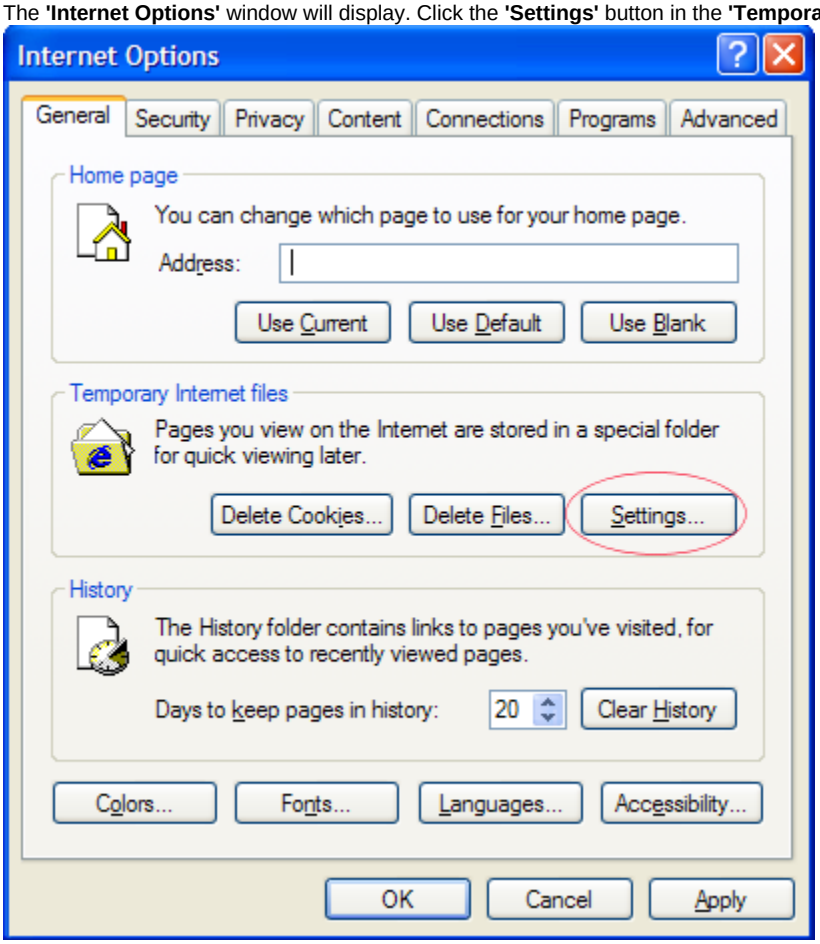
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

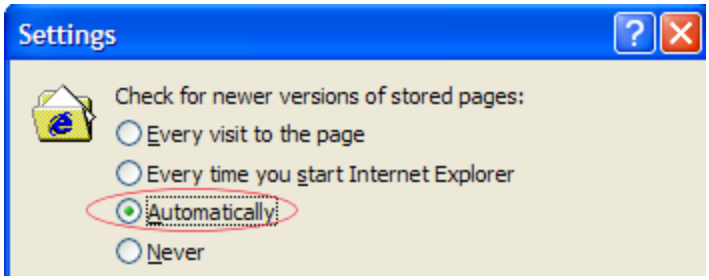
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Average Age Report

The 'Average Age' report is a bar chart showing the average age (in days) of unresolved issues at given points in time. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does the 'Average Age' report look like?

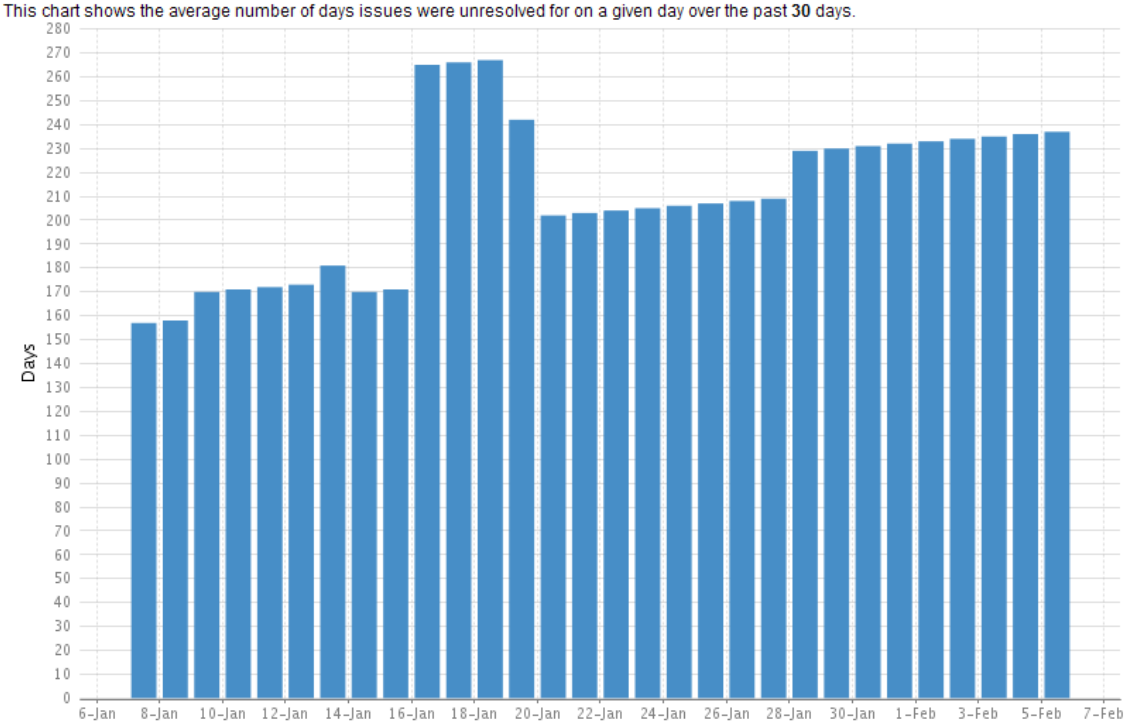
The report generated will look something like this:

Report: Average Age Report

Project: Book Request

Chart:

This chart shows the average number of days issues were unresolved for on a given day over the past **30** days.

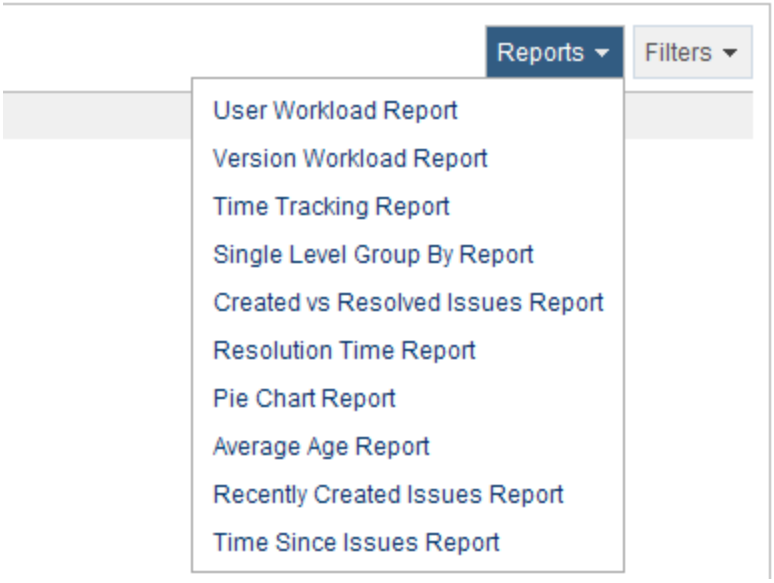


Data Table

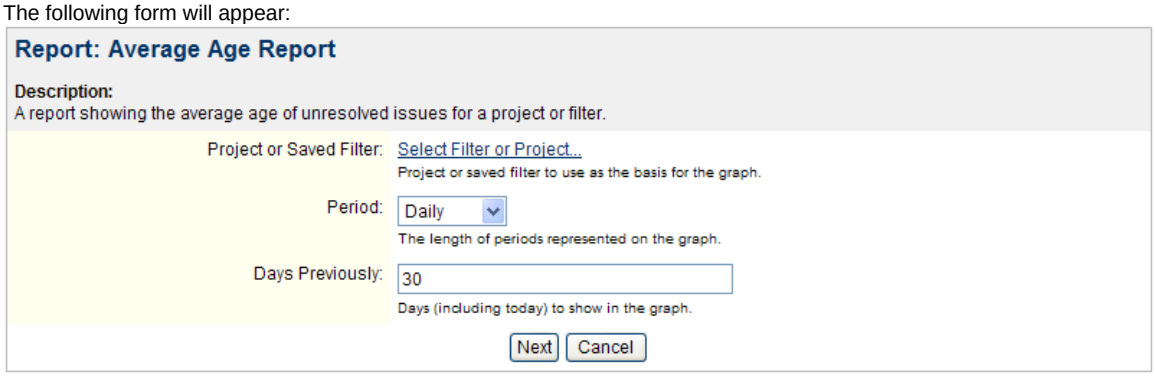
Period	Issues Unresolved	Total Age	Avg. Age
7-January-2009	17	2682	157
8-January-2009	17	2699	158
9-January-2009	15	2553	170
10-January-2009	15	2568	171
11-January-2009	15	2583	172
12-January-2009	15	2598	173
13-January-2009	14	2545	181
14-January-2009	15	2560	170
15-January-2009	15	2575	171

Generating an 'Average Age' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Average Age Report'.**
5. The following form will appear:

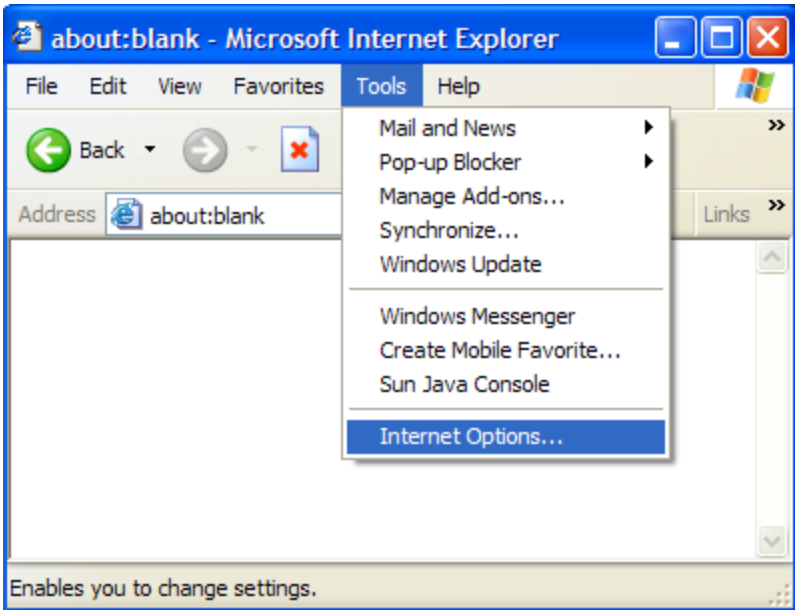


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Period'** field, select the timeframe on which the report will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
9. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.

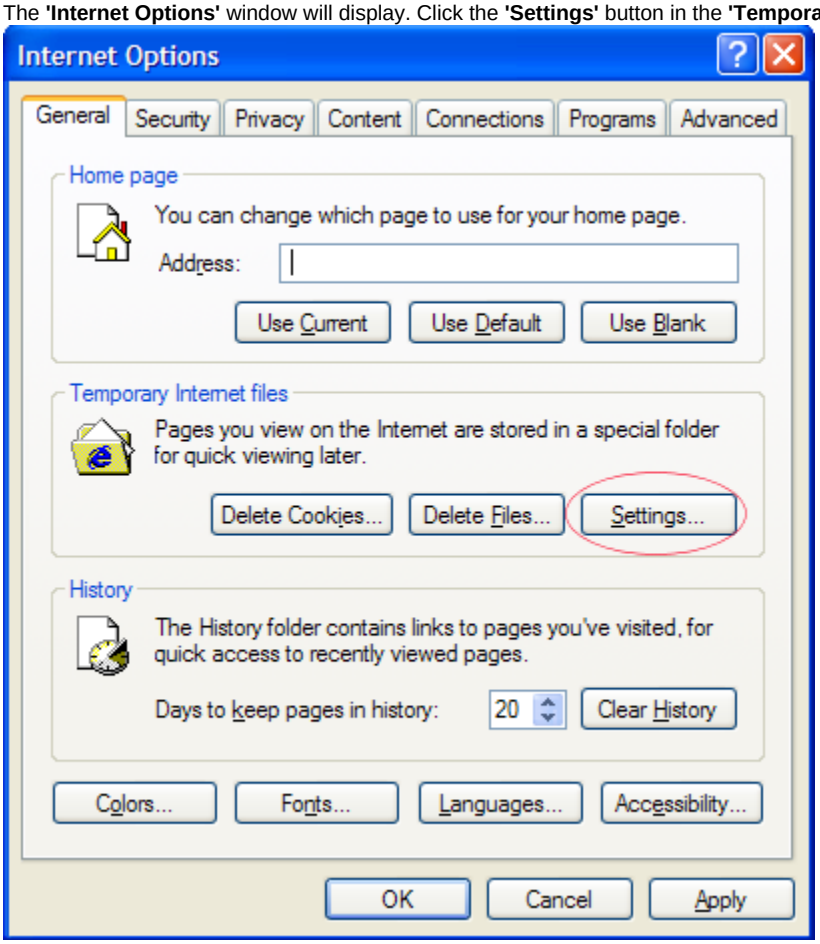
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

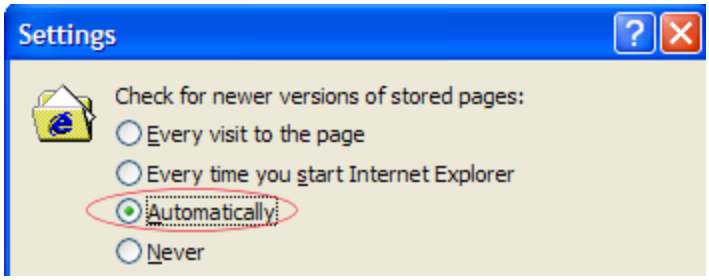
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Recently Created Issues Report

The 'Recently Created Issues' report is a bar chart showing the rate at which issues are being created, as well as how many of those created issues are resolved. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does the 'Recently Created Issues' report look like?

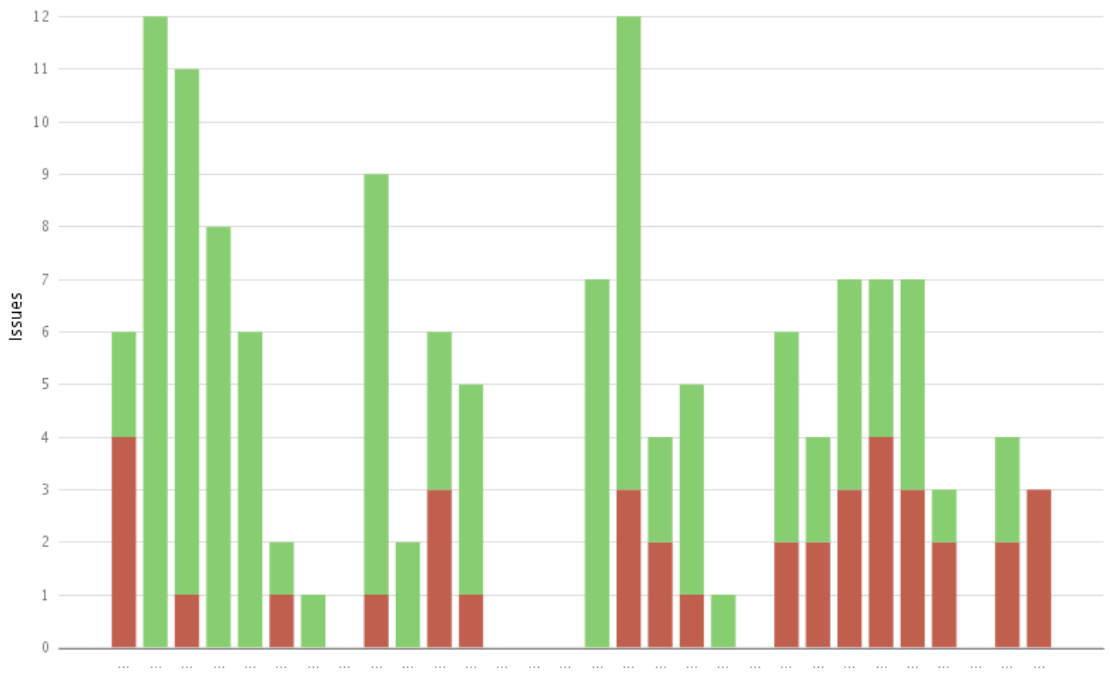
The report generated will look something like this:

Report: Recently Created Issues Report

Project: SysAdmin

Chart

This chart shows the issues created in the last **30** days



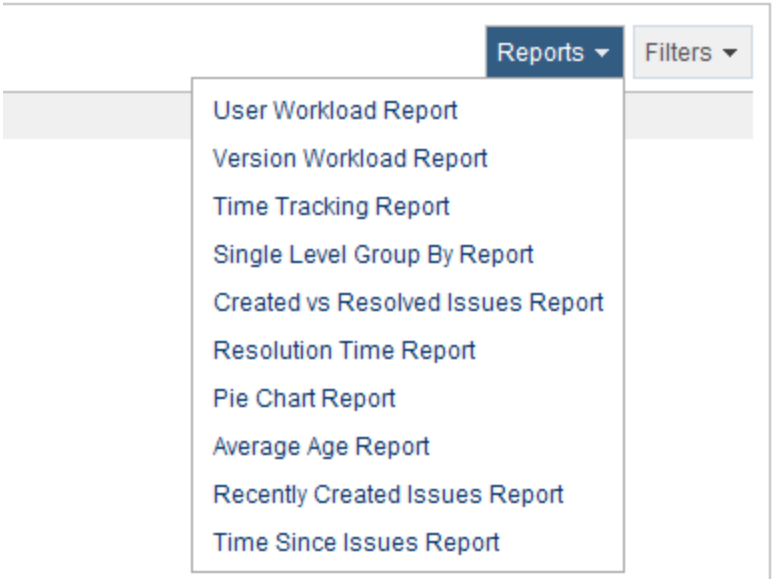
Data Table

Period	Created Issues (Unresolved)	Created Issues (Resolved)
11-January-2009	4	2
12-January-2009	0	12
13-January-2009	1	10
14-January-2009	0	8
15-January-2009	0	6
16-January-2009	1	1
17-January-2009	0	1
18-January-2009	0	0
19-January-2009	1	8
20-January-2009	0	2

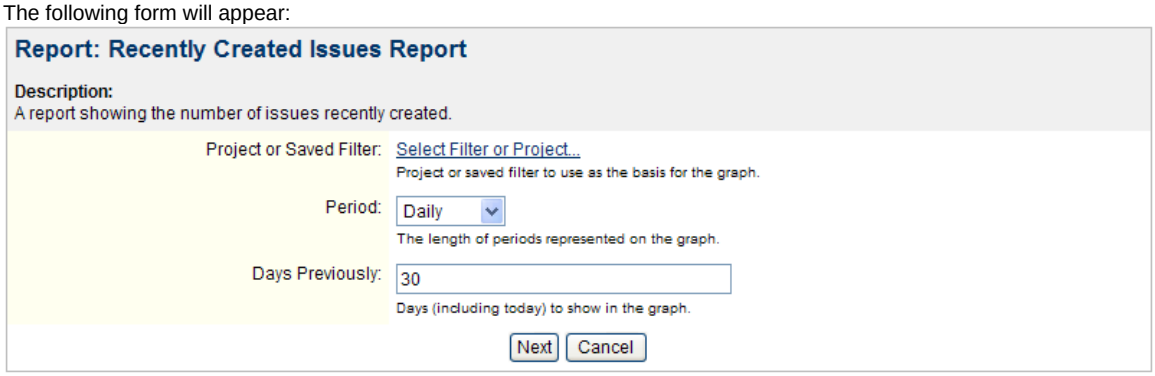
The green portion of the bar shows the created issues which are resolved. The red portion shows created but as yet unresolved issues.

Generating a 'Recently Created Issues' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Recently Created Issues Report'.**
5. The following form will appear:

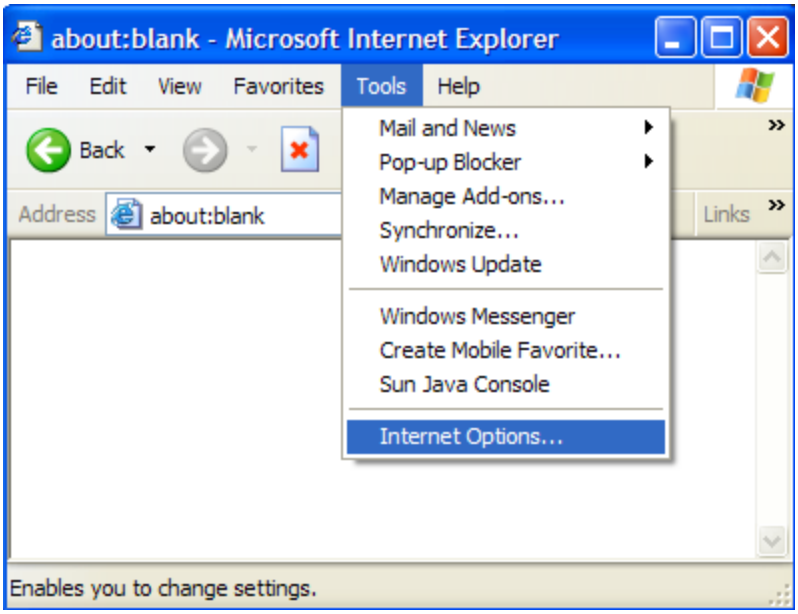


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Period'** field, select the timeframe on which the report will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
9. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.

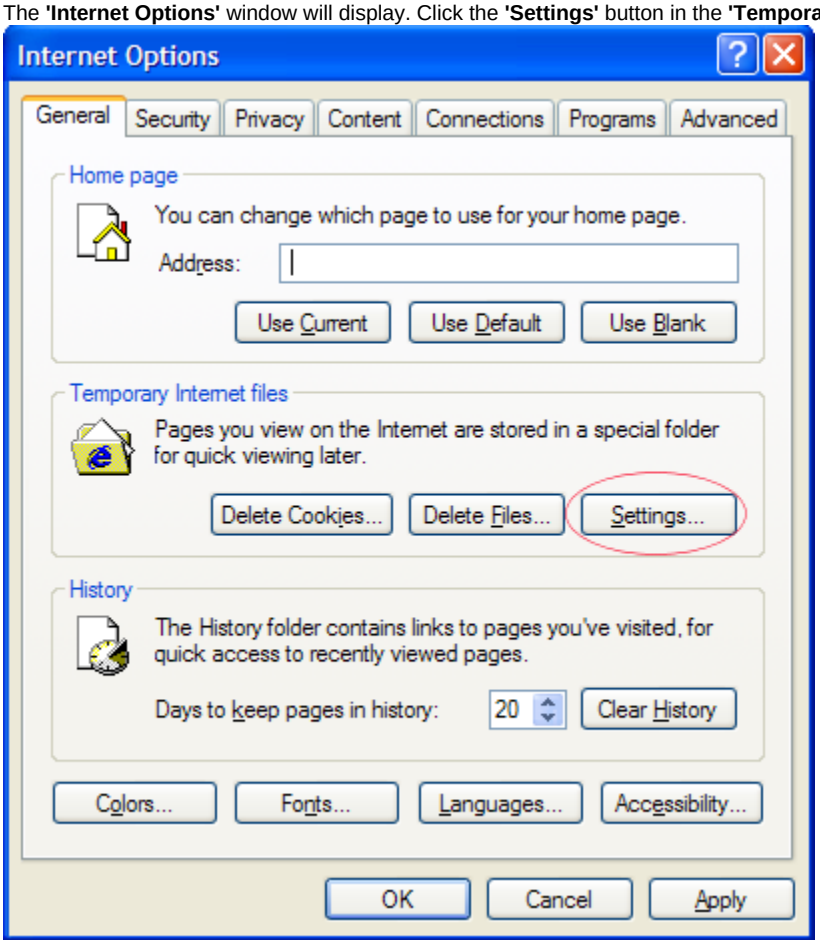
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

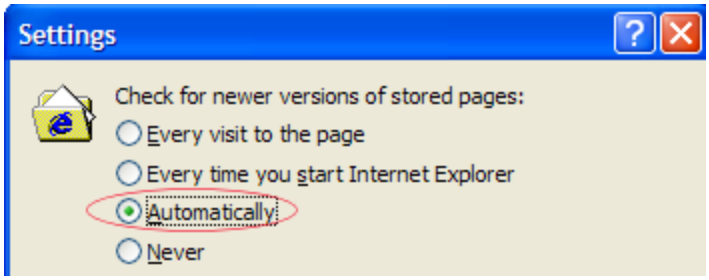
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Time Since Issues Report

The 'Time Since Issues' report is a bar chart showing the number of issues for which your chosen date field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a custom field) was set on a given date. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does the 'Time Since Issues' report look like?

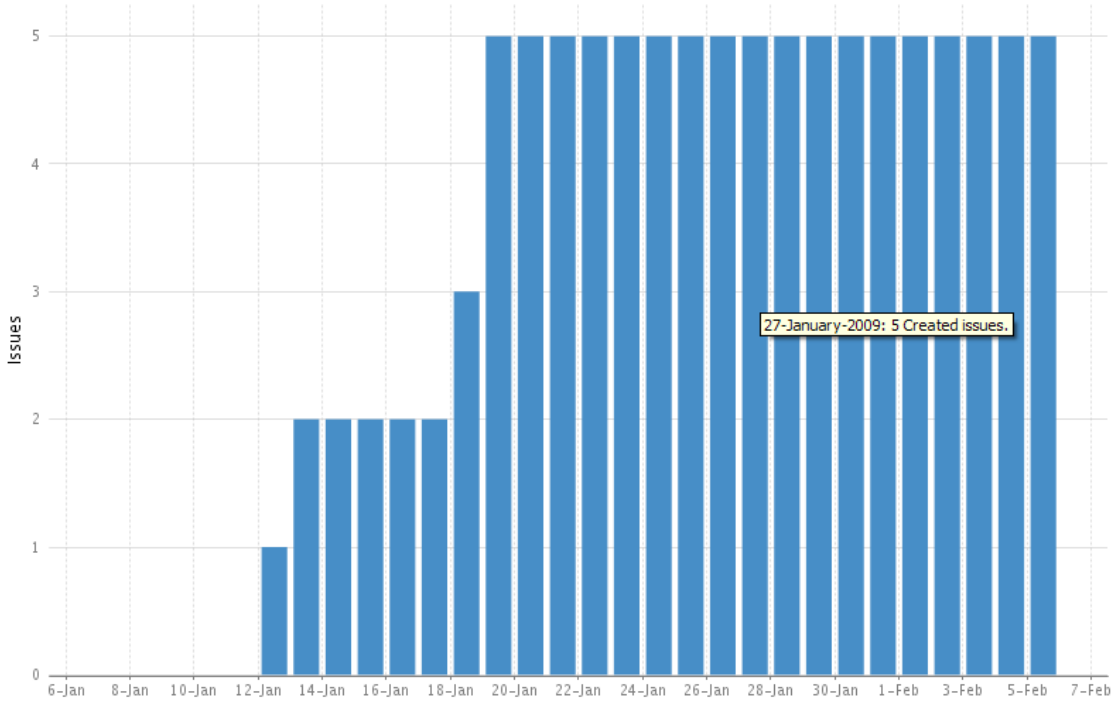
The report generated will look something like this:

Report: Time Since Issues Report

Project: Book Request

Chart:

This chart shows the number of issues based on the **Created** field on a given date over the past **30** days.

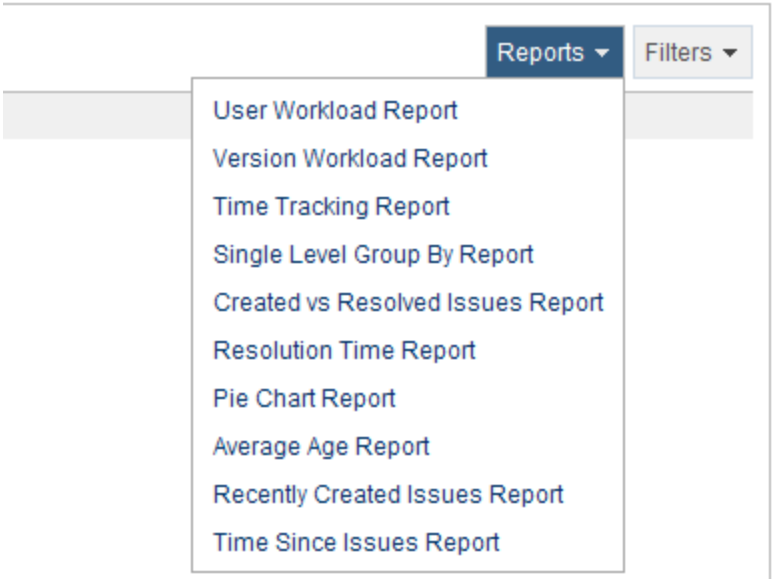


Data Table

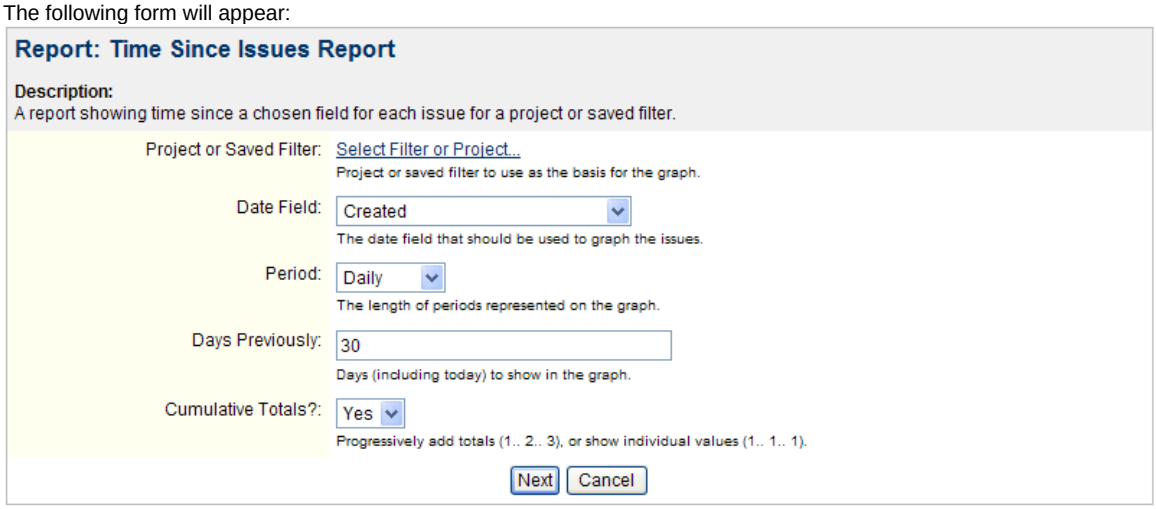
Period	Created
7-January-2009	0
8-January-2009	0
9-January-2009	0
10-January-2009	0
11-January-2009	0
12-January-2009	1
13-January-2009	1

Generating a 'Time Since Issues' report

1. Click **'Browse Project'** in the top navigation panel.
2. Select any project, if presented with a list.
3. Click **'Reports'** at the right of the screen. A popup list will appear:



4. Select **'Time Since Issues Report'.**
5. The following form will appear:

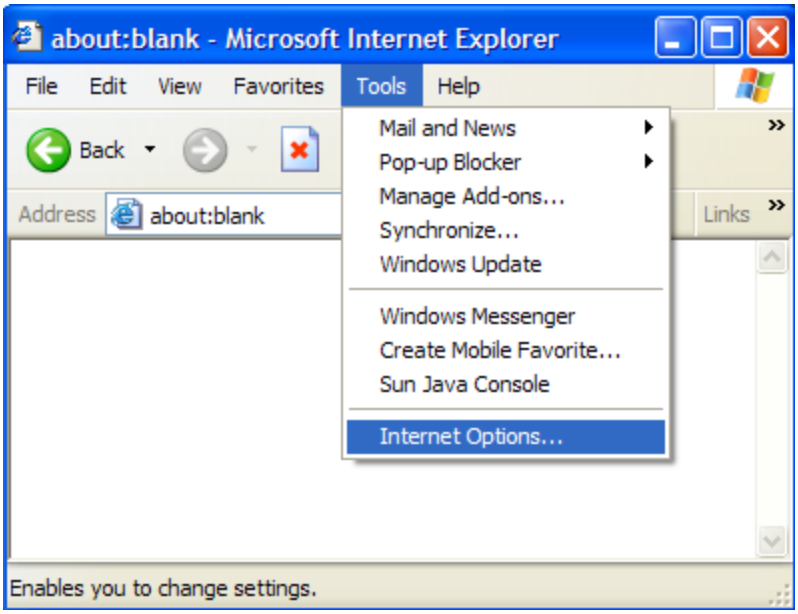


6. Click **'Select Filter or Project'.**
7. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the form.
8. In the **'Date Field'** field, select the date in which you are interested (e.g. **'Created', 'Updated', 'Due' , *'Resolved',** or a custom field of type 'Date').
   *Note: only available if time tracking has been enabled by your JIRA administrator.
9. In the **'Period'** field, select the timeframe on which the report will be based:
   - **'Hourly'**
   - **'Daily'**
   - **'Weekly'**
   - **'Quarterly'**
   - **'Yearly'**
10. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.
11. In the **'Cumulative Totals?'** field, choose either:
    - **'Yes'** to progressively add data to the preceding column; or
    - **'No'** to show just a single value in each column.

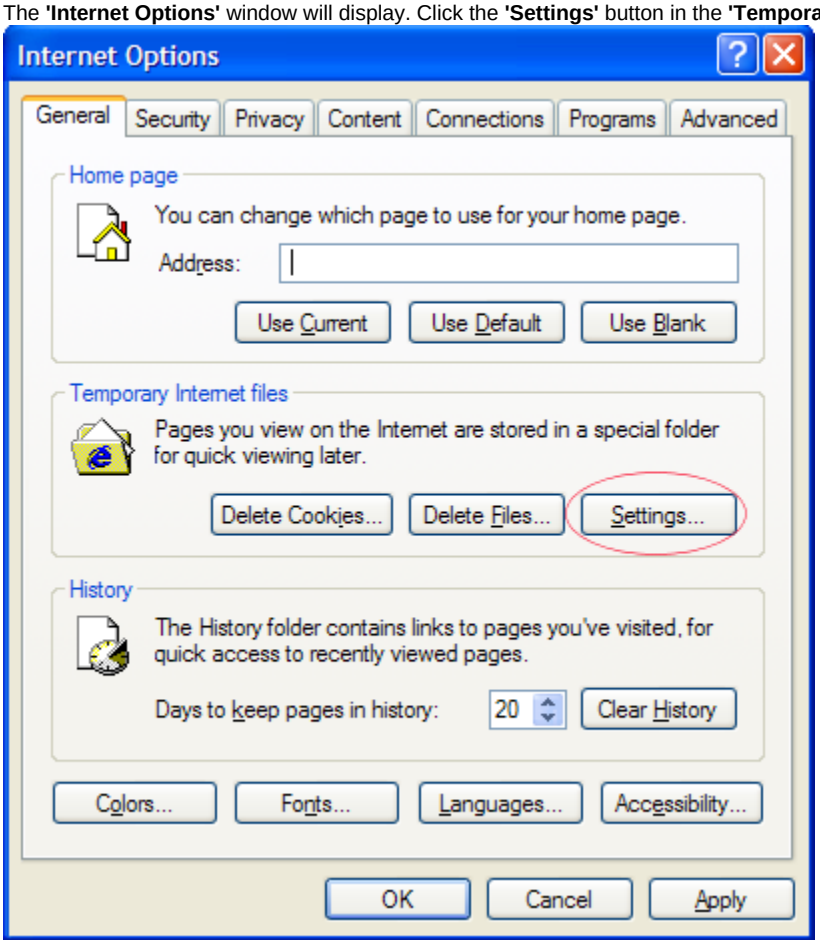
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

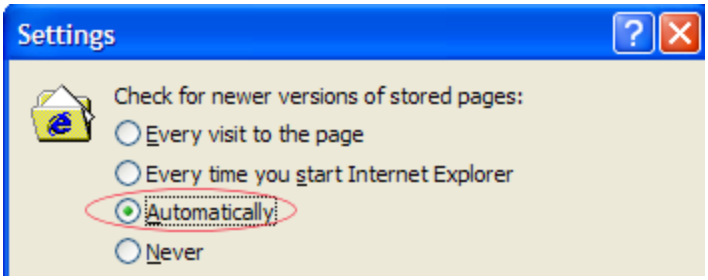
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Browsing a Project

Browsing a project, or its components or versions, shows you summaries of your project's progress.

You can browse the following:

- Project:
  - Summary — Shows recent activity in your project, plus a list of issues that are due soon.
  - Issues — Shows summaries of: all issues in a project, grouped by Status; and unresolved issues, grouped by Assignee, Priority, Version and Component.
  - Road Map — Shows unresolved issues for upcoming versions of a project.
  - Change Log — Shows resolved issues for previous versions of a project.
  - Popular Issues — Shows a project's unresolved issues, ordered by popularity (votes).
  - Builds * — Shows recent Bamboo builds for a given project.
  - FishEye ** — Shows recent FishEye changesets for a given project.
- Component:
  - Component Summary — Shows recent activity in a given component of a project, plus a list of issues that are due soon.
  - Component Issues — Shows issues belonging to a given component of a project.
  - Component Road Map — Shows unresolved issues for a given component, for upcoming versions of the project.
  - Component Change Log — Shows resolved issues for a given component, for previous versions of the project.
  - Component Popular Issues — Shows unresolved issues for a given component, ordered by popularity (votes).
- Version:
  - Version Summary — Shows recent activity in a given version of a project, plus a list of issues that are due soon.
  - Version Issues — Shows issues belonging to a given version of a project.
  - Version Popular Issues — Shows unresolved issues for a given version, ordered by popularity (votes).
  - Version Builds * — Shows recent Bamboo builds for a given version.

** Only available if your organisation uses Atlassian Bamboo and your administrator has integrated Bamboo with JIRA.*

*** Only available if your organisation uses Atlassian FishEye and your administrator has integrated FishEye with JIRA.*

See also

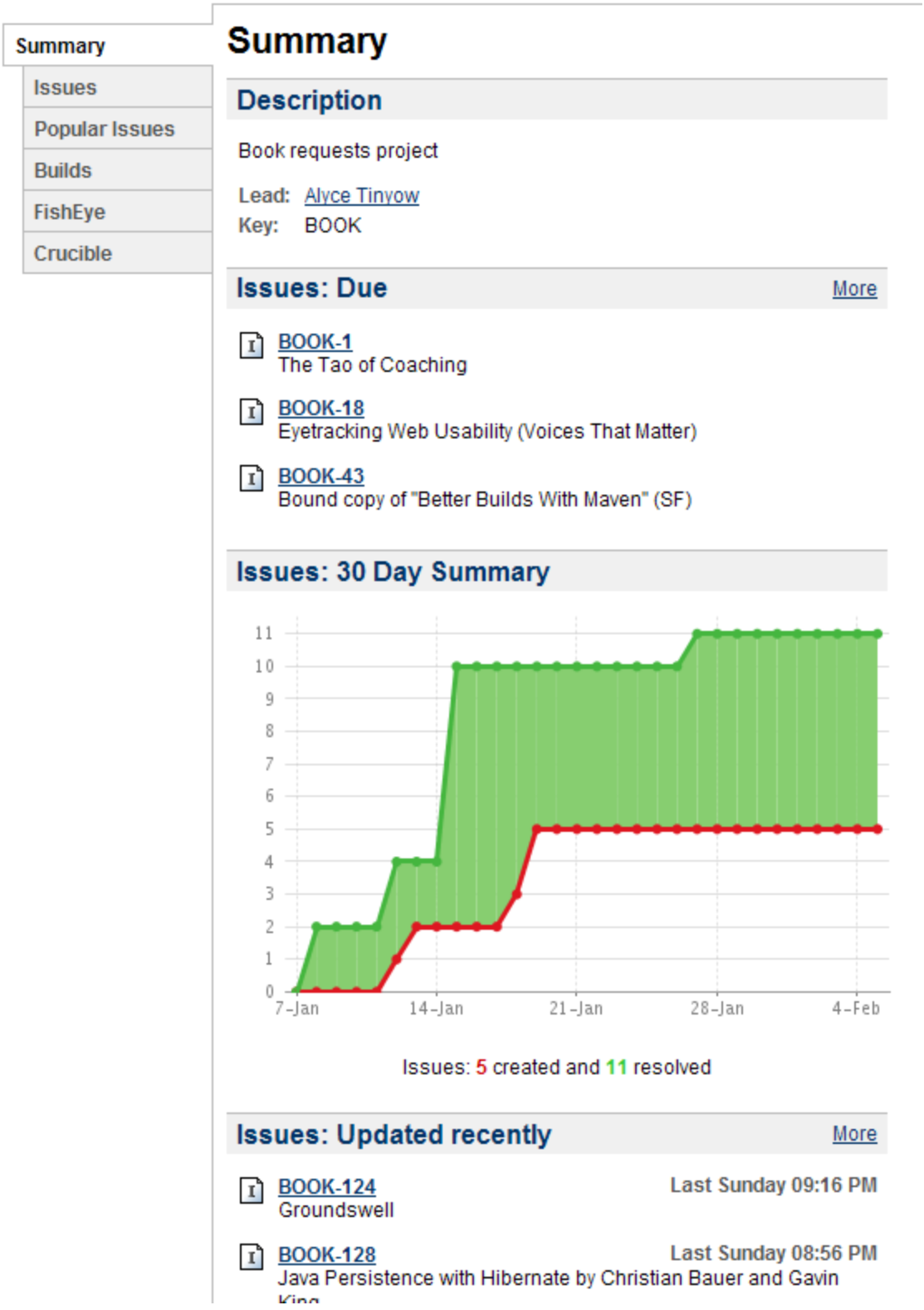
- JIRA Reports Overview

Browsing a Project's Summary

JIRA's **Summary** report shows recent activity in your project, plus a list of issues that are due soon.

To browse the Summary,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Summary'** tab. You will see something like this:



You can click **'More'** to go through to the Issue Navigator and see the full list of issues.

See also

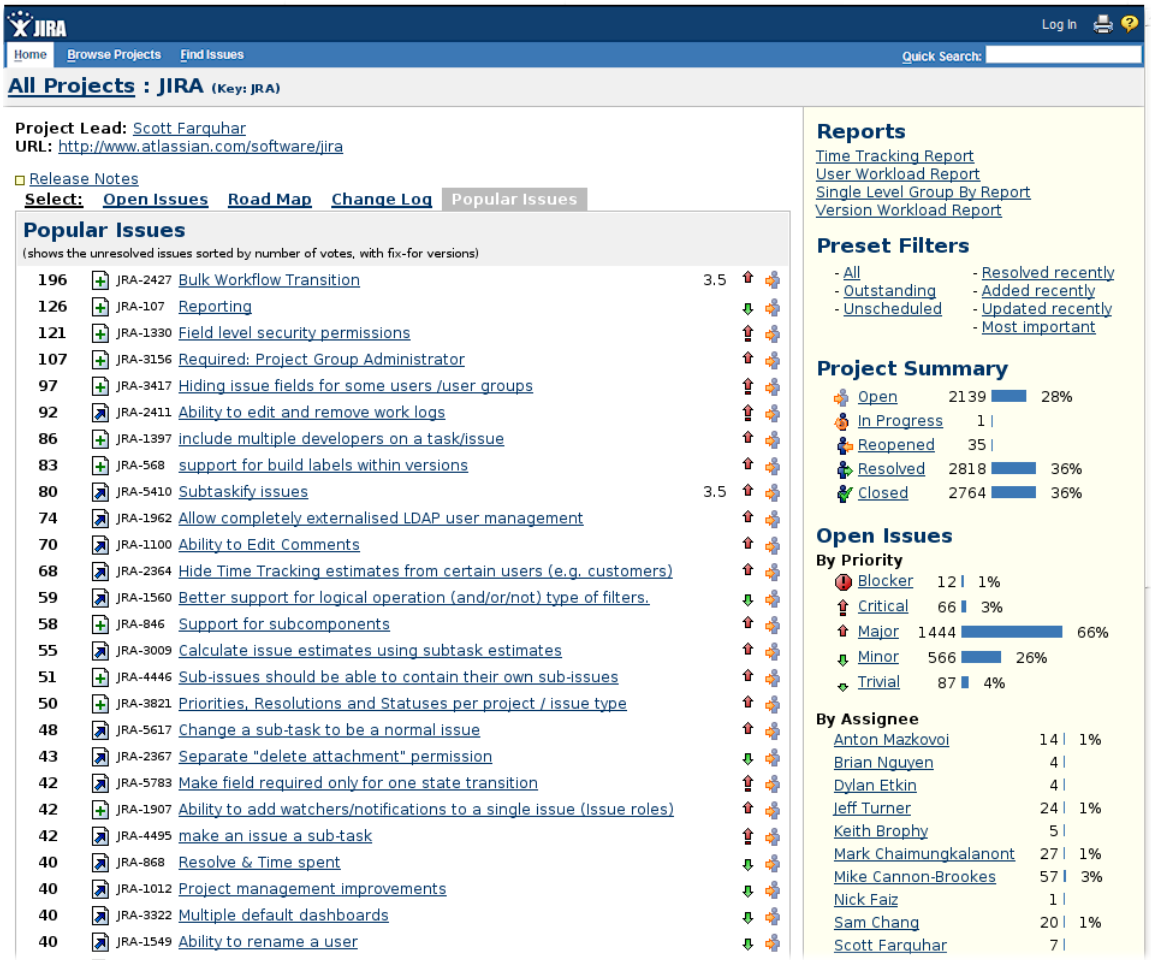
- Browsing a Project
- JIRA Reports Overview

Browsing a Project's Popular Issues

JIRA's **Popular Issues** report shows unresolved issues in a project, sorted by number of votes. (Note: This report is only visible if voting is enabled).

To browse the Popular Issues,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Popular Issues'** tab. You will see something like this:



http://jira.atlassian.com" />

A live version of this example can be seen online.

See also

- Browsing a Project
- JIRA Reports Overview

Browsing a Project's Bamboo Builds

If your organisation uses Atlassian's Bamboo and your administrator has integrated Bamboo with JIRA, JIRA enables you to view the Bamboo build plan status and recent build activity for a project.

The **Builds** tab provides you with the build information related to the project, including:

- the list of the builds which are related to the project, i.e. builds that have issues from the project linked to them (either as 'Fixed' or 'Related'). (See the Bamboo documentation for instructions on linking issues to builds).
- The latest status of the build plans for the related builds, i.e. the build plan contains a build that has an issue from the project linked to it.

To view the Bamboo build information related to a project,

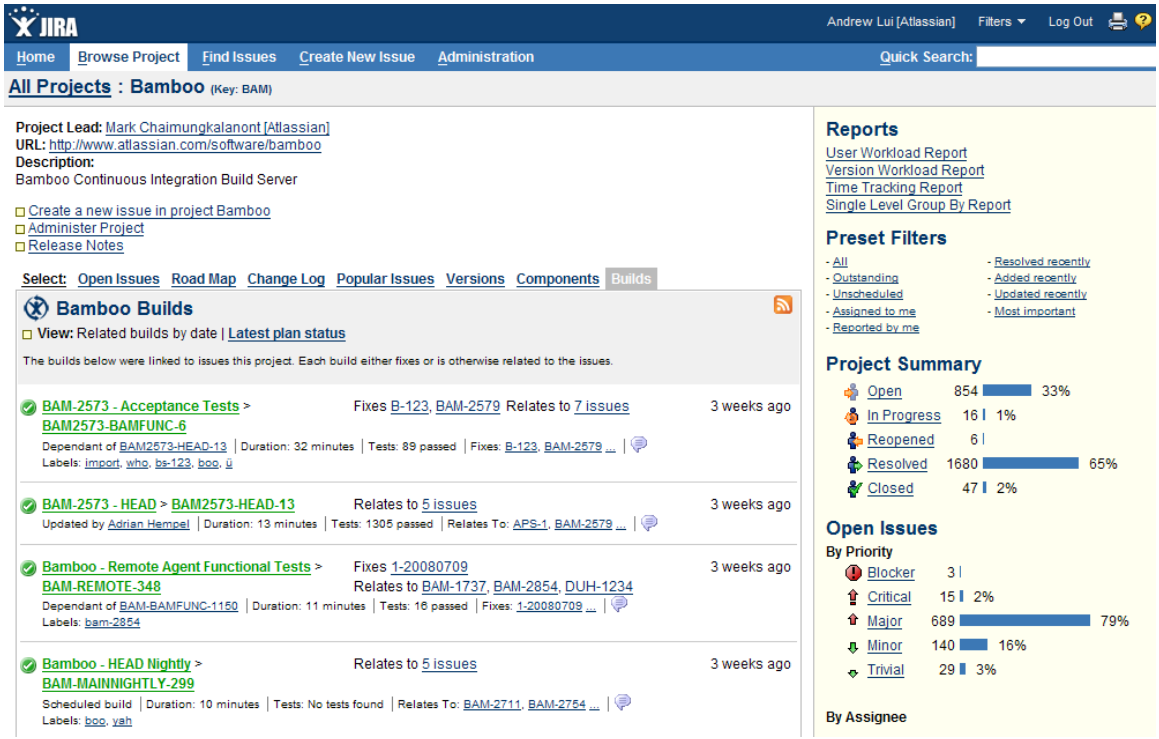
1. Click **Browse Projects** in the top navigation panel.

2. Select the project you are interested in, if presented with a list.
3. Click the **Builds** tab.

> If you cannot see the **Builds** tab, your administrator may need to add the **'View Version Control'** permission to your project.

Viewing the Builds related to the Project

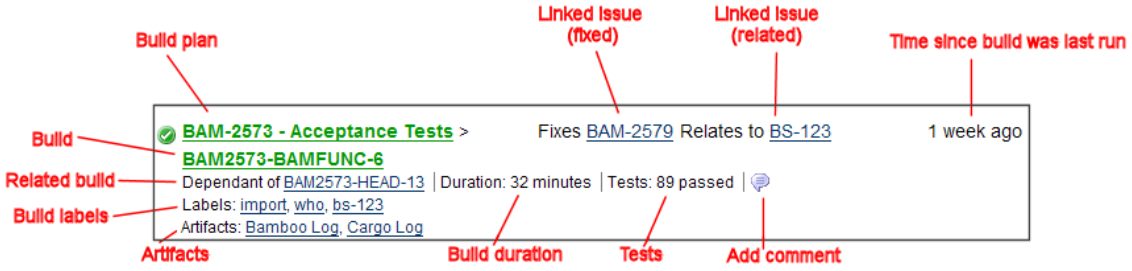
By default, the **Builds** tab will display the list of related builds, ordered by build date in descending order.



You can set up an RSS feed to track this information by clicking on the RSS icon in the top right of tab panel. Each entry in the list will display information about the related build, including:

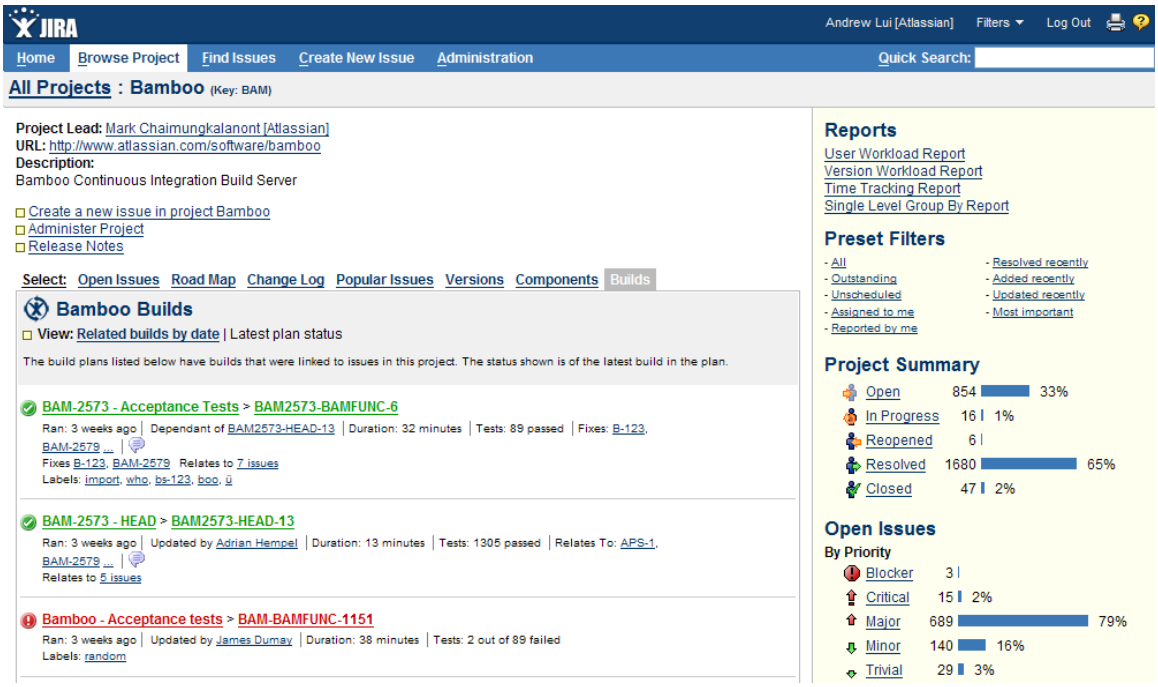
- the build name and name of the build plan
- when the build was last run
- summary information, such as related builds, duration of the build, tests passed
- build labels (if any)
- links to build artifacts (if any)

See the diagram below for details:



Viewing the Status of Build Plans related to the Project

To view the status of build plans related to the project, click the **'Latest plan status'** link at the top of the **Build** tab. The build plans listed will show the status of the build plan, (i.e. status of the latest build), including information about the latest build in the plan (similar to the diagram above). Build plans will be sorted by the plan name.



See also:

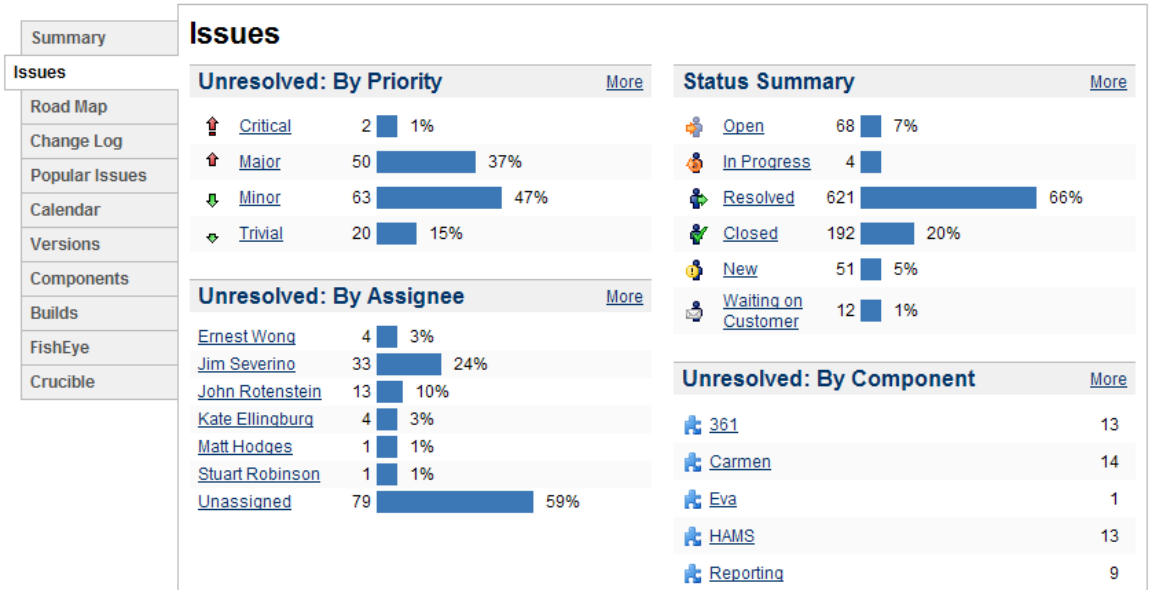
- Viewing the Bamboo Builds related to an Issue
- Browsing a Version's Bamboo Builds

Browsing a Project's Issues

JIRA's **Issues** report shows summaries of: all issues in a project, grouped by Status; and unresolved issues, grouped by Assignee, Priority, Version and Component.

To browse a project's Issues,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Issues'** tab. You will see something like this:



To see which issues have a particular priority, assignee or status, or belong to a particular component or version of the project, click the name of the relevant priority/assignee/status/component/version.

See also

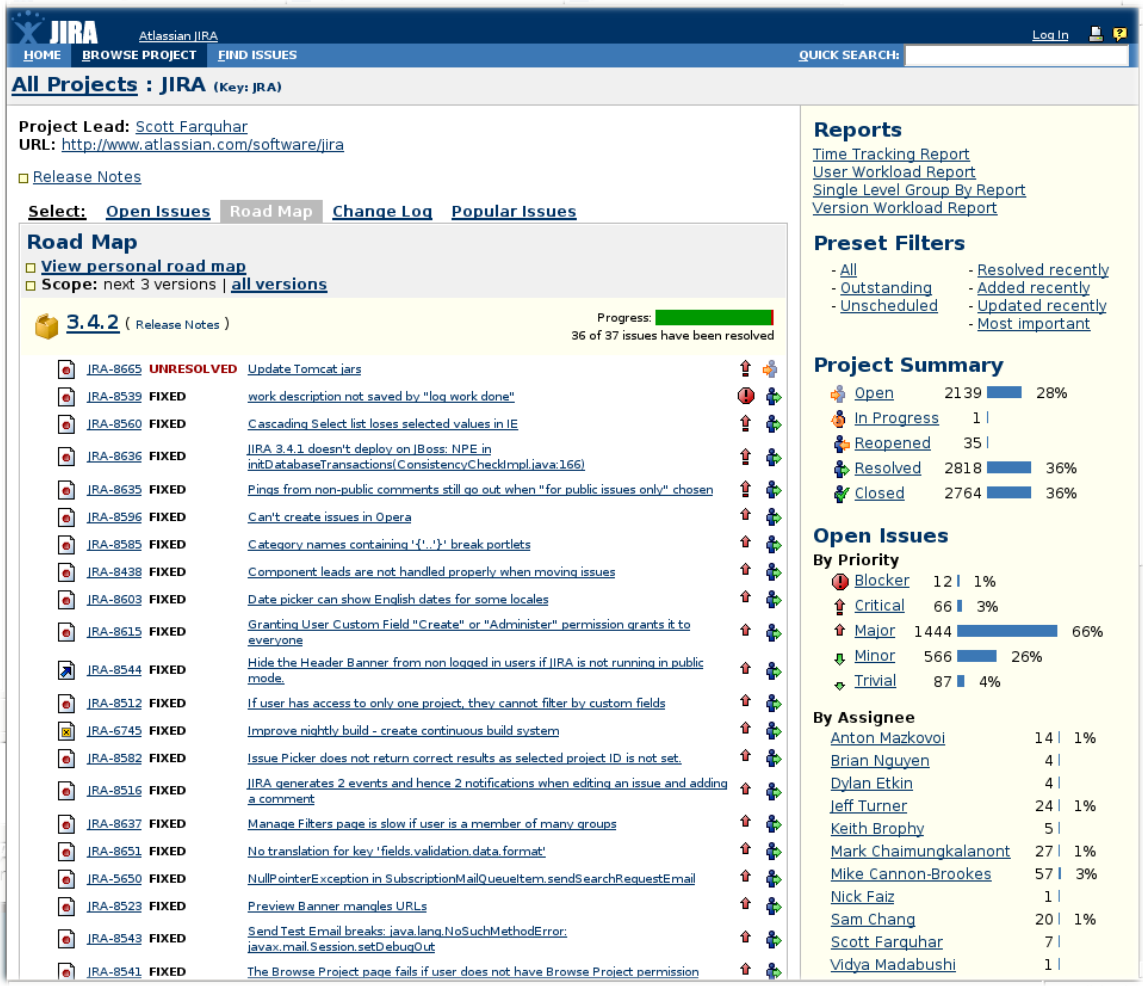
- Browsing a Project
- JIRA Reports Overview

Browsing a Project's Road Map

JIRA provides a **Road Map** for each project, which shows issues scheduled for the next ten unreleased versions (whereas the Change Log shows released versions). The Road Map provide an overview of progress made towards releasing a version.

To browse a project's Road Map,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Road Map'** tab. You will see something like this:



http://jira.atlassian.com" />

A live version of this example can be seen online.

To see the roadmap for all unreleased versions (not just the next ten), click 'all versions'.

If your administrator has hidden the '**Fix For Version**' field, the Road Map report will not be available.

Personal Road Map report

- To see issues assigned to you for the next three unreleased versions of a project, click the **'View Personal Road Map'** link in the global

Road Map report shown above.

See also

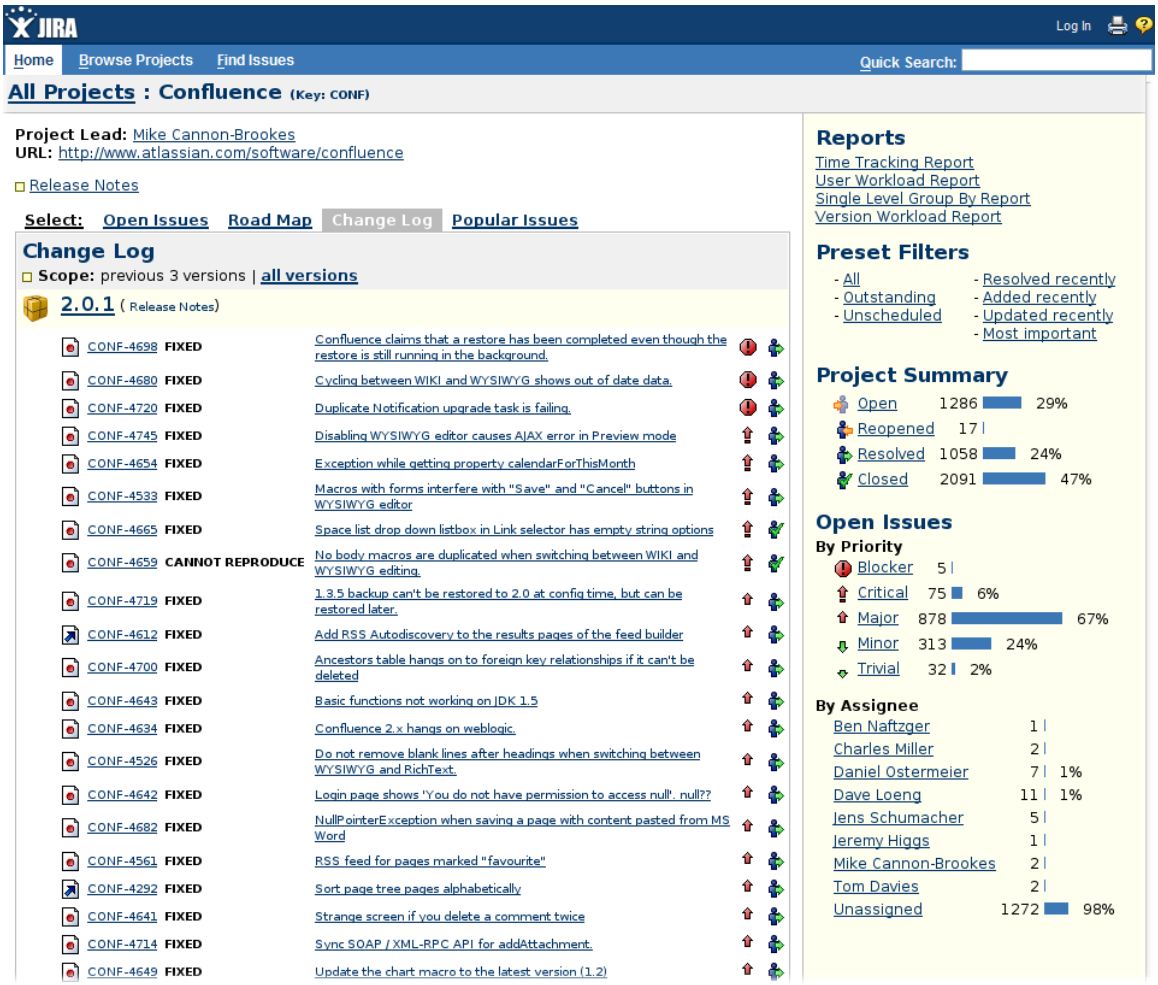
- The Change Log — looking back at recent releases rather than forward
- Browsing a Project
- JIRA Reports Overview

Browsing a Project's Change Log

JIRA's **Change Log** report shows resolved issues in the last ten released versions of a project. Whereas the Road Map looks forward, the Change Log looks back, giving an overall view of issues resolved in recent versions.

To browse a project's Change Log,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Change Log'** tab. You will see something like this:



A live version of this example can be seen online.

> If your administrator has hidden the '**Fix For Version**' field, the Change Log report will not be available.

See also

- The Road Map — looking forward to future releases
- Browsing a Project
- JIRA Reports Overview

Browsing a Project's FishEye Changesets

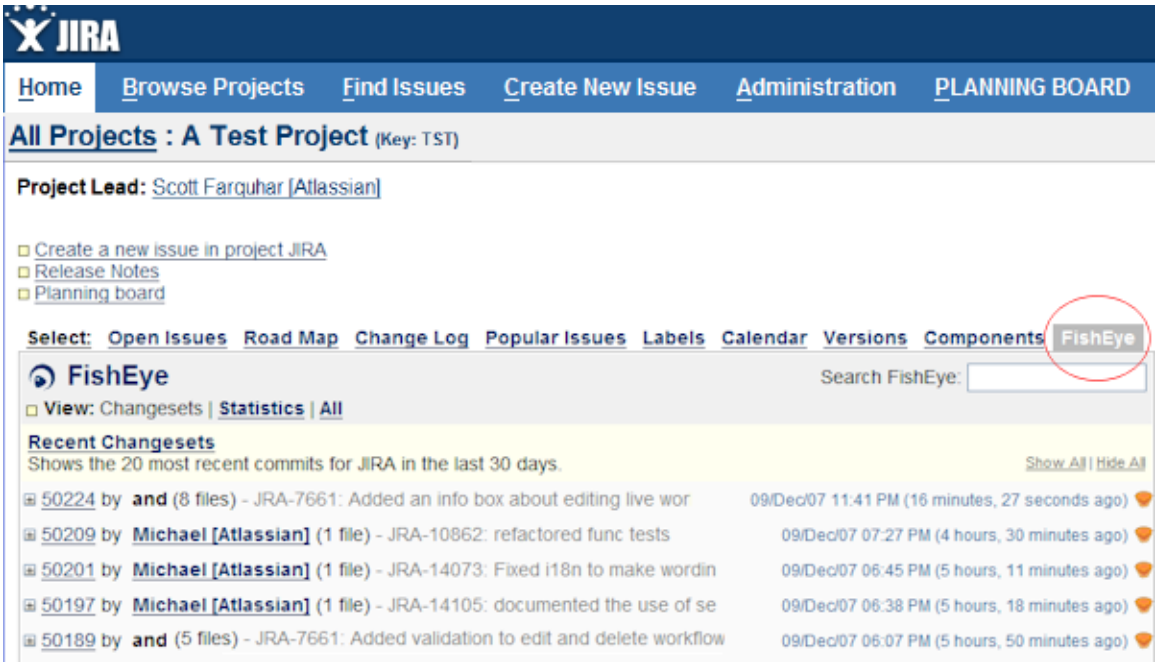
JIRA's **Changeset** report allows you to view recent changeset activity for a project (that is, where a JIRA issue key belonging to the project was referenced in the commit message), if you are using a source-code repository together with Atlassian FishEye. You can:

- View all 'Recent Changesets' for all repository changesets across the entire project.
- View 'Activity Statistics' on LOC (lines-of-code), files or commits for the project, issue or author.
- Search the FishEye repository linked to the JIRA project currently being browsed.

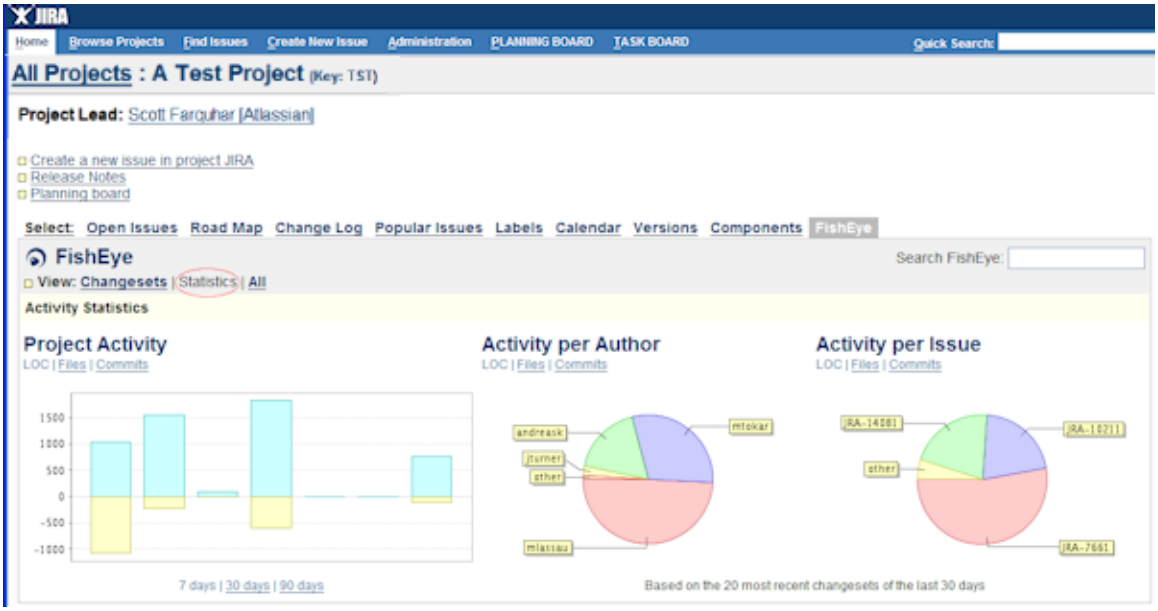
> To be able to view the changeset report, your JIRA administrator must have configured the FishEye plugin appropriately. You will also need the 'View Version Control' permission in the appropriate projects.

To view the changeset activity for a project,

1. Click **Browse Projects** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **FishEye** tab.
   By default, you will see a listing of the most recent changesets for a project:



You can also view the Activity Statistics on LOC, Files or Commits for the project, by clicking the 'Statistics' link. The Activity Statistics for the project will display:



If you wish to see the above two sets of information together, you can click on the 'All' link to view it all on one page.

See also

- Viewing an Issue's FishEye Changesets

Browsing a Version's Summary

JIRA provides a **Summary** of each version of a project, which shows recent activity in that version, plus a list of issues that are due soon.

To browse a version's Summary,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Versions'** tab.
4. A list of versions will be displayed. Click the name of the version in which you are interested.
5. Click the **'Summary'** tab. You will see something like this:

You can click **'More'** to go through to the Issue Navigator and see the full list of issues.

See also

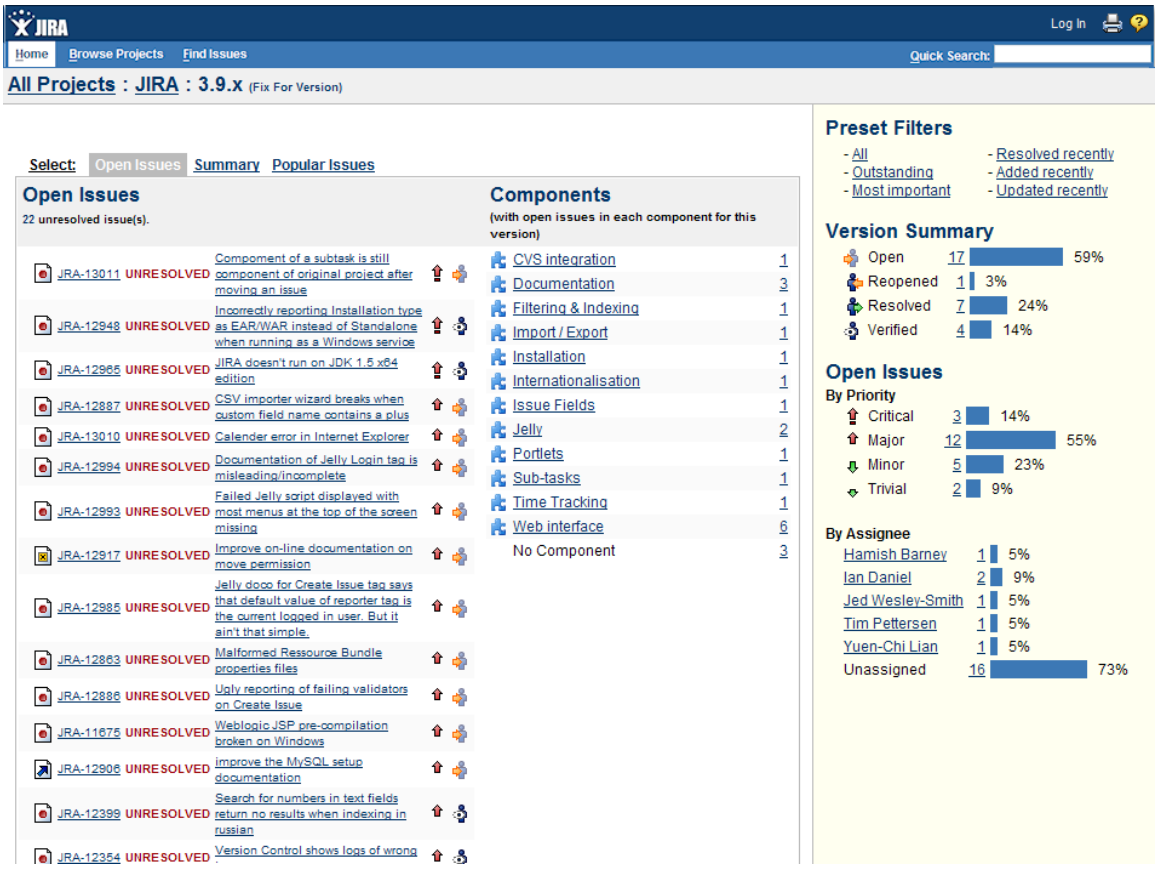
- Browsing a Project
- JIRA Reports Overview

Browsing a Version's Issues

JIRA provides a list of issues for each version of a project.

To browse a version's issues,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Versions'** tab.
4. A list of versions will be displayed. Click the name of the version in which you are interested.
5. Click the **'Issues'** tab. You will see something like this:



To see which issues have a particular priority, assignee or status, or belong to a particular component of the project, click the name of the relevant priority/assignee/status/component.

See also

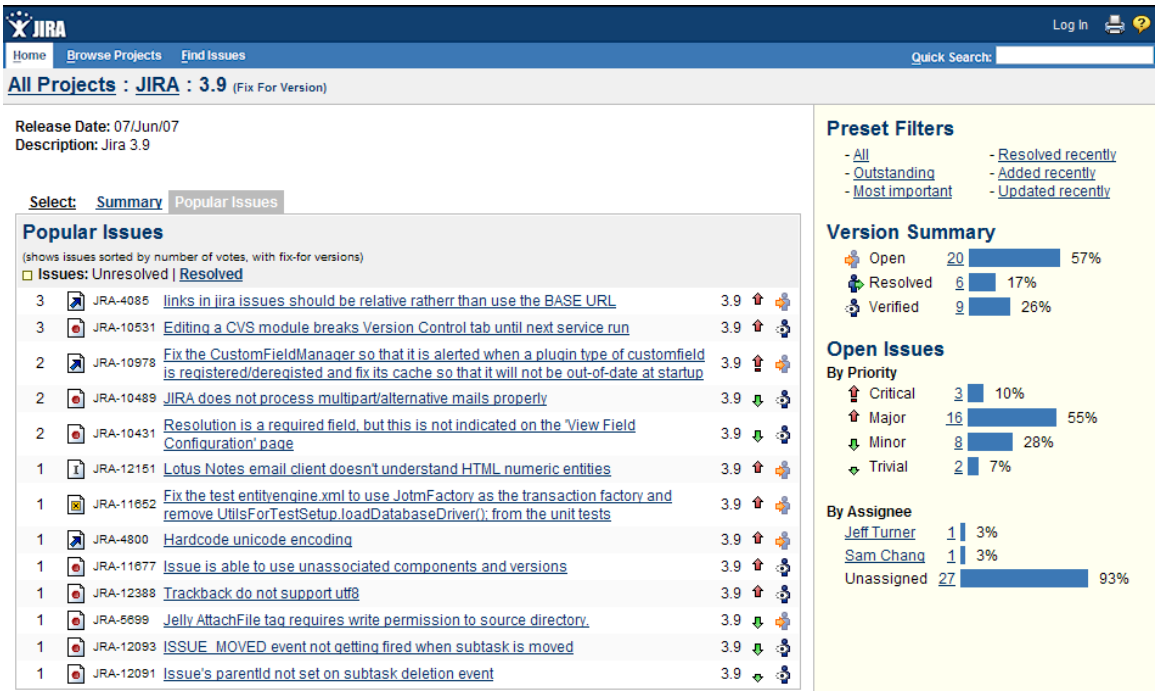
- Browsing a Project
- JIRA Reports Overview

Browsing a Version's Popular Issues

JIRA's **Popular Issues** report shows unresolved issues in a given version of a project, sorted by number of votes. It is particularly useful on public JIRA installations. This report is only visible if voting is enabled.

To browse a version's Popular Issues,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Versions'** tab.
4. A list of versions will be displayed. Click the name of the version in which you are interested.
5. Click the **'Popular Issues'** tab. You will see something like this:



To see resolved popular issues (instead of unresolved popular issues), click **'View Resolved Issues'.**

See also

- Browsing a Project
- JIRA Reports Overview

Browsing a Version's Bamboo Builds

If your organisation uses Atlassian's Bamboo and your administrator has integrated Bamboo with JIRA, JIRA enables you to view the Bamboo build plan status and recent build activity for a version of a project.

The **Builds** tab provides you with a list of the builds which are related to the project version, including:

- the list of the builds which are related to the version, i.e. builds that have issues from the project version linked to them (either as 'Fixed' or 'Related'). (See the Bamboo documentation for instructions on linking issues to builds.)
- the latest status of the build plans for the related builds, i.e. the build plan contains a build that has a project issue linked to it. The status of a build plan for a version is determined as follows:
    - If the project version has not been released — the build plan status is the status of the latest build in the Bamboo build plan, regardless of whether the latest build is related to the version (i.e. has issues from the project version linked to it).
    - If the project version has been released — the build plan status is the status of the latest build in the Bamboo build plan, that is related to the version (i.e. has issues from the project version linked to it) and is prior to or equal to the release date (or current date, if there is no release date).

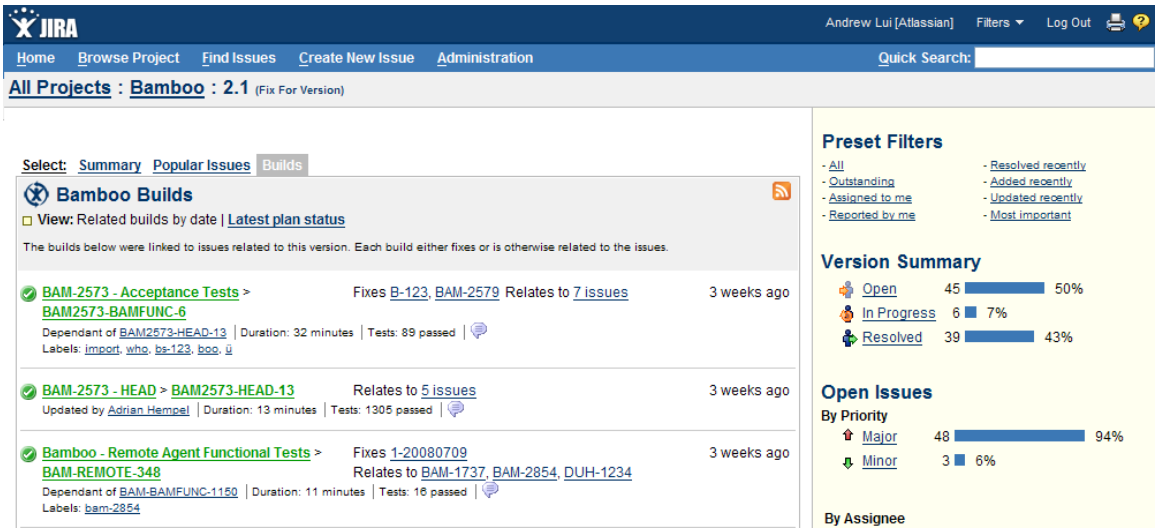
To view the Bamboo build information related to a version,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Versions'** tab.
4. A list of versions will be displayed. Click the name of the version in which you are interested.
5. Click the **Builds** tab.

If you cannot see the **Builds** tab, your administrator may need to add the **'View Version Control'** permission to your project.

Viewing the Builds related to the Project Version

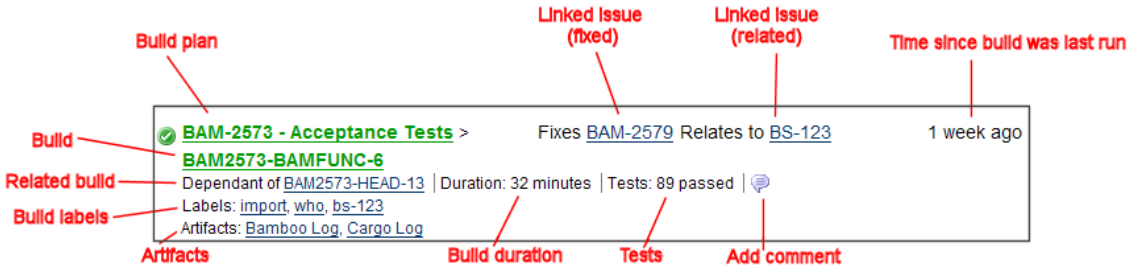
By default, the **Builds** tab will display the list of related builds, ordered by build date in descending order.



You can set up an RSS feed to track this information by clicking on the RSS icon in the top right of tab panel. Each entry in the list will display information about the related build, including:

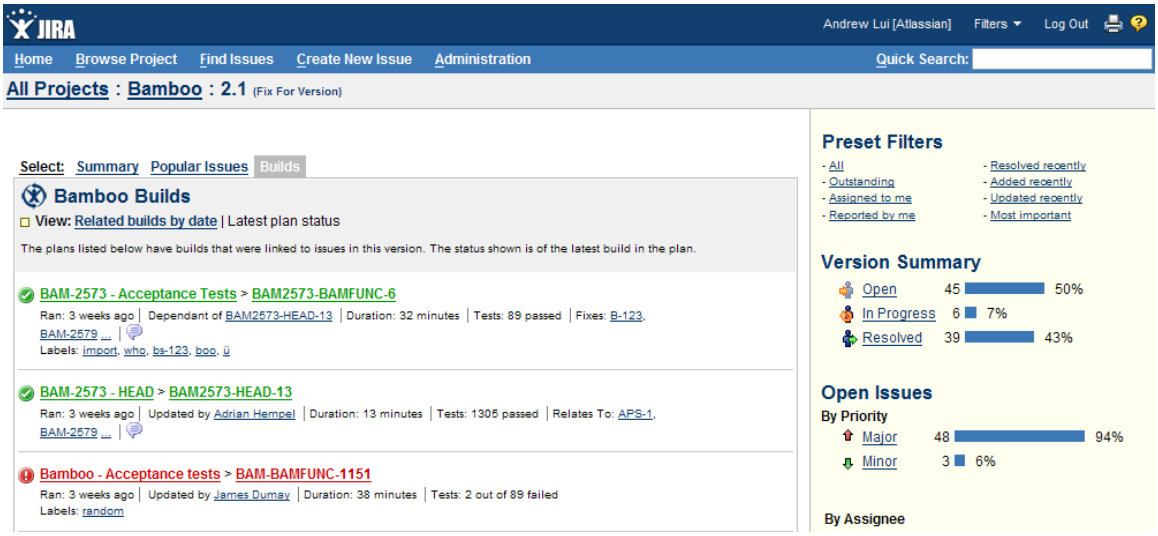
- the build name and name of the build plan
- when the build was last run
- summary information, such as related builds, duration of the build, tests passed
- build labels (if any)
- links to build artifacts (if any)

See the diagram below for details:



Viewing the Status of Build Plans related to the Project Version

To view the status of build plans related to the project version, click the **'Latest plan status'** link at the top of the **Build** tab. The build plans listed will show the status of the Build Plan, including information about the latest build in the plan (similar to the diagram above). Build plans will be sorted by plan name.



See also:

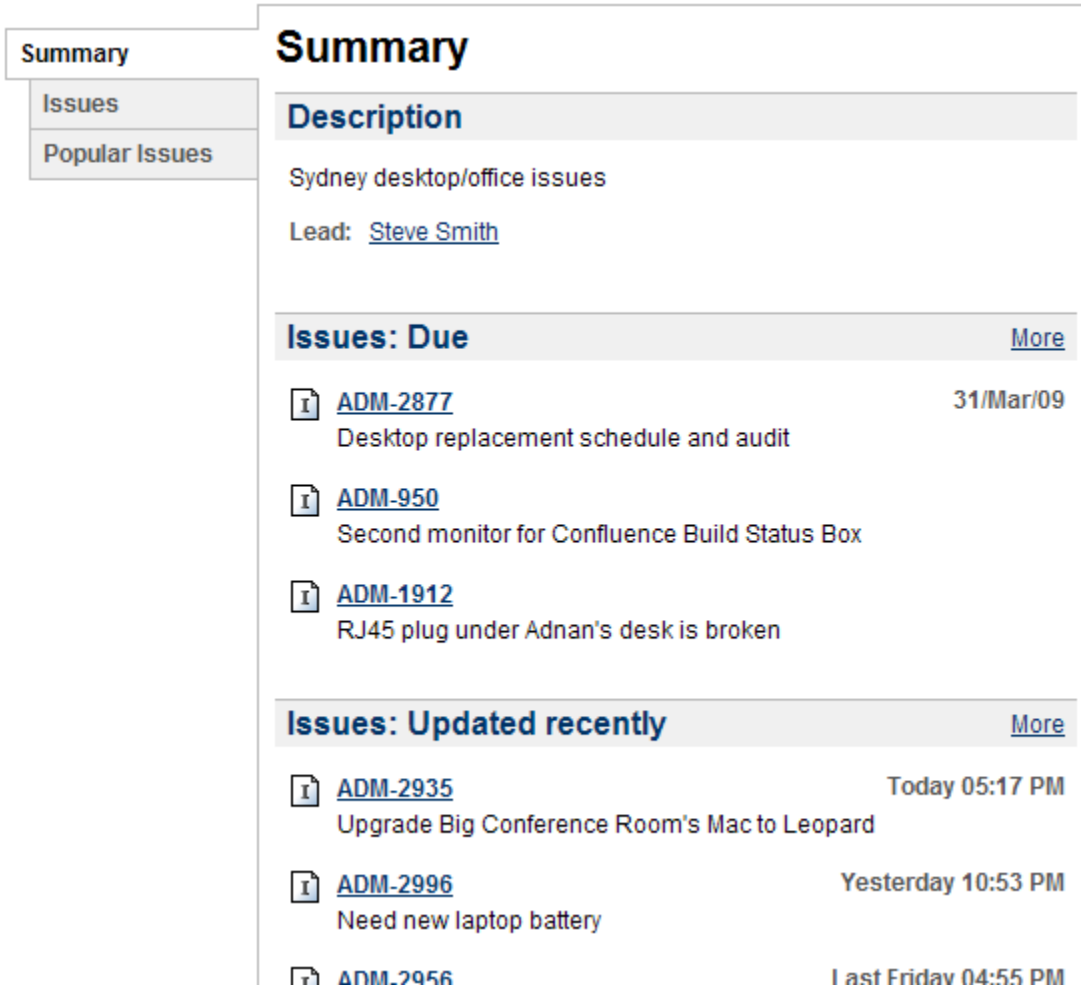
- Viewing the Bamboo Builds related to an Issue
- Browsing a Project's Bamboo Builds

Browsing a Component's Summary

JIRA provides a **Summary** of each component of a project, which shows recent activity in the component, plus a list of issues that are due soon.

To browse a component's Summary,

1. Click on **'Browse Projects'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Component'** tab.
4. A list of components will be displayed. Click the name of the component in which you are interested.
5. Click the **'Summary'** tab. You will see something like this:



You can click **'More'** to go through to the Issue Navigator and see the full list of issues.

See also

- Browsing a Project
- JIRA Reports Overview

Browsing a Component's Issues

JIRA provides a list of all the issues for each component of a project.

To browse a component's issues,

1. Click **'Browse Project'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Components'** tab.
4. A list of components will be displayed. Click the name of the component in which you are interested.
5. Click the **'Issues'** tab. You will see something like this:

To see which issues belong to a particular version of the project, click the component name.

See also

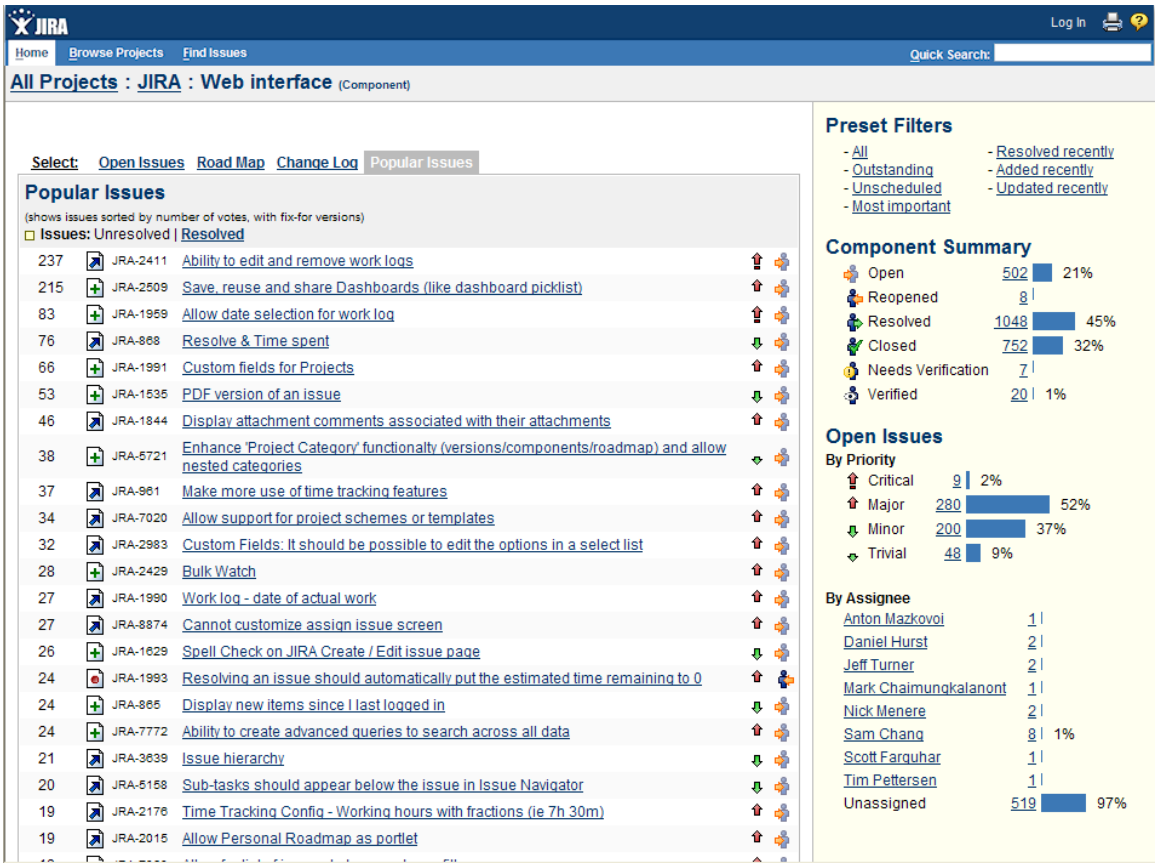
- Browsing a Project
- JIRA Reports Overview

Browsing a Component's Popular Issues

JIRA's **Popular Issues** report shows unresolved issues in a given component of a project, sorted by number of votes. It is particularly useful on public JIRA installations. (Note: this report is only visible if voting is enabled).

To browse a component's Popular Issues,

1. Click **'Browse Project'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Components'** tab.
4. A list of components will be displayed. Click the name of the component in which you are interested.
5. Click the **'Popular Issues'** tab. You will see something like this:



To see resolved popular issues (instead of unresolved popular issues), click **'View Resolved Issues'.**

See also

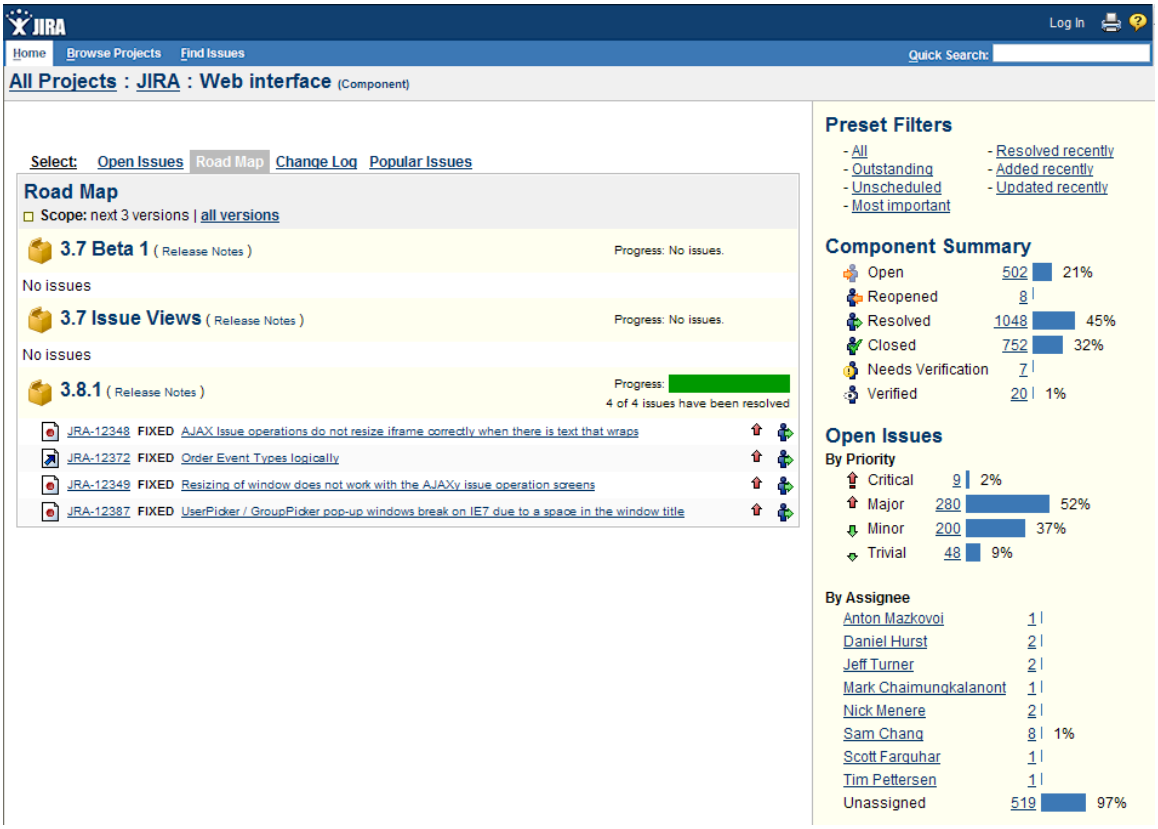
- Browsing a Project
- JIRA Reports Overview

Browsing a Component's Road Map

JIRA provides a **Road Map** for each component of a project, which shows issues scheduled for the next ten unreleased versions (whereas the Change Log shows released versions). The Road Map provides an overview of progress made towards releasing a version.

To browse a component's Road Map,

1. Click **'Browse Project'** in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **'Components'** tab.
4. A list of components will be displayed. Click the name of the component in which you are interested.
5. Click the **'Road Map'** tab. You will see something like this:



To see the Road Map for all unreleased versions (not just the next ten), click 'all versions'.

If your administrator has hidden the '**Fix For Version**' field, the Road Map report will not be available.

See also

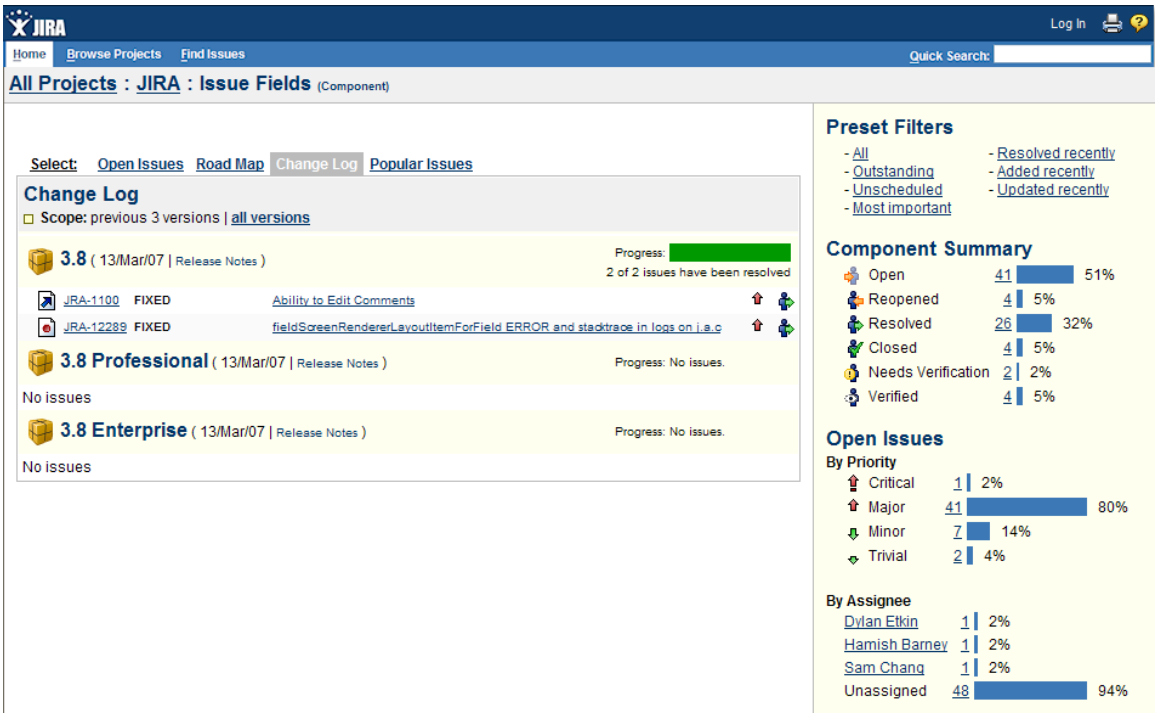
- The Change Log — looking back at recent releases rather than forward
- Browsing a Project
- JIRA Reports Overview

Browsing a Component's Change Log

JIRA's **Change Log** report shows resolved issues in the last ten released versions of a project. Whereas the [Road Map] looks forward, the Change Log looks back, giving an overall view of issues resolved in recent versions.

To browse a component's Change Log,

1. Click 'Browse Project' in the top navigation panel.
2. Select the project you are interested in, if presented with a list.
3. Click the **Components** tab.
4. A list of components will be displayed. Click the name of the component in which you are interested.
5. Click the **Change Log** tab. You will see something like this:



To see the Change Log for all released versions (not just the latest ten), click **'all versions'.**

If your administrator has hidden the '**Fix For Version**' field, the Change Log report will not be available.

See also

- The Road Map — looking forward to next releases
- Browsing a Project
- JIRA Reports Overview

Customising the Dashboard

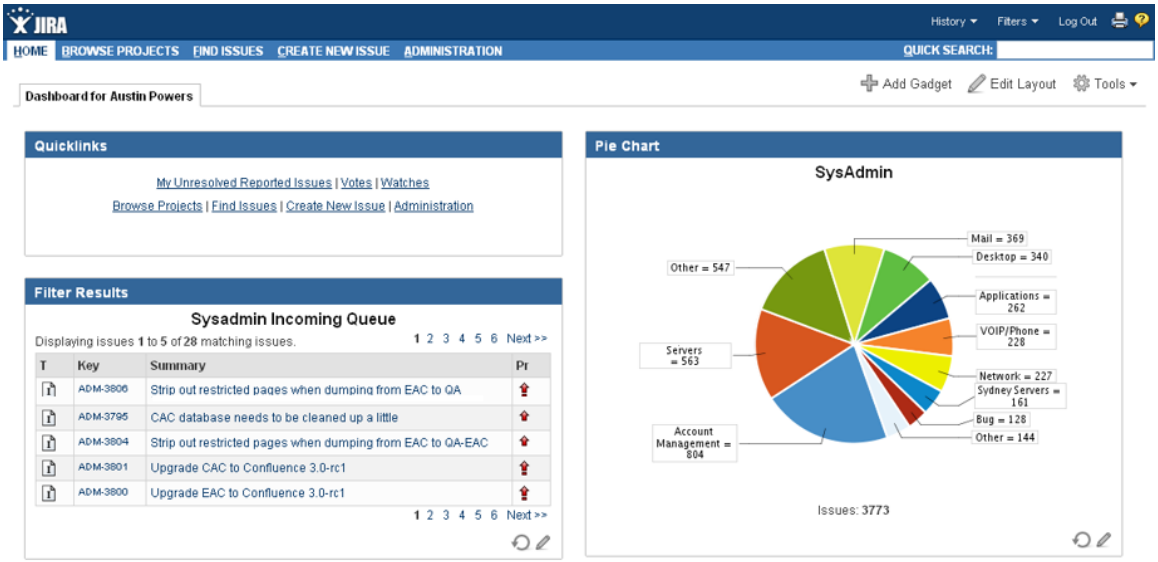
On this page:

- About Dashboards and Gadgets
- Choosing a Dashboard Layout
- Adding a Gadget
- Moving a Gadget
- Removing a Gadget

About Dashboards and Gadgets

The **Dashboard** is the first screen you see when you login to JIRA. It can be configured to display many different types of information, depending on your areas of interest.

The information boxes on the dashboard are called **Gadgets:**



You can easily customise your dashboard by choosing a different layout, adding more gadgets, dragging the gadgets into different positions, and changing the look of individual gadgets.

You can also create more pages for your dashboard, share your pages with other people and choose your favourites pages, as described in Managing Multiple Dashboard Pages . Each page can be configured independently, as per the instructions below.

Available gadgets include:

Gadget	Description
Administration Gadget	The Administration gadget displays quick links to administrative functions conveniently on the dashboard.
Assigned To Me Gadget	The Assigned To Me gadget displays all open issues in all projects assigned to the current user viewing the dashboard.
Average Age Gadget	The Average Age gadget displays a bar chart showing the average number of days that issues have been unresolved.
Bamboo Plan Summary Gadget *	The Bamboo Plan Summary gadget displays a graphical summary of a build plan.
Bamboo Status Gadget *	The Bamboo Status gadget displays a list of all plans on the Bamboo server, and each plan's current status.
Bugzilla ID Search Gadget	The Bugzilla ID Search gadget allows the user to search all JIRA issues for references to Bugzilla IDs.
Calendar Gadget *	The Calendar gadget shows issues and versions in a calendar format based on their due date. Calendars can be based on an issue filter or on a project.
Created vs Resolved Gadget	The Created vs Resolved gadget displays a difference chart showing the issues created vs resolved over a given period.
Favourite Filters Gadget	The Favourite Filters gadget displays a list of all the issue filters that have currently been added by you as a favourite filter.
Filter Statistics Gadget	The Filter Statistics gadget displays the collection of issues returned from a specified filter broken down by a specified field.
2D Filter Statistics Gadget	The Two Dimensional Filter Statistics gadget displays statistical data based on a specified filter in a configurable table format.
In Progress Gadget	The In Progress gadget displays all issues that are currently in progress and assigned to the current user viewing the dashboard.
Introduction Gadget	The Introduction gadget displays a configurable introduction message on the dashboard.

Pie Chart Gadget	The Pie Chart gadget displays displays issues from a project or issue filter, grouped by a statistic type, in pie-chart format. The issues can be grouped by any statistic type (e.g. Status, Priority, Assignee, etc).
Project Gadget	The Project gadget provides information and various filters related to a specified project on the dashboard.
Projects Gadget	The Projects gadget provides information and various filters related to all projects within JIRA.
Project Statistics Gadget	The Project Statistics gadget allows various per-project statistical data to be displayed on the dashboard.
Project Table Gadget	The Project Table gadget displays all the project names in a table in the dashboard.
Quick Links Gadget	The Quick Links gadget displays a number of useful links to issues associated with the current user. Each link directs the user to the Issue Navigator, displaying the relevant issues such as Reported Issues, Voted Issues and Watched Issues.
Recently Created Issues Gadget	The Recently Created Issues gadget displays a bar chart showing the rate at which issues are being created, as well as how many of those created issues are resolved.
Resolution Time Gadget	The Resolution Time gadget displays a bar chart showing the average resolution time (in days) of resolved issues.
Road Map Gadget	The Road Map gadget shows versions which are due for release within a specified period of time, and a summary of progress made towards completing the issues in those versions.
Saved Filter Gadget	The Saved Filter gadget displays the results of a specified issue filter on the dashboard.
Text Gadget	The Text gadget displays a configurable HTML text on the dashboard.
Time Since Issues Gadget	The Time Since Issues gadget displays a bar chart showing the number of issues for which your chosen date field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a custom field) was set on a given date.
Voted Gadget	The Voted Issues gadget shows issues for which you have voted.
Watched Gadget	The Watched Issues gadget shows issues which you are watching.

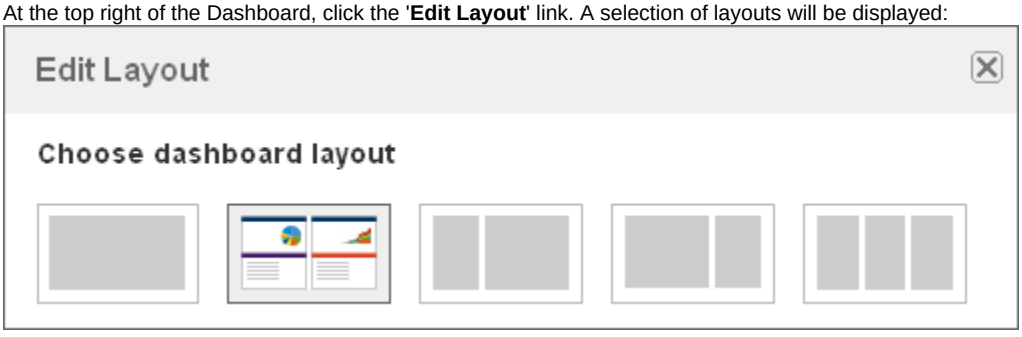
* *This gadget will only be available if it has been installed by your JIRA administrator.*

You can return to the dashboard from anywhere in JIRA by clicking the **'Home'** link on the top navigation bar.

Choosing a Dashboard Layout

To choose a different layout for your dashboard page (e.g. three columns instead of two):

1. At the top right of the Dashboard, click the '**Edit Layout**' link. A selection of layouts will be displayed:

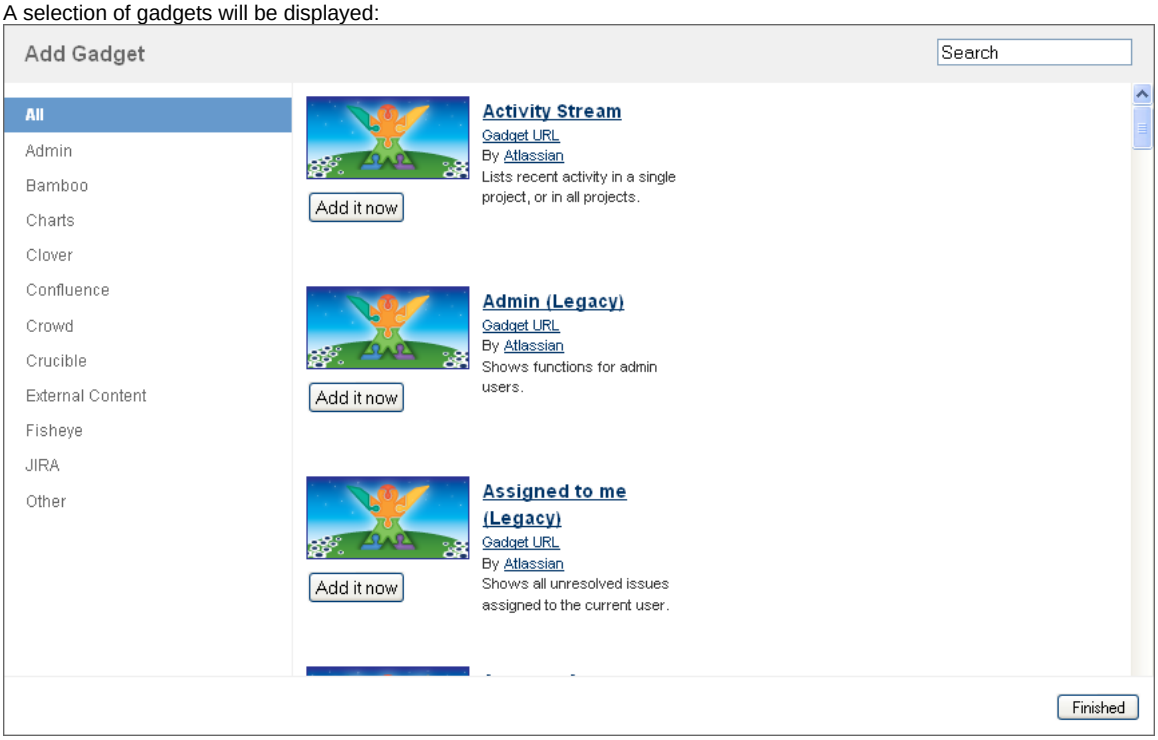


2. Click your preferred layout.

- The first time that you configure your dashboard, a new dashboard is created for you (based on the default dashboard) and added as your "favourite", with "private" sharing .
- If you are using multiple dashboard pages, you can only configure dashboard pages that you own. The default "System Dashboard" can only be configured by your JIRA administrator.

Adding a Gadget

1. At the top right of the Dashboard, click the '**Add Gadget**' link.
2. A selection of gadgets will be displayed:



3. Click the '**Add it now**' button beneath your chosen gadget.
4. Click the '**Finished**' button to return to your Dashboard.
5. If the gadget you have selected requires configuration, you will be presented with the gadget's configuration page. Configure appropriately and click '**Save**'.

Moving a Gadget

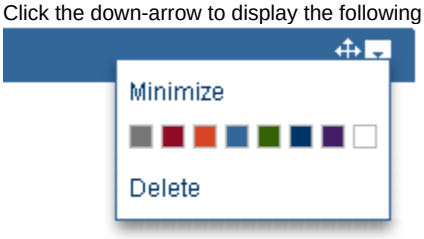
To move a gadget to a different position on your dashboard:

1. Click the gadget and drag it into its new position.

Removing a Gadget

To remove a gadget from your dashboard:

1. Hold your mouse over the top right corner of the gadget, until a down-arrow appears.
2. Click the down-arrow to display the following menu:



3. Click '**Delete**'.

Adding the Activity Stream Gadget

The **Activity Stream** gadget displays a summary of recent activity in JIRA projects (and/or by particular people) in which you are interested.

The '**Activity Stream**' gadget also provides an RSS feed, allowing you to create very specific RSS feeds of only the information that is most relevant to you. Simply add the '**Activity Stream**' gadget to your dashboard, specify the people/projects of interest (see instructions below), then

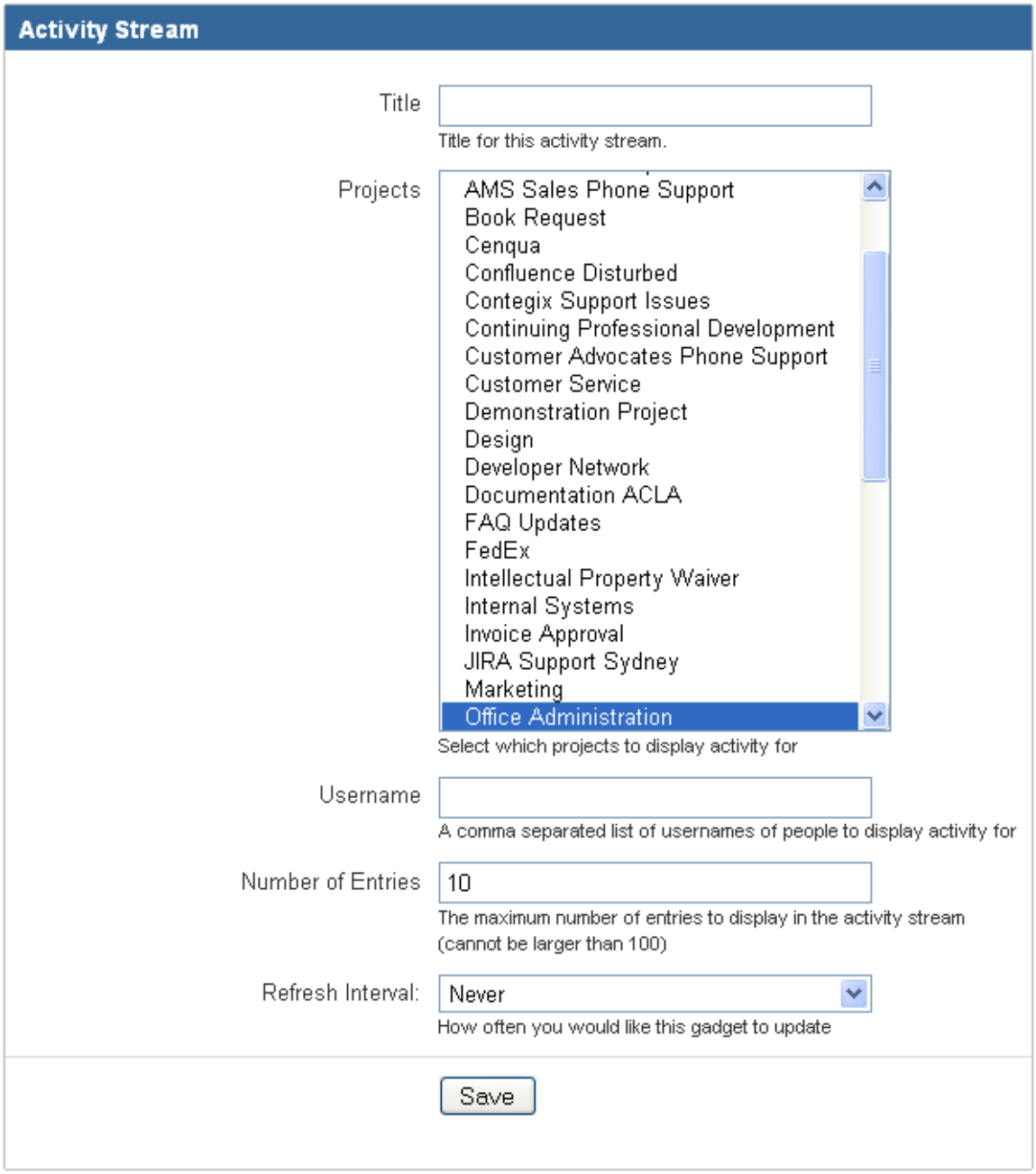
click the RSS icon: .

What does it look like?

The **'Activity Stream'** gadget should appear as follows on the dashboard:

Adding the 'Activity Stream' gadget to your Dashboard

1. Go to your JIRA dashboard and click '**Add Gadget**'.
2. Select the **'Activity Stream'** gadget.
3. The Activity Stream gadget will appear on your dashboard as follows, ready for you to configure:



'**Title**' — type a heading for this gadget.

a. '**Projects**' — select one or more projects (or 'All Projects') for which to display activity.
b. '**Username**' — type the usernames of one or more people whose activity you wish to display (or leave blank to display everyone's activity).
c. '**Number of Entries**' — type the number of activities that you want the gadget to display.
d. '**Refresh Interval**' — select how often you want the gadget to update the displayed activity (never / every 15 minutes / every 30 minutes / every hour / every two hours).

4. Click the **'Save'** button.

Changing the Look and Behaviour of a Gadget

On this page:

- Hiding or Changing the Colour of the Gadget's Frame
- Minimising and Expanding a Gadget
- Opening the Maximised or Canvas View of a Gadget
- Editing a Gadget's Settings

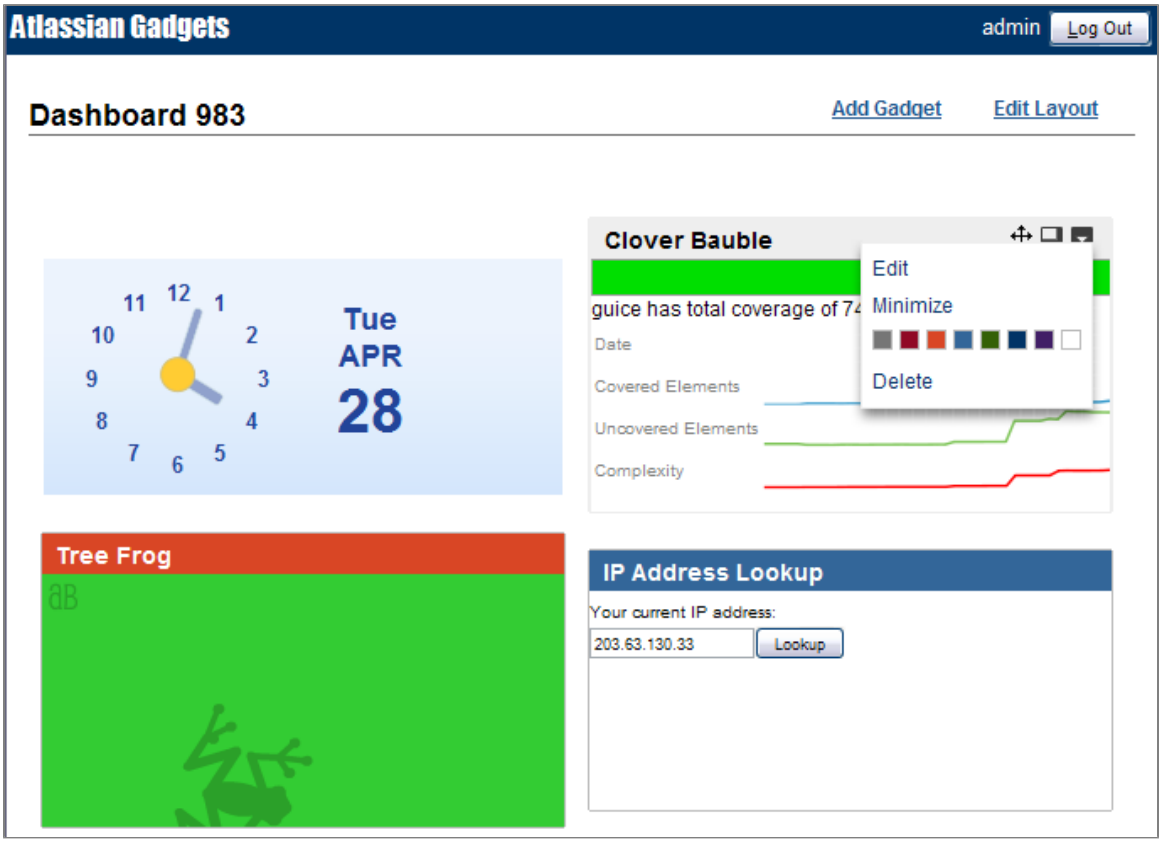
Hiding or Changing the Colour of the Gadget's Frame

You can change the colour of the frame surrounding a gadget on your dashboard. You can even hide the gadget's frame altogether, so that it only shows when you move your mouse pointer over the gadget. In the screenshot below, the top two gadgets have hidden frames. The frame for the top gadget on the left is not visible. The frame for the top gadget on the right is currently visible because the mouse pointer is hovering over the gadget.

To hide or change the colour of a gadget's frame,

1. Go to the dashboard by clicking the '**Dashboard**' link or the '**Home**' link at the top left of the screen.
2. The dashboard will appear, looking something like the screenshot below. Move your mouse pointer over the gadget you want to change. If the gadget's frame is hidden, the frame will appear now.
3. Click the dropdown menu icon at top right of the gadget frame.
4. The dropdown menu will appear, as shown in the screenshot below. Click the colour you want for your gadget's frame. To hide the gadget's frame, select the empty (or white) colour box.

Screenshot: Hiding or changing the colour of a gadget's frame



Minimising and Expanding a Gadget

You can shrink (minimise) a gadget on your dashboard so that it displays only the top bar of the gadget frame. In the screenshot below, the top left gadget ('**Date & Time**') has been minimised.

- If you minimise a gadget that has a hidden frame, the gadget will not be visible on the dashboard until you move your mouse pointer over the gadget. See the section above on hiding or changing the colour of the gadget frame.
- You can minimise/expand a gadget even if you do not have update permissions on the dashboard.
- The minimise/expand setting is stored in a cookie, and is not saved to the dashboard server.

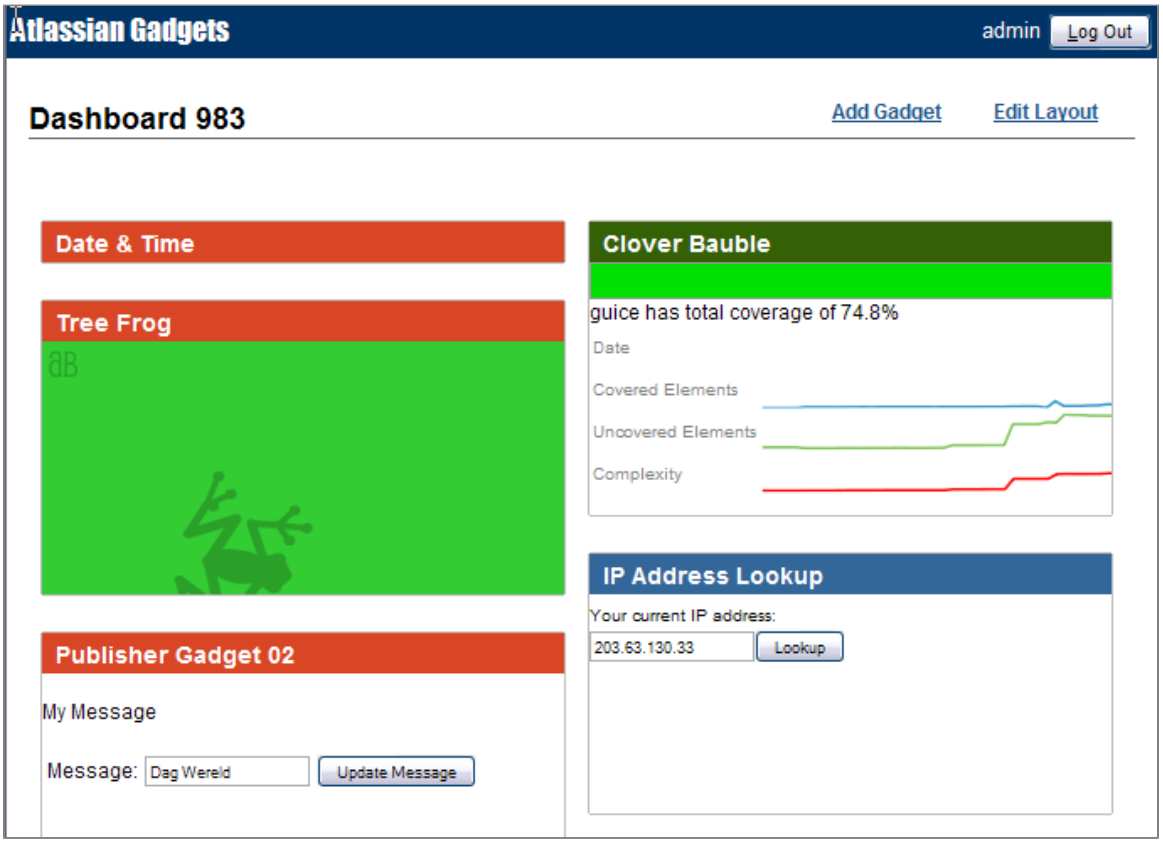
To minimise a gadget,

1. Move your mouse pointer over the gadget you want to change.
2. The gadget menu icons will appear. Click the dropdown menu icon at top right of the gadget frame.
3. The dropdown menu will appear, as shown in the screenshot above. Click '**Minimise**'.

To expand a gadget that has been minimised,

1. Move your mouse pointer over the gadget you want to change.
2. The gadget menu icons will appear. Click the dropdown menu icon at top right of the gadget frame.
3. The dropdown menu will appear. Click '**Expand**'.

Screenshot: A minimised gadget



Opening the Maximised or Canvas View of a Gadget

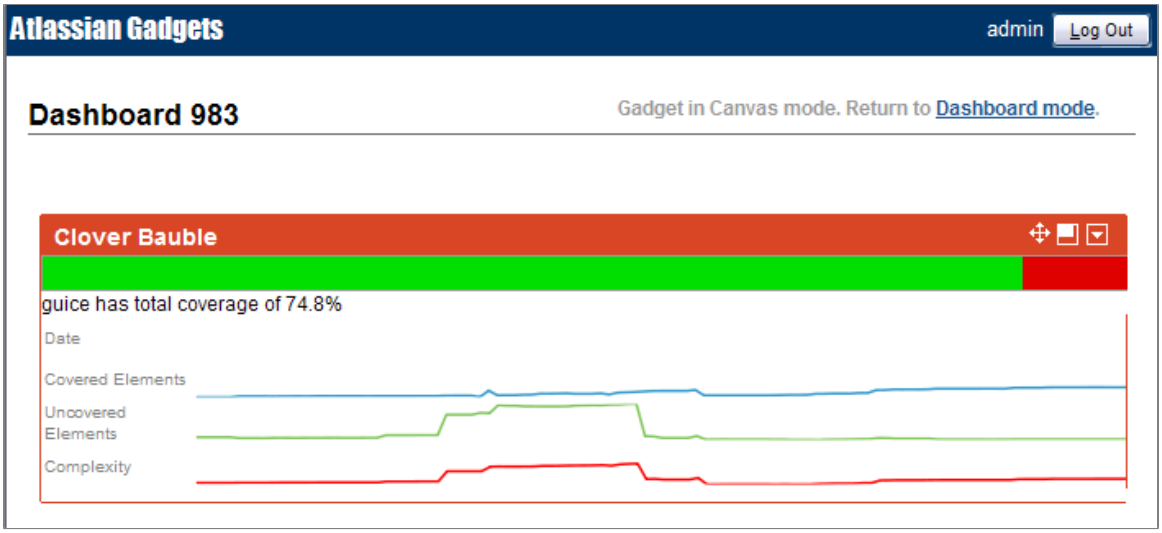
Some gadgets allow you to expand themselves so that they take up the entire space allowed by the dashboard. This is also known as 'canvas view'.

- The maximised or canvas view of a gadget often provides additional functionality, i.e. more than is available in the standard view of the gadget.
- This is not the same as minimising and then expanding a gadget (see above).
- Only some gadgets provide the maximised or canvas view.
- You can open the canvas view of a gadget even if you do not have update permissions on the dashboard.
- The maximised/canvas view setting is stored in a cookie, and is not saved to the dashboard server.

To open the maximised or canvas view of a gadget,

1. Move your mouse pointer over the gadget you want to change.
2. The gadget menu icons will appear. Click the maximise icon at top right of the gadget frame. This icon will appear only if the gadget provides a maximised or canvas view.
3. The gadget's maximised view will open, as shown in the screenshot below.

Screenshot: The maximised or canvas view of a gadget



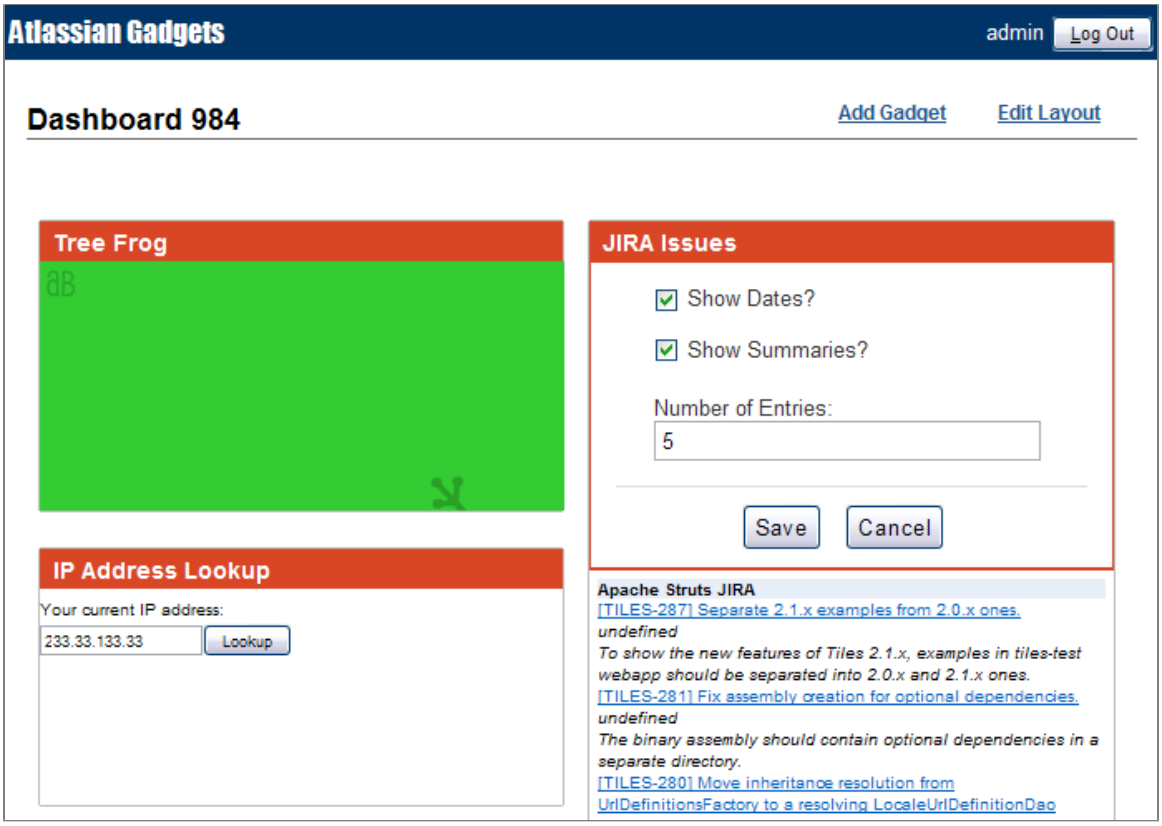
Editing a Gadget's Settings

Some gadgets provide specific properties or settings that you can edit. These settings will be different for each gadget. For example, a gadget may allow you to customise its welcome message, or to define the server where the gadget will find its information.

To edit a gadget's settings,

1. Move your mouse pointer over the gadget you want to change.
2. The gadget menu icons will appear. Click the dropdown menu icon at top right of the gadget frame.
3. The dropdown menu will appear. Click '**Edit**'.
4. A panel will open, showing the settings offered by the selected gadget. In the screenshot below, the top gadget on the right has its settings panel open.
5. Adjust the settings as required then click '**Save**'.

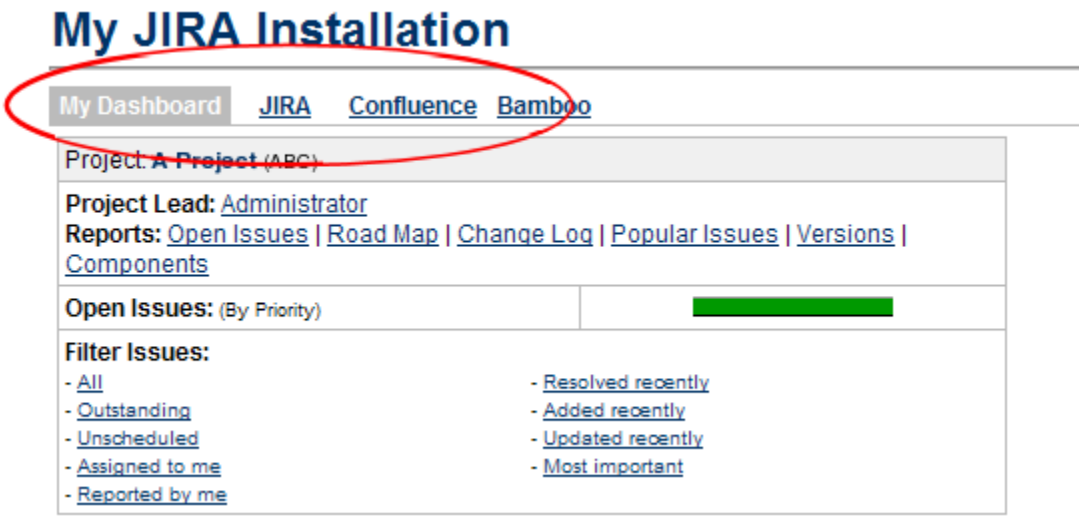
Screenshot: Editing a gadget's settings



Managing Multiple Dashboard Pages

The Professional and Enterprise editions of JIRA allow you to configure more than one dashboard page. Each dashboard page can be configured independently, allowing you to neatly organise related information by context. You can also share your dashboard pages with other users, as well as adding dashboards shared by other users as favourites.

Each dashboard page appears as a separate "tab" on the dashboard. You can view a dashboard page by simply clicking its name in the tab list.



On this page:

- Managing your Dashboard
- Creating new Dashboard Pages
- Displaying a Dashboard Page on your Dashboard ('Favourite Dashboards')
- Sharing Dashboard Pages
- Finding an existing Dashboard Page

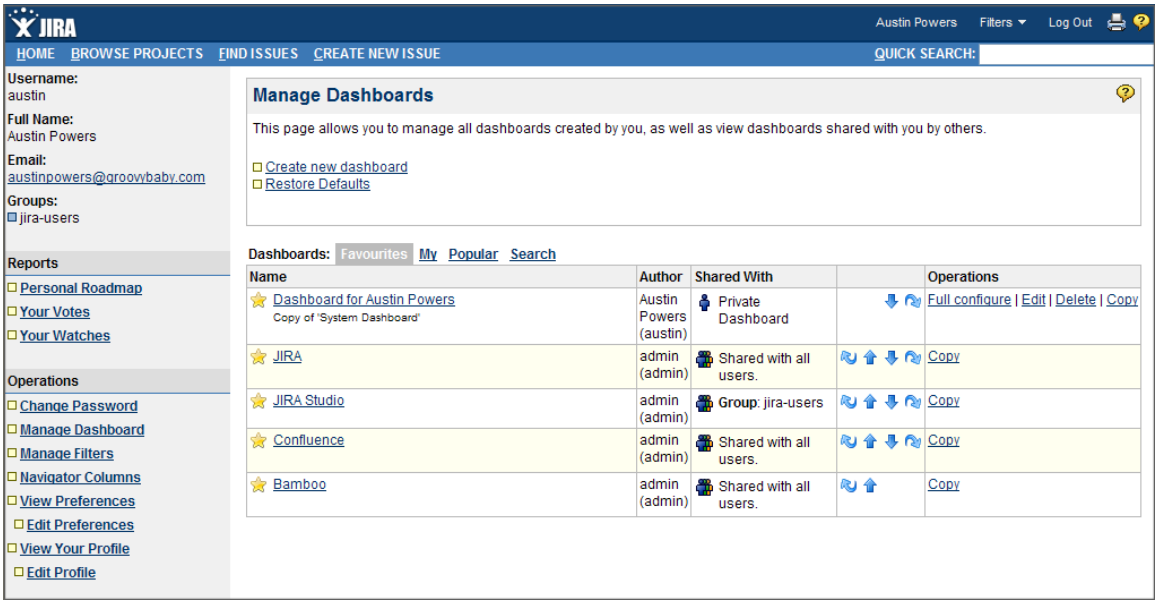
- Editing an existing Dashboard Page's details
- Copying an existing Dashboard Page
- Deleting an existing Dashboard Page

Managing your Dashboard

The Manage Dashboards page allows you to view and configure dashboard pages that you have created, as well as work with dashboard pages that other users have shared with you.

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. From this page, you can:
    - Create a new dashboard page.
    - Add a dashboard page as a favourite.
    - Share a dashboard page that you have created, with other users.
    - Search for dashboard pages that has been created by you or shared with you by other users.
    - Configure an existing dashboard or edit an existing dashboard's details of a dashboard that you have created.
    - Copy a dashboard page that has been created by you or shared with you by other users.
    - Delete a dashboard page that you have created.

Click the above links for further details on each function.



You can also re-order your dashboard pages on this page, by using the arrow icons:

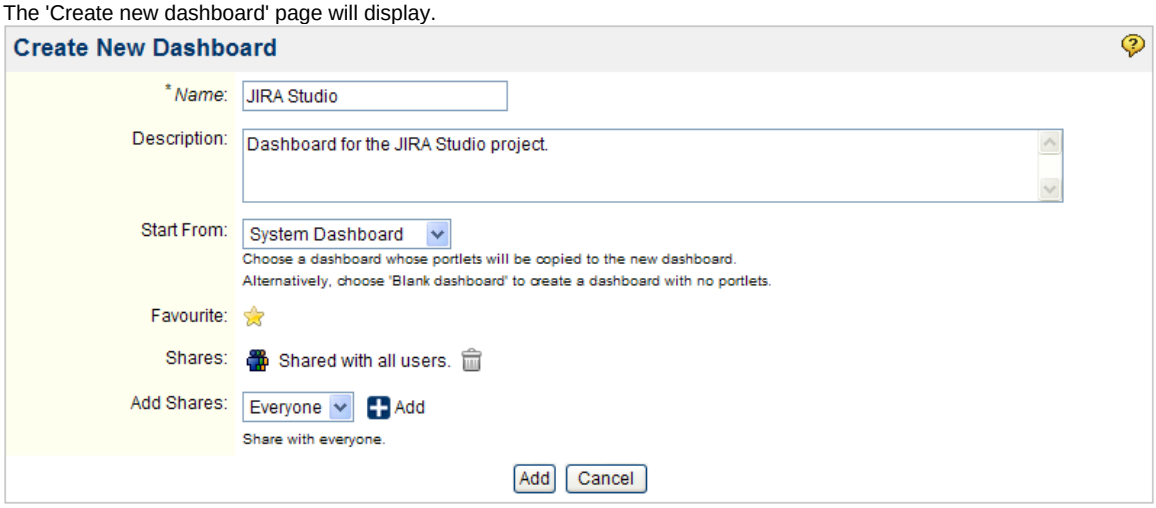
Move a dashboard up (promote tab to the left, on the dashboard)	Click the up arrow ( ) for the dashboard that you wish to move.
Move a dashboard down (demote tab to the right, on the dashboard)	Click the down arrow ( ) for the dashboard that you wish to move.
Move a gadget to the top (promote to leftmost tab on the dashboard)	Click the curly up arrow ( ) for the dashboard that you wish to move.
Move a gadget to the bottom of the column (demote to rightmost tab on the dashboard)	Click the curly down arrow ( ) for the dashboard that you wish to move.

Creating new Dashboard Pages

To create a new dashboard page please follow these steps:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. This page will list all currently configured dashboard pages. Click the '**Create new dashboard**' link.

3. The 'Create new dashboard' page will display.



   Provide a name for the new dashboard page and optionally enter a short description. You can also choose an existing page as a starting point for the new page. This means that the configuration of the existing page will be duplicated for the newly created page. Alternatively, if you would like to create a page with no gadgets, leave the 'Blank dashboard' option selected.
4. Your new dashboard page will be added as a 'favourite' dashboard page by default upon creation, which means that it will display as a tab on your JIRA dashboard. If you do not wish to display this dashboard page as a tab on your JIRA dashboard, deselect the star icon. You can add the dashboard page as a favourite after it has been created. Read more about adding an existing dashboard page as a favourite.
5. The sharing of your new dashboard page depends on your sharing preference in your user profile. If you have not specified a personal preference, then the global default for sharing will apply (i.e. 'Private', unless changed by your JIRA Administrator under 'User Defaults' in the Administration menu). If you wish to change the sharing of your dashboard page, refer to the instructions on sharing dashboard pages below.

   > Please note, you need the 'Create Shared Object' global permission to be able to share your dashboard page. If you cannot see any dashboard sharing functionality, contact your JIRA Administrator to be granted this permission.

6. Click the '**Add**' button. Your new page will be listed under the 'My' tab of the 'Manage Dashboards' page. If you selected the new dashboard page as a favourite, it will also appear under the 'Favourite' tab and will be displayed as a tab on your JIRA dashboard.
7. If you need to configure the gadgets on your new dashboard page, click the '**Full configure**' link in the 'Operations' column to proceed to the dashboard configuration interface.

Displaying a Dashboard Page on your Dashboard ('Favourite Dashboards')

Dashboard pages that you have created, or that have been shared by other people, can be added as a 'favourite'. This means that the dashboard page will appear as a tab on your JIRA dashboard. There is no system restriction on the number of dashboard pages that you can display on your JIRA dashboard.

> Please note, if you have added another user's shared dashboard as a favourite and a gadget(s) is not displaying correctly, the gadget(s) may be using an issue filter that is not shared with you. You will need to contact the author of the issue filter to change the filter sharing.

To add an existing dashboard page to the dashboard please follow these steps:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to display on your dashboard. If you created the dashboard, it will be listed under the 'My' tab, otherwise you can search for dashboards shared by other users via the 'Search' tab.
3. Click the star icon next to the name of the desired dashboard page to add it as a favourite. The dashboard page will be displayed on your main dashboard.

> Please note, if you do not have any dashboard pages added as favourites, the default dashboard will be displayed on your dashboard with an error message. You can choose to keep the default dashboard displayed on your dashboard, but you will need to add it as a favourite to stop the error message from showing. You may need to search for the 'System Default' dashboard to add it as a favourite.

Dashboards: Favourite | My | Popular | Search

Name	Shared With	Operations
Bamboo Bamboo issues	Shared with all users.	Full configure | Edit | Delete | Copy
Clover Clover Issues	Shared with all users.	Full configure | Edit | Delete | Copy
Confluence Confluence Issues	Shared with all users.	Full configure | Edit | Delete | Copy

To remove a dashboard page from your dashboard please follow these steps:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to remove from your dashboard under the 'Favourites' tab.
3. Click the star icon next to the name of the dashboard page. The dashboard page will be removed from your main dashboard.

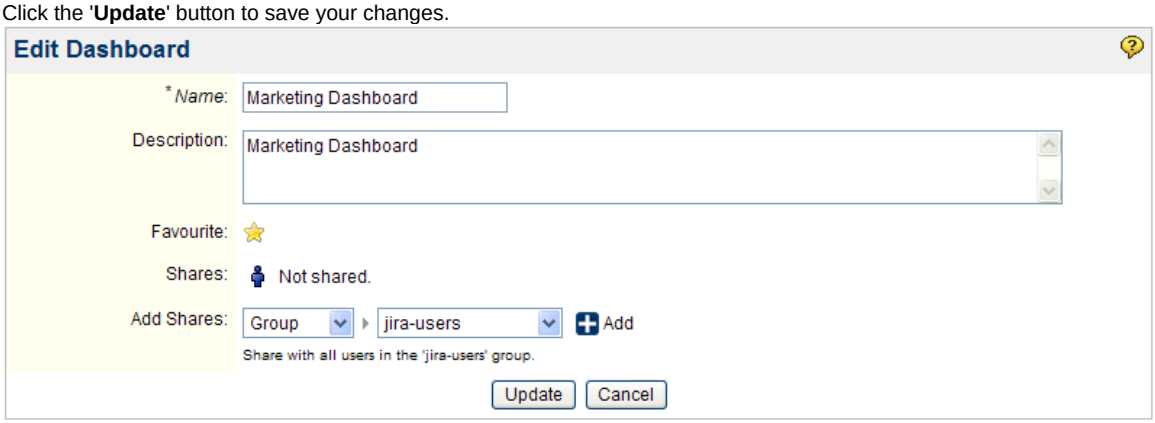
Sharing Dashboard Pages

The Professional and Enterprise editions of JIRA also allow you to share any dashboard pages that you have configured. Dashboard pages can be shared with other users via user groups, projects and project roles. Dashboard pages can also be shared globally. Sharing a dashboard page allows other users to display it on their JIRA dashboard, by selecting it as a favourite.

> Please note, you may need to review the sharing permissions for any issue filters used in portlets on your shared dashboard. If another user adds your dashboard as a favourite, but cannot access a filter for a portlet, then the portlet will display with an error message.

To share an existing dashboard page to the dashboard please follow these steps:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to display on your dashboard under the 'My' tab and click the '**Edit**' link for the dashboard in the '**Operations**' column.
3. The 'Edit Dashboard' page will display. Select the group, project or project role that you want to share the dashboard with, or share it with all users, if you wish. Click the '**Add**' link to add the share. You can add further share permissions if you wish.
4. Click the '**Update**' button to save your changes.

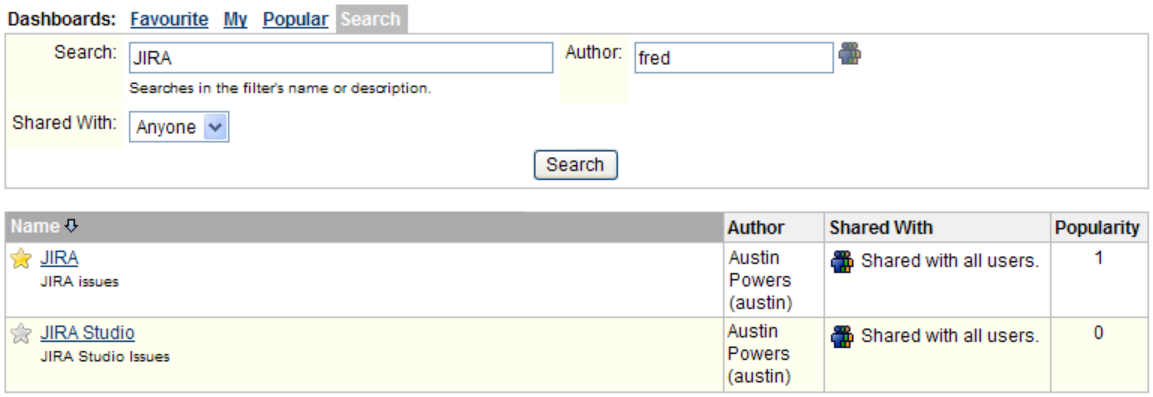


Finding an existing Dashboard Page

Dashboard pages that you have created or have been shared by other users, can be found via the dashboard Search function of the 'Manage Dashboards' page. If it is a popular dashboard (i.e. added as a favourite by many users), you can also locate it on the 'Popular' tab of the 'Manage Dashboards' page which lists the top twenty most popular dashboards.

Follow the steps below to search for an existing dashboard page:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. Click the '**Search**' tab. The dashboard Search will display. Enter your search criteria and click '**Search**' to run the search.
3. Your search results will be displayed on the same page. You can sort the search results by any of the columns, by clicking the column headers. Click the name of any dashboard page to temporarily display it on your dashboard (i.e. it will be removed from your dashboard when you navigate away). To keep the dashboard page as a tab on your dashboard, click the '**add it as a favourite**' link.

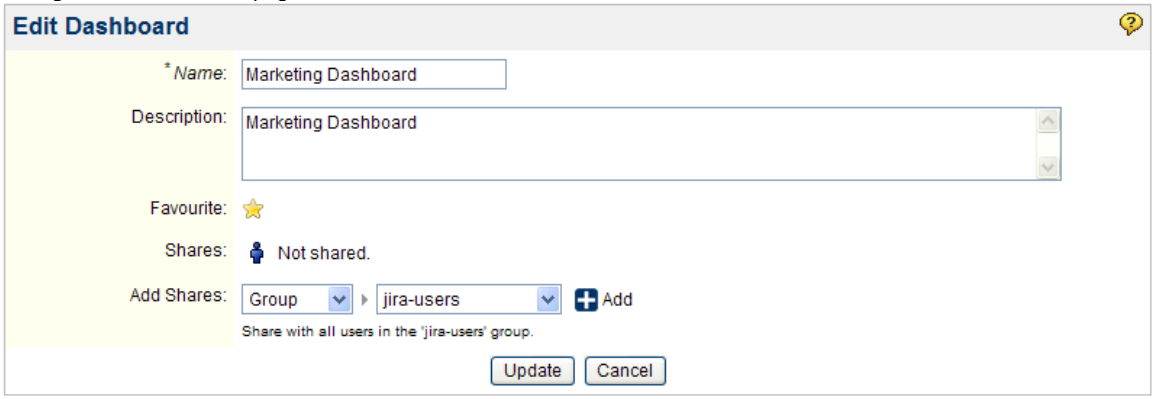


Editing an existing Dashboard Page's details

You can always update the details, i.e. Name, Description, Sharing, Favourite, of an existing dashboard page after its creation. Please note that you can only update the details of dashboard pages which you have created.

Follow the steps below to update the details of one of your existing dashboard pages:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to update and click the '**Edit**' link for the dashboard in the 'Operations' column.
3. The 'Edit Dashboard' page will display. Update the details of the dashboard page as desird. If you wish to change the sharing or favourite settings for the dashboard page, refer to the relevant instructions above.



4. Click the '**Update**' button to save your changes.

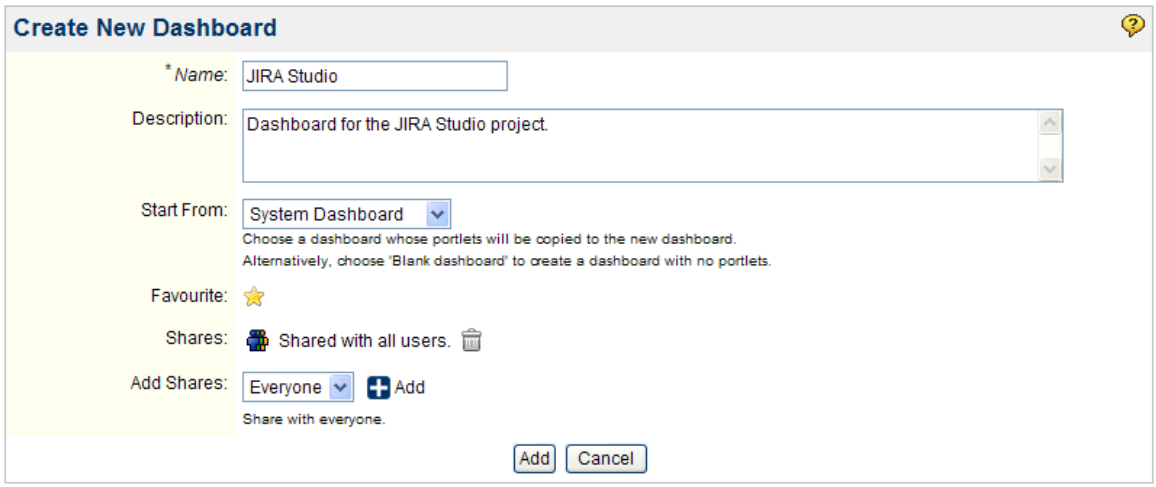
> The ability to share and add a dashboard page as a favourite is only available for the Professional and Enterprise editions of JIRA.

Copying an existing Dashboard Page

You can make a copy of an existing dashboard page (created by you or shared with you), which creates a new dashboard page with the same gadget configuration as the existing dashboard page.

Follow the steps below to update the details of one of your existing dashboard pages:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to copy and click the '**Copy**' link for the dashboard in the 'Operations' column.
3. The 'Create New Dashboard' page will display. Update the details of the dashboard page as desired. If you wish to change the sharing or favourite settings for the dashboard page, refer to the relevant instructions above.



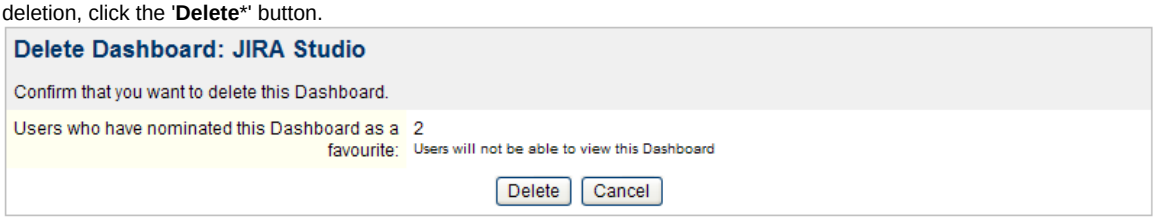
4. Click the '**Update**' button to save your changes.

The ability to share and add a filter as a favourite is only available for the Professional and Enterprise editions of JIRA.

Deleting an existing Dashboard Page

Follow the steps below to delete a dashboard pages. Please note that you can only delete dashboard pages that you created:

1. On the top right of the Dashboard, click on the link labelled '**Manage Dashboard**'.
2. The 'Manage Dashboards' page will display. Locate the dashboard page that you wish to copy and click the '**Delete**' link for the dashboard in the 'Operations' column.
3. The confirmation page will inform you of how many other users have this page selected as a favourite. If you wish to continue with the deletion, click the '**Delete***' button.

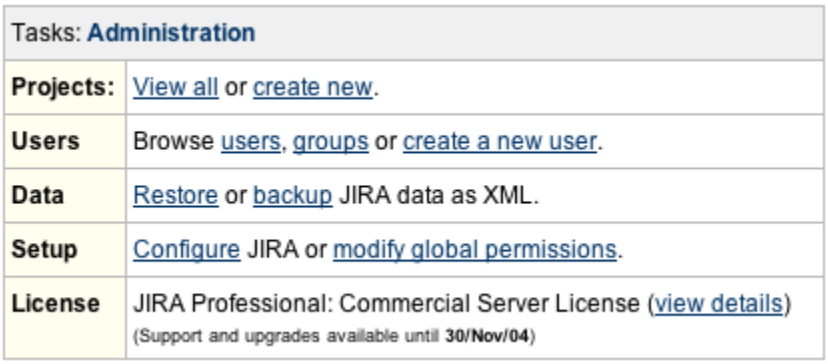


Adding the Administration Gadget

The **Administration** gadget displays quick links to administrative functions conveniently on the dashboard. This gadget is for people who have the **'JIRA Administrators'** or the **'JIRA System Administrators'** global permission.

What does it look like?

The **'Administration'** gadget should appear as follows on the dashboard:



Note: if you only have the **'JIRA Administrators'** permission (and not the **'JIRA System Administrators'** permission), the following words will not be links: 'Restore', 'Backup' and 'License: view details'.

Adding the 'Administration' gadget to your Dashboard

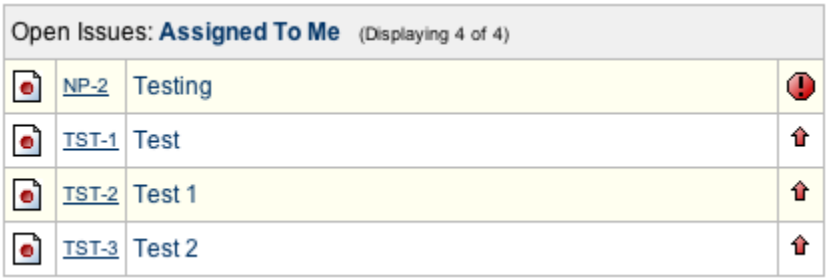
1. Navigate to the dashboard configuration page.
2. Select the **'Administration'** gadget.
3. Click **'Add'.**
4. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Assigned To Me Gadget

The **Assigned To Me** gadget displays all open issues in all projects assigned to the current user viewing the dashboard.

What does it look like?

The **Assigned to Me** portlet should appear as follows on the dashboard:



Adding the 'Assigned To Me' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Assigned To Me' gadget.
3. Click Add.
4. Select the maximum number of issues to display in the gadget.

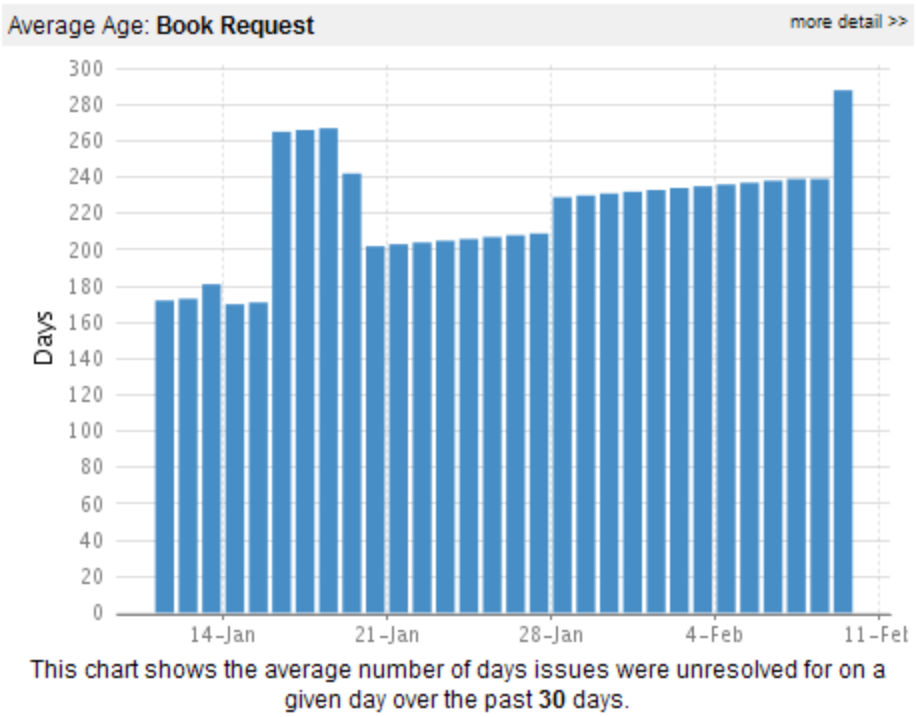
5. Click 'Save'.
6. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Average Age Gadget

The **'Average Age'** gadget displays a bar chart showing the average age (in days) of unresolved issues at given points in time. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does it look like?

The **'Average Age'** gadget will appear as follows on the dashboard:

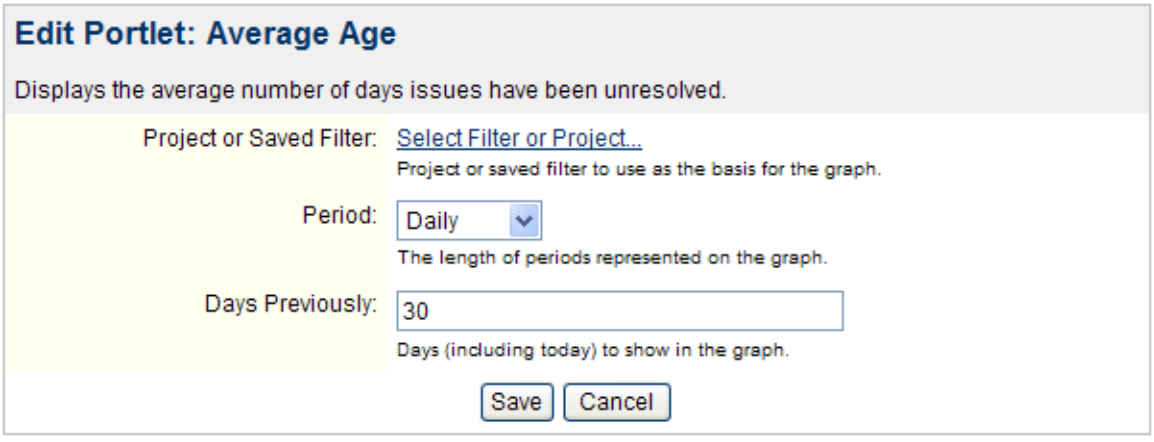


Click the **'more detail'** link to go to the full-size report and data table.

Adding the 'Average Age' gadget to your Dashboard

To add the **'Average Age'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Average Age'** gadget.
3. Click the **Add** button.
4. The **'Edit Gadget'** screen will be displayed:

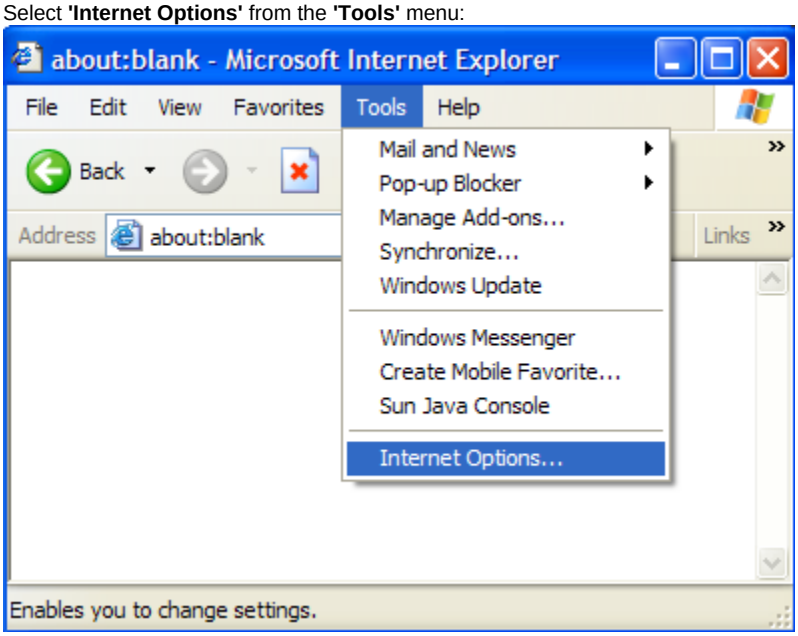


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Period'** field, select the timeframe on which the chart will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
8. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the chart.
9. Click the **'Save'** button.
10. The gadget can be positioned on the dashboard through the dashboard configuration page.

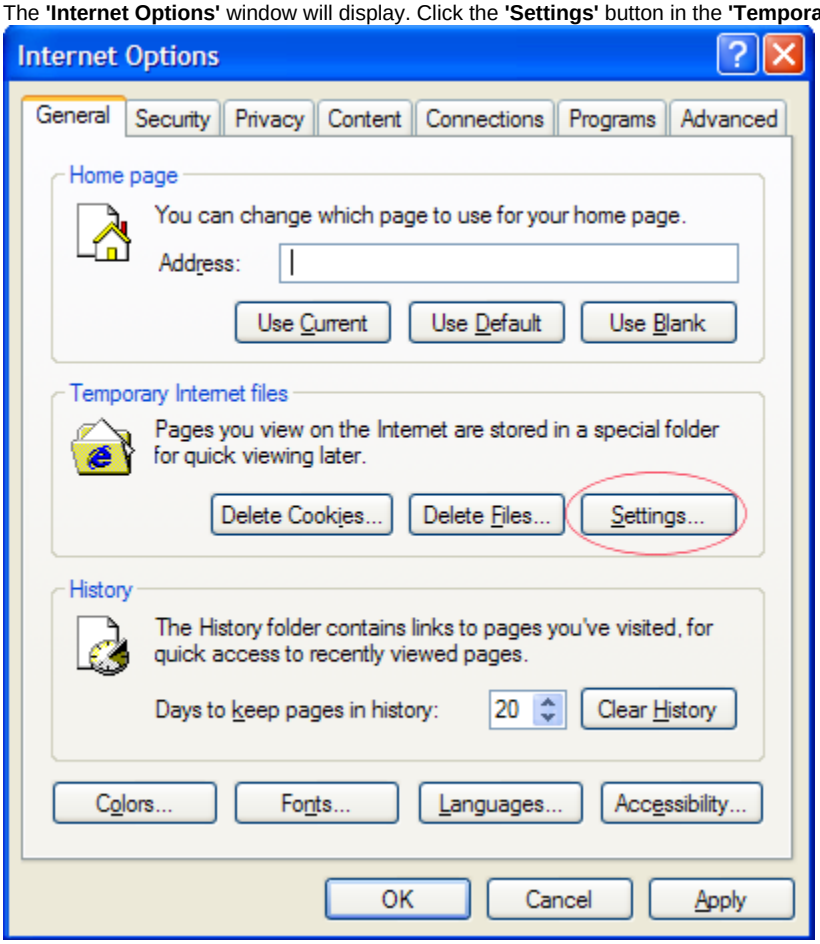
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

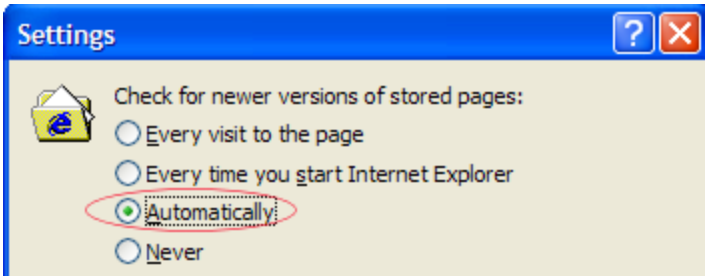
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Adding the Bamboo Plan Summary Gadget

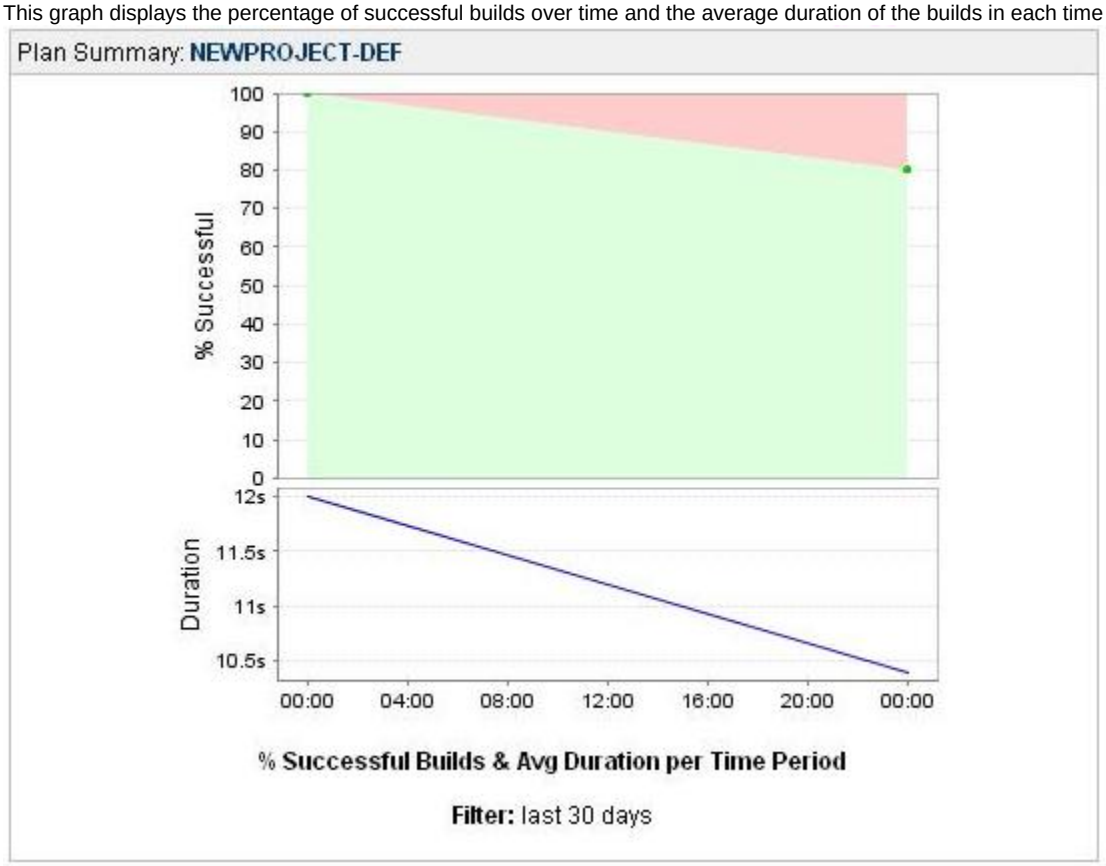
The **Bamboo Plan Summary** gadget displays a graphical summary of a Bamboo build plan.

What does it look like?

There are two graph types available with the **Bamboo Status** gadget:

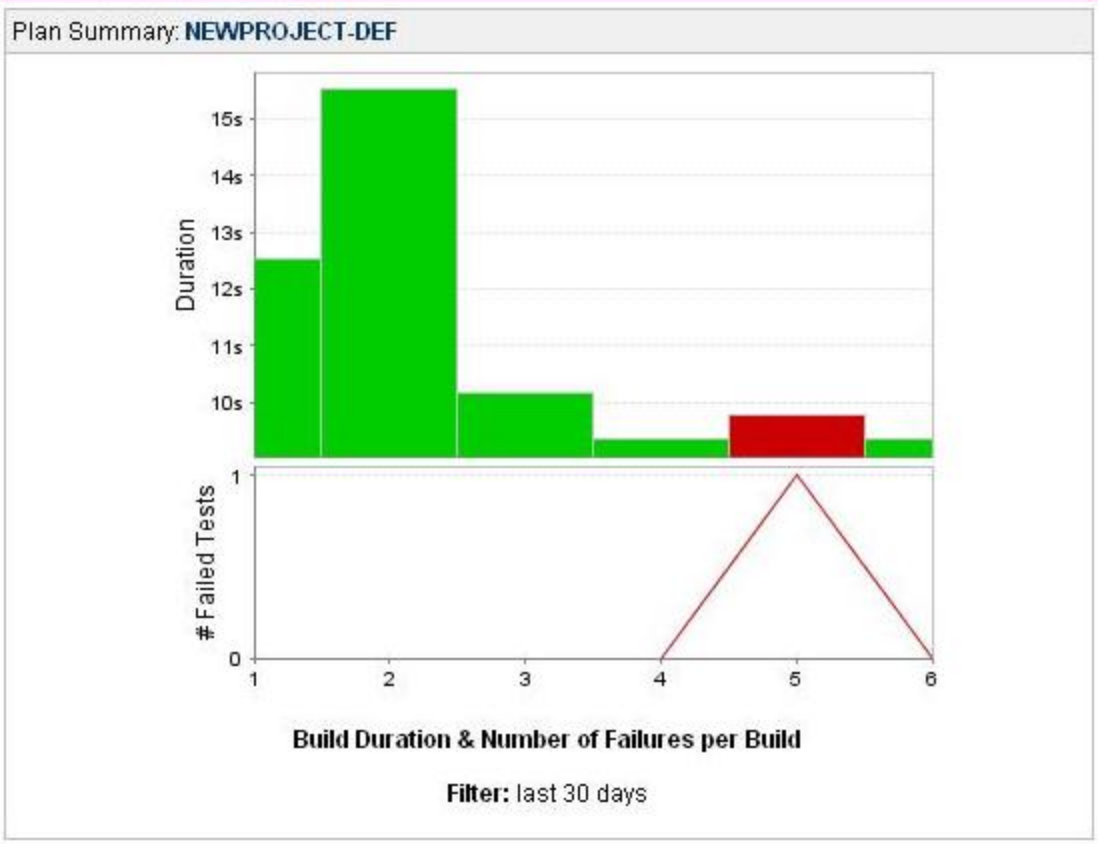
1. Group By Time Period

This graph displays the percentage of successful builds over time and the average duration of the builds in each time period:



2. Group By Build

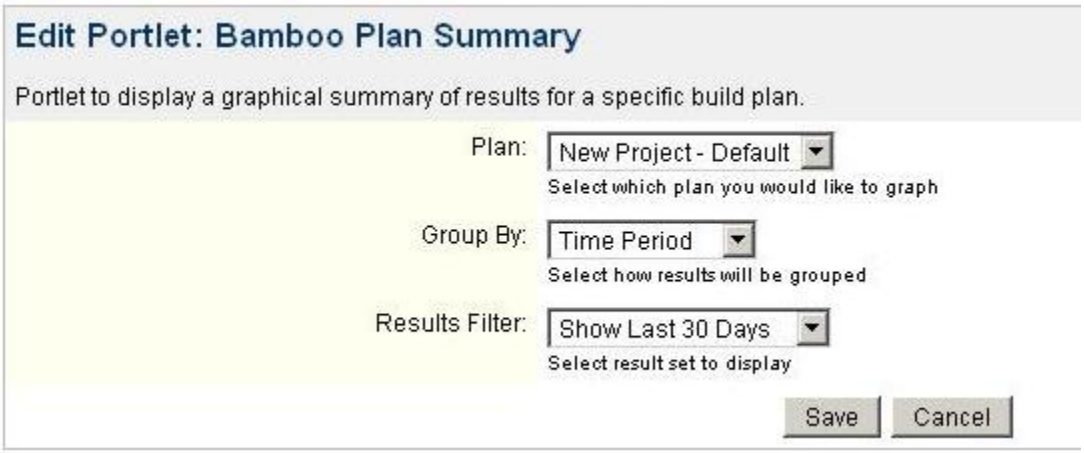
This graph displays the duration of each of the builds and the number of failed tests per build.:



The Bamboo Plan Summary gadget will only be available to add to your dashboard if your JIRA administrator has installed the Bamboo plugin for JIRA.

Adding the 'Bamboo Plan Summary' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Bamboo Plan Summary' gadget.
3. Click Add.
4. The 'Edit Gadget' screen will be displayed:

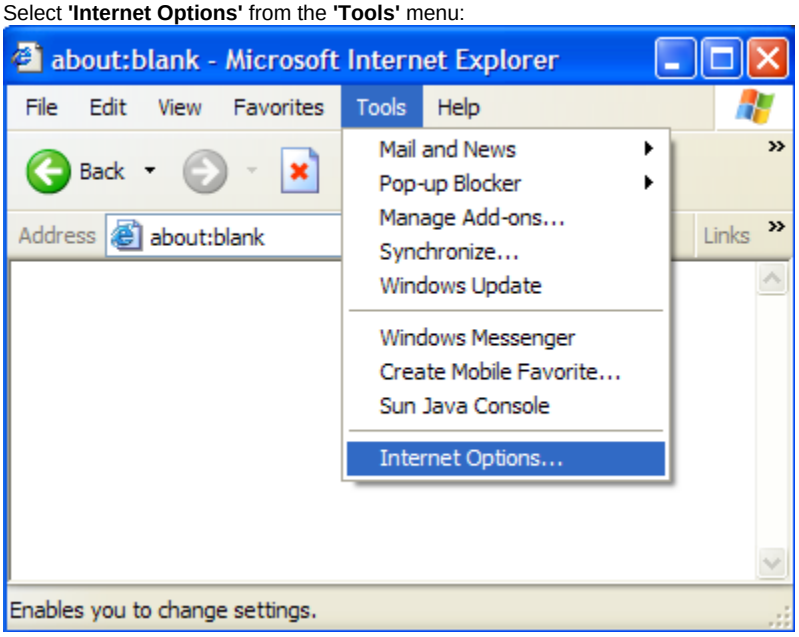


5. You have several options available when configuring this gadget:
   - **Plan** --- Select the Bamboo plan for which you would like to show a summary.
   - **Group By** --- Select how you would like the graph to be displayed.
   - **Results Filter** --- Select how many days worth of builds you would like to include.
6. The gadget can be positioned on the dashboard through the dashboard configuration page.

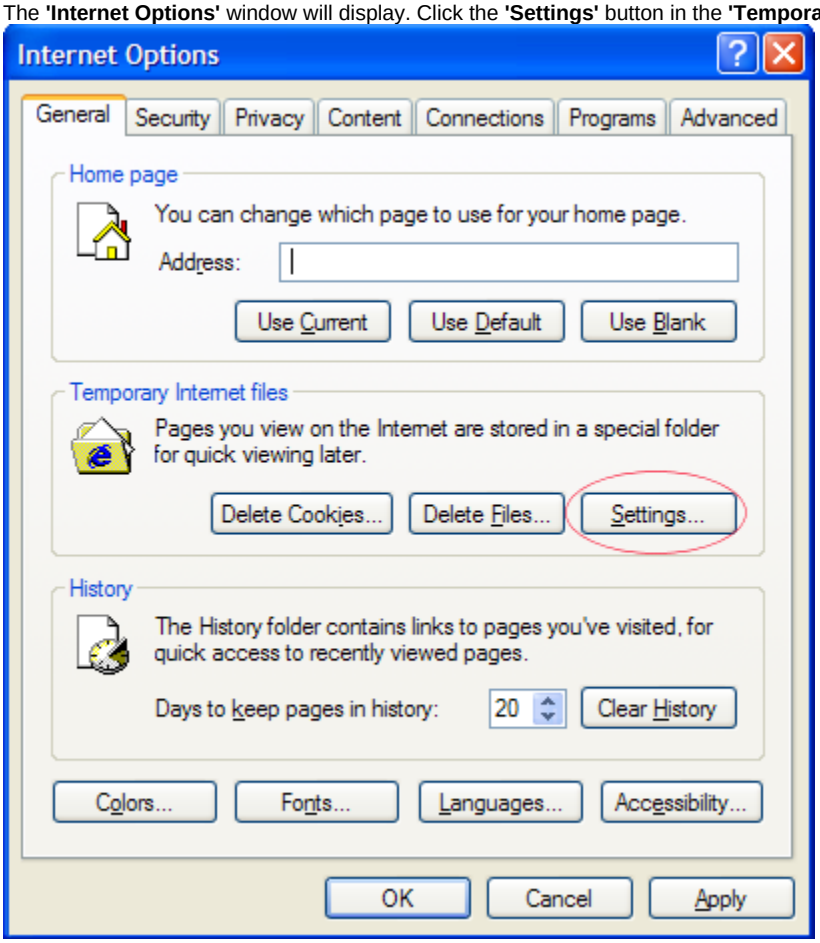
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

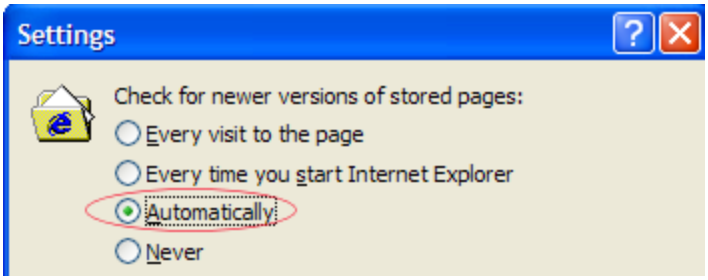
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Adding the Bamboo Status Gadget

The **Bamboo Status** portlet displays a list of all plans on the Bamboo server, and each plan's current status.

What does it look like?

The **Bamboo Status** gadget should appear as follows on the dashboard:

> The Bamboo Status gadget will only be available to add to your dashboard if your JIRA administrator has installed the Bamboo plugin for JIRA.

Adding the 'Bamboo Status' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Bamboo Status' gadget.
3. Click 'Add'.
4. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Bugzilla ID Search Gadget

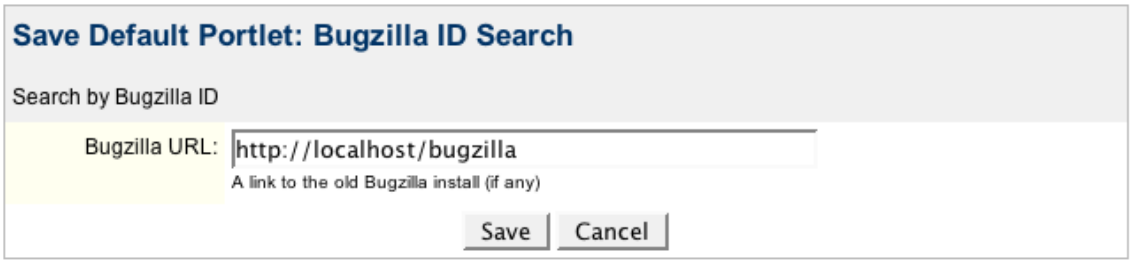
The **Bugzilla ID Search** gadget allows the user to search all JIRA issues for references to Bugzilla IDs. If the specified ID is not found within JIRA, the gadget redirects to the Bugzilla issue (if a Bugzilla server URL has been specified). This allows JIRA to become the one interface for all JIRA and Bugzilla data.

What does it look like?

The **Bugzilla ID Search** gadget should appear as follows on the dashboard:

Adding the 'Bugzilla ID Search' gadget to Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Bugzilla ID Search' gadget.
3. Click Add.
4. Optionally specify a URL for the Bugzilla server.



5. Click Save.
6. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Calendar Gadget

The **Calendar** gadget shows issues and versions in a calendar format based on their due date. Calendars can be based on an issue filter or on a project.

What does it look like?

The **Calendar** gadget should appear as follows on the dashboard:

> The Calendar gadget will only be available to add to your dashboard if your JIRA administrator has installed the Calendar plugin.

Adding the 'Calendar' gadget to your Dashboard

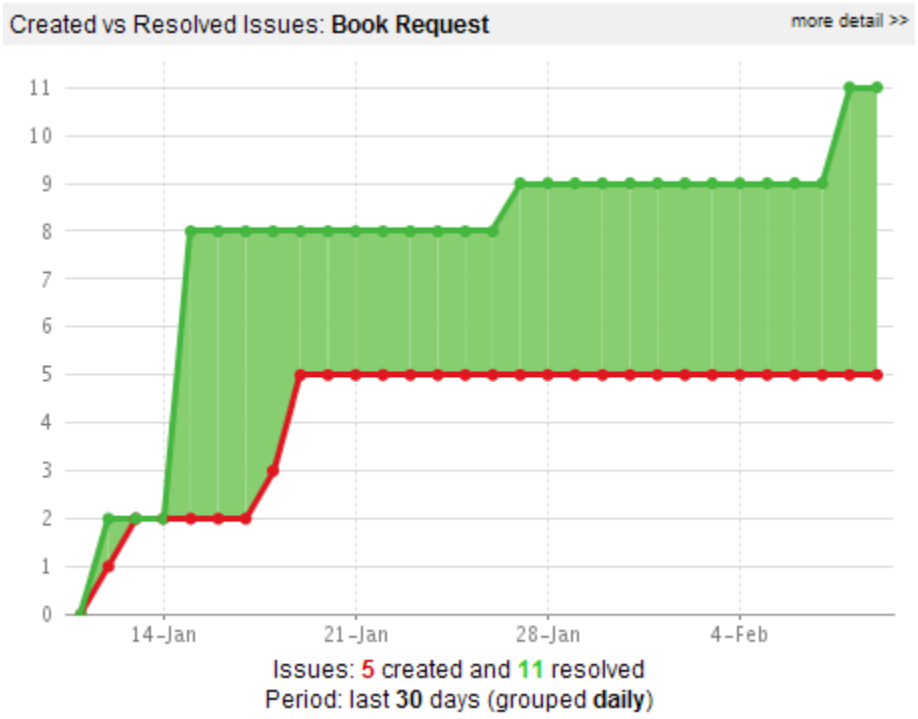
1. Navigate to the dashboard configuration page.
2. Select the 'Calendar' gadget.
3. Click 'Add'.
4. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Created vs Resolved Gadget

The **'Created vs Resolved Issues'** gadget displays a difference chart showing the number of issues created vs number of issues resolved over a given period of time. The chart is based on your choice of project or issue filter, and the chart can either be cumulative or not.

What does it look like?

The **'Created vs Resolved Issues'** portlet will appear as follows on the dashboard:

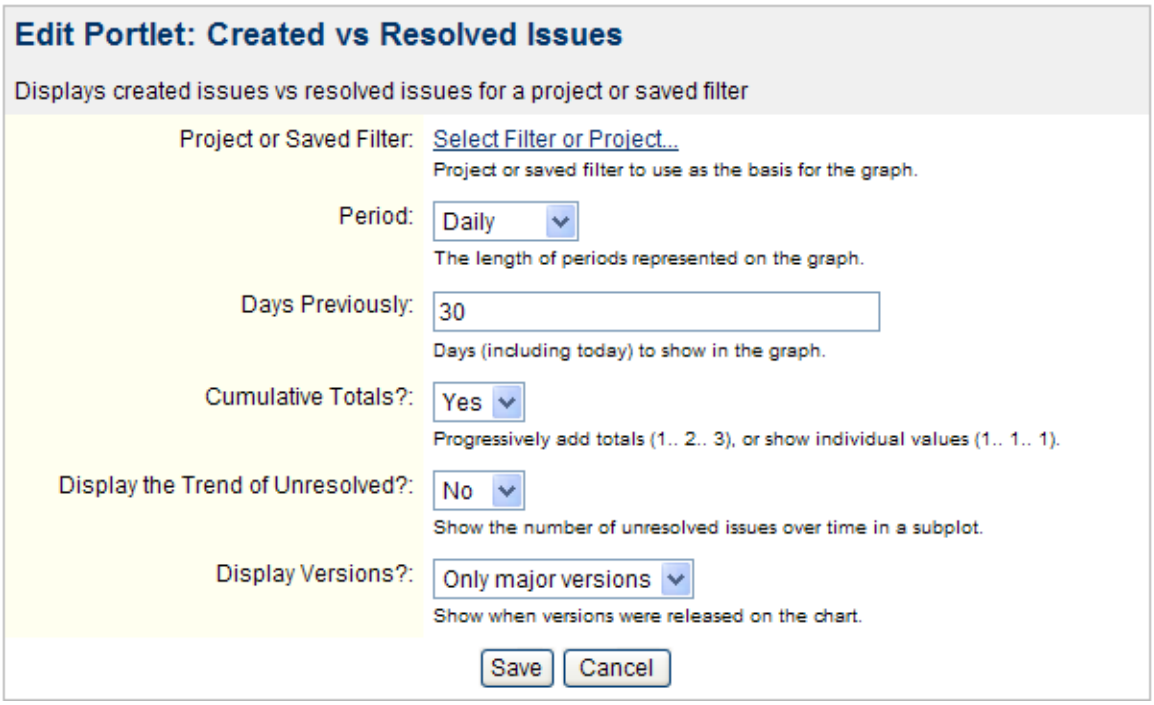


Click the **'more detail'** link to go to the full-size report and data table.

Adding the 'Created vs Resolved Issues' gadget to your Dashboard

To add the **'Created vs Resolved Issues'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Created vs Resolved Issues'** gadget.
3. Click the **Add** button.
4. The **'Edit Gadget'** screen will be displayed:

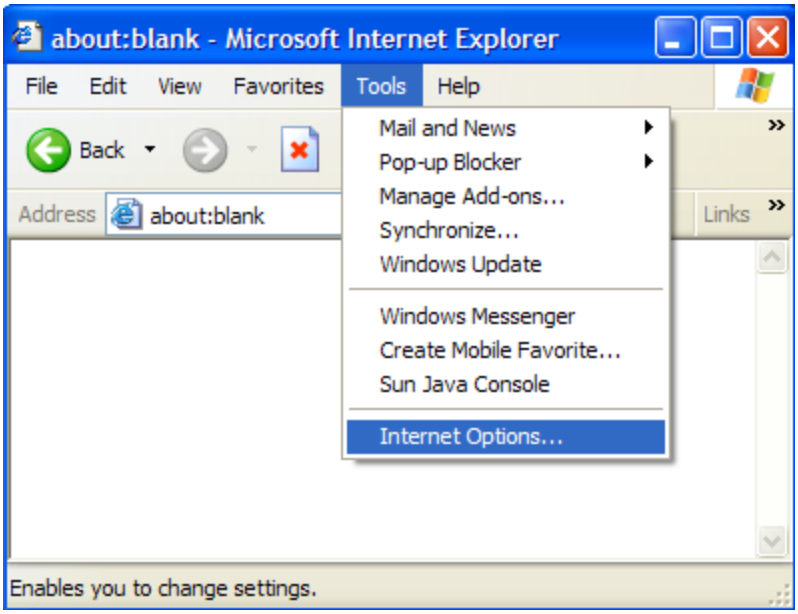


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Period'** field, select the timeframe on which the chart will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
8. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the chart.
9. In the **'Cumulative Totals?'** field, choose either:
    - **'Yes'** to progressively add data to the preceding column; or
    - **'No'** to show just a single value in each column.
10. In the **'Display Versions?'** field, choose either:
    - **'All versions'** to show version release dates on the chart, for all released versions; or
    - **'Only major versions'** to show version release dates on the chart, for released versions that are named 'x.x' only; or
    - **'None'** to not show version release dates on the chart.
11. In the **'Display the Trend of Unresolved?'** field, choose either:
    - **'Yes'** to display an additional line graph showing the number of unresolved issues over time; or
    - **'No'** to show just the difference chart of issues created vs issues resolved.
12. Click the **'Save'** button.
13. The gadget can be positioned on the dashboard through the dashboard configuration page.

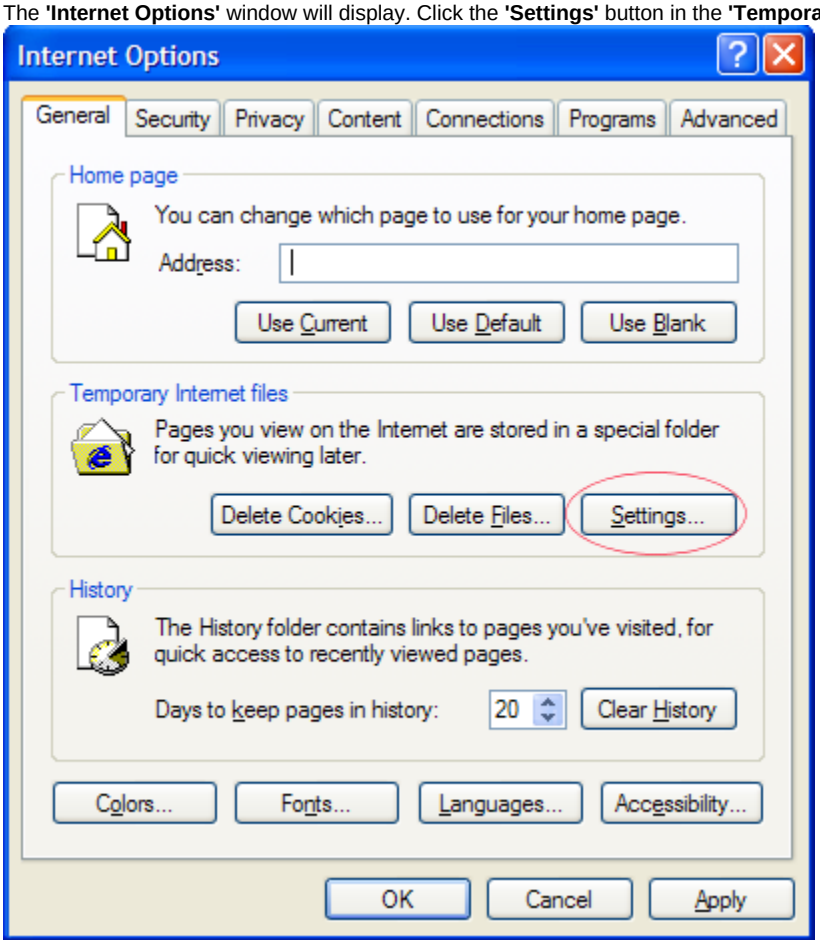
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

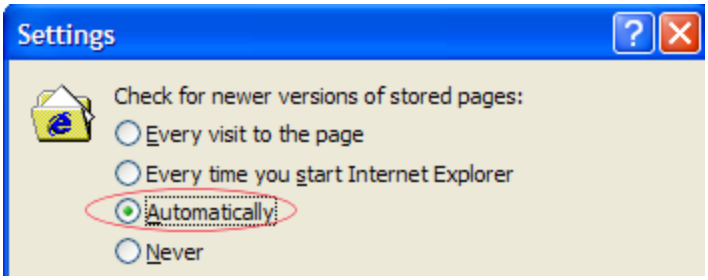
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Adding the Favourite Filters Gadget

The **Favourite Filters** gadget displays a list of all the issue filters that have currently been added by you as a 'favourite' filter.

Read more about adding an issue filter as a favourite filter in the issue filters documentation.

What does it look like?

The **Favourite Filters** gadget should appear as follows on the dashboard:

Favourite Filters (Create New | Manage Filters)	
All issues	5
Bugs Filter	4
Critical and blockers	0
Improvements	1

Adding the 'Favourite Filters' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Favourite Filters' gadget.
3. Click 'Add'.
4. The gadget ortlet can be positioned on the dashboard through the dashboard configuration page.

The Favourite Filters gadget is added by default to the 'System Default' dashboard.

Note: The '**Favourite Filters**' gadget has replaced the '**List All Filters**' portlet.

Adding the Filter Statistics Gadget

The **Filter Statistics** gadget displays the collection of issues returned from a specified filter, broken down by a specified field.

For instance, a filter can be created to return all open issues from all projects. The gadget can then be configured to display these issues broken down by a field (e.g. Assignee).

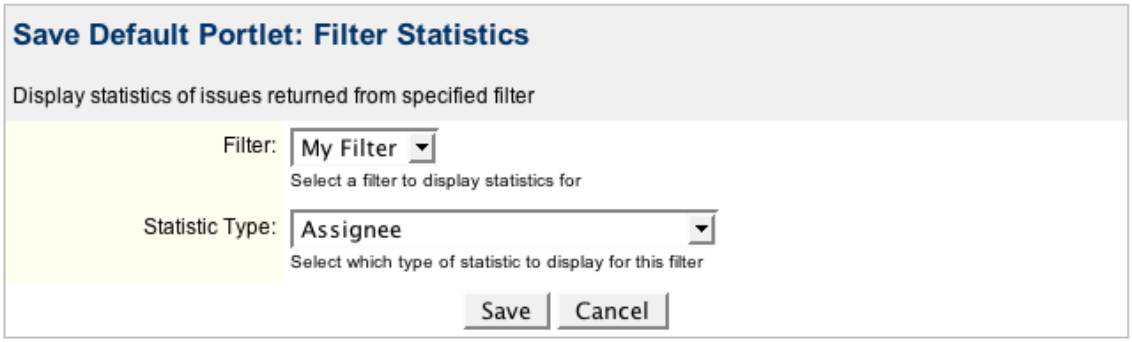
What does it look like?

The **Filter Statistics** gadget should appear as follows on the dashboard:

Statistics: **My Filter** (Assignee)		Total Issues: 8
Administrator	5	62%
Developer	2	25%
Test User	1	13%

Adding the 'Filter Statistics' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Filter Statistics' gadget.
3. Click '**Add**'.
4. Click the '**SELECT FILTER**' link. In the popup window that appears, select the filter on which the gadget will focus.
5. Select the field on which the statistics will focus.



6. Click '**Save**'.
7. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Two-Dimensional Filter Statistics Gadget

The **Two Dimensional Filter Statistics** gadget displays statistical data based on a specified filter in a configurable table format.

For example, a filter can be created to retrieve all open issues in all projects. The gadget can then be configured to display the statistical data on this collection of issues in a table with configurable axes — e.g. Assignee versus Status.

What does it look like?

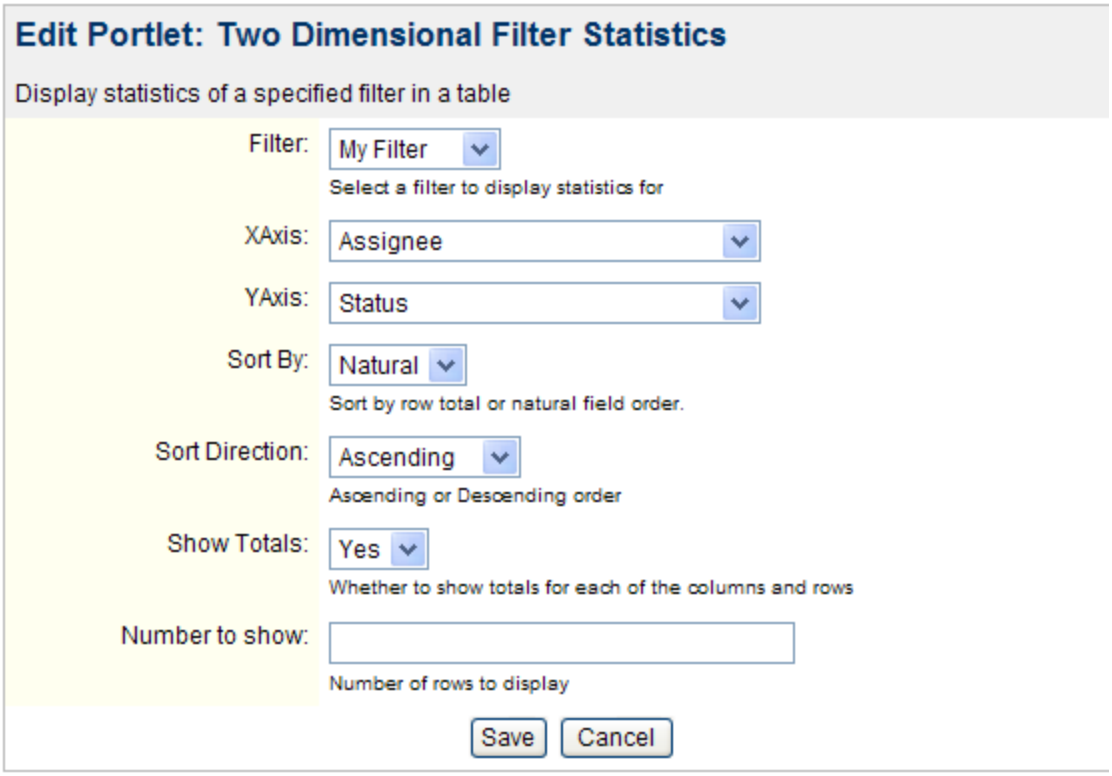
The **Two Dimensional Filter Statistics** gadget should appear as follows on the dashboard:

Statistics Table: My Filter

Assignee / Status	Administrator	Developer	Test User	T:
Open	4	2	1	7
Closed	1	0	0	1
Total:	**5**	**2**	**1**	**8**

Adding the Two Dimensional Filter Statistics gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the **'Two Dimensional Filter Statistics'** gadget.
3. Click **'Add'.** The 'Edit Gadget' screen will be displayed:



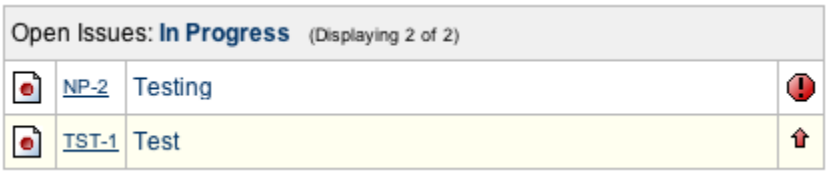
4. In the **'Filter'** field, click the 'SELECT FILTER' link and select the filter on which the portlet will be based, in the popup window that appears.
5. In the **'X-Axis'** field, select the field on which the horizontal axis will be based.
6. In the **'Y-Axis'** field, select the field on which the vertical axis will be based.
7. In the **'Sort By'** field, either:
    - select **'Total'** to sort the rows by their respective totals; or:
    - select **'Natural'** to sort the rows in the order that is natural for the chosen field.
8. In the **'Sort Direction'** field, select whether rows and columns will be sorted in ascending or descending order.
9. In the **'Show Totals'** field, select whether or not to display row and column totals.
10. In the **'Number to show'** field, select the number of issues which will be displayed in the portlet.
11. Click **'Save'.**
12. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the In Progress Gadget

The **In Progress** gadget displays all issues that are currently in progress and assigned to you.

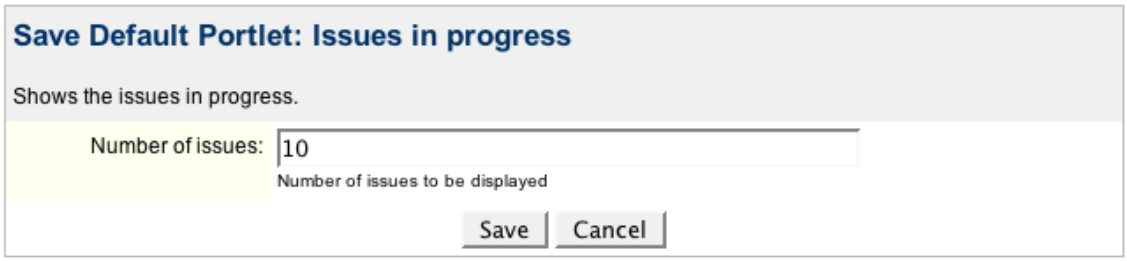
What does it look like?

The **In Progress** gadget should appear as follows on the dashboard:



Adding the 'In Progress' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the '**In Progress**' gadget.
3. Click '**Add**'.
4. Select the maximum number of issues which the portlet will gadget.



5. Click '**Save**'.
6. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Introduction Gadget

The **Introduction** gadget displays a configurable introduction message on the dashboard.

What does it look like?

The **Introduction** gadget should appear as follows on the dashboard:

Adding the 'Introduction' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Introduction' gadget.
3. Click 'Add'.
4. The gadget can be positioned on the dashboard through the dashboard configuration page.

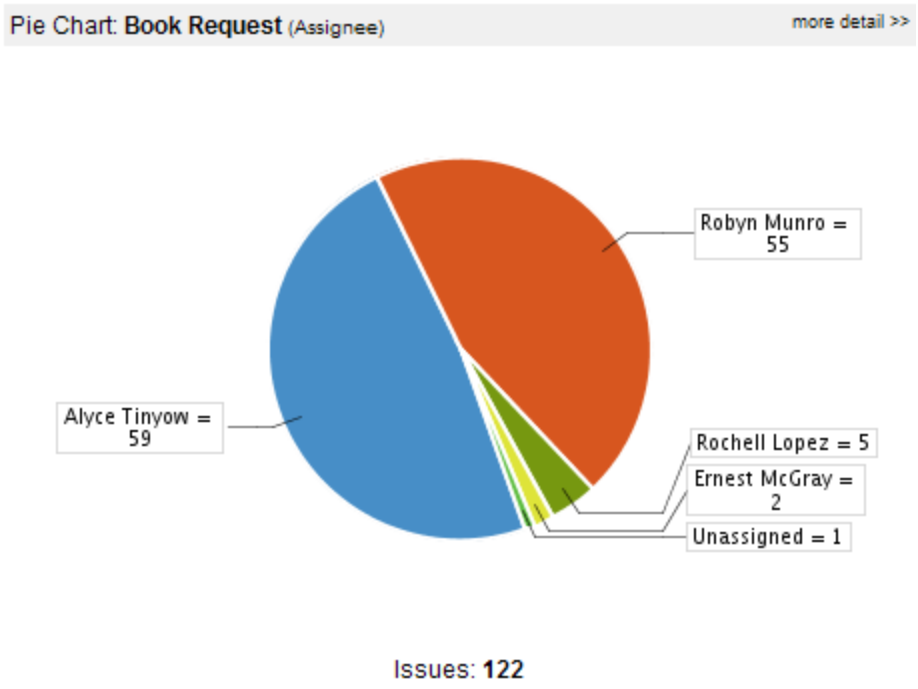
> The text/html displayed in the **Introduction** gadget is configured by your JIRA administration, through the JIRA configuration page.

Adding the Pie Chart Gadget

The **'Pie Chart'** gadget displays issues returned from a specified project or issue filter, grouped by a specified field. For example, an issue filter can be created to retrieve all open issues for a particular version of a particular project. The 'Pie Chart' gadget can then be used to display these issues grouped by a specified field (e.g. Assignee).

What does it look like?

The **'Pie Chart'** gadget will appear as follows on the dashboard:

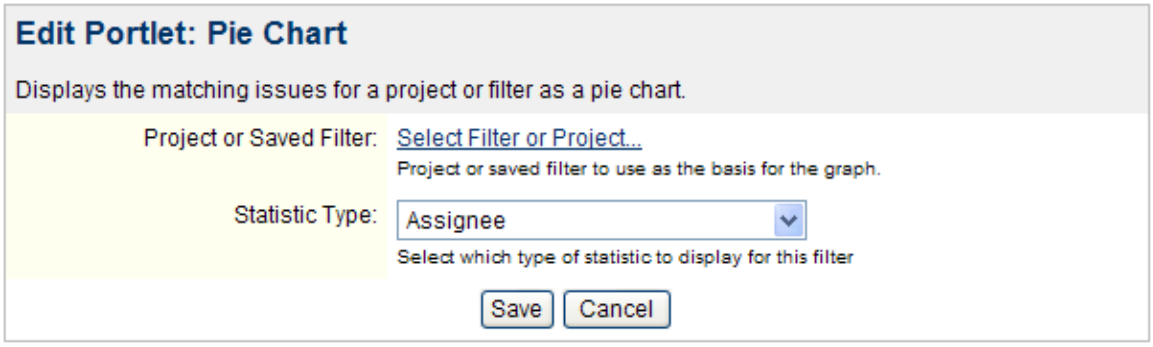


Click the **'more detail'** link to go to the full-size report and data table.

Adding Gadget to Dashboard

To add the **'Pie Chart'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Pie Chart'** gadget.
3. Click the **Add** button.
4. The **'Edit Gadget'** screen will be displayed:

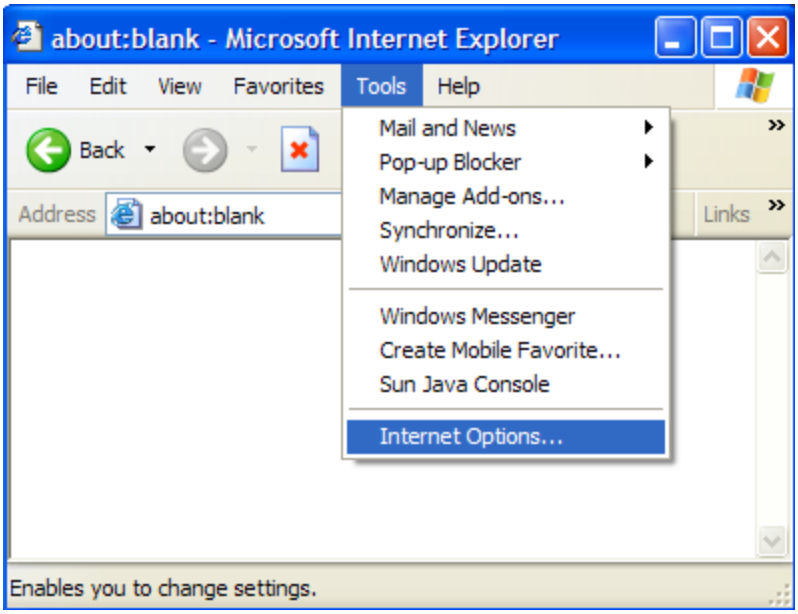


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Statistic Type'** field, select the field on which the pie chart will be based.
8. Click the **'Save'** button.
9. The gadget can be positioned on the dashboard through the dashboard configuration page.

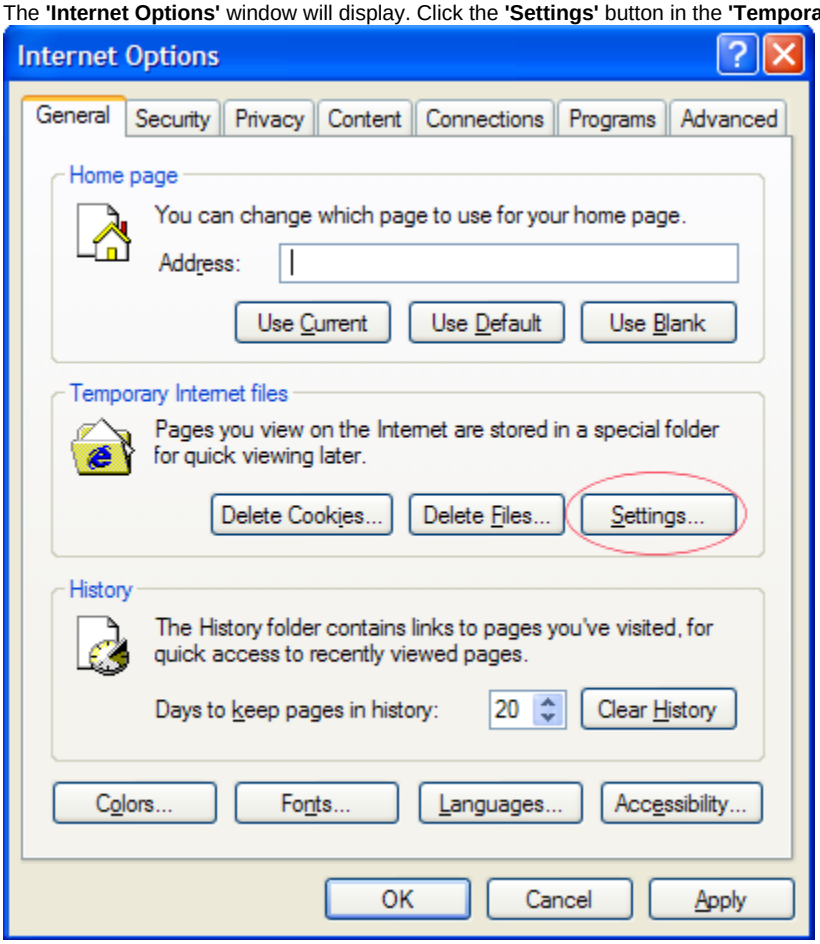
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

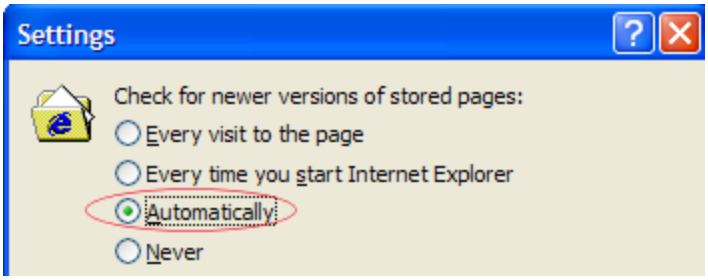
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.



Adding the Project Table Gadget

The **Project Table** gadget displays all the project names as clickable links in a table in the dashboard.

What does it look like?

The **Project Table** gadget should appear as follows on the dashboard:

Project Table gadget on dashboard grouped by **'All Projects':**

> Note: Standard and Professional can only view by All Projects

All Projects		
Category 1 Project (ONE)	Category 2 Project (TWO)	Category 2 Project 2 (SEC)
No Category Project (NONE)		

Project Table portlet on dashboard grouped by **'All Categories':**

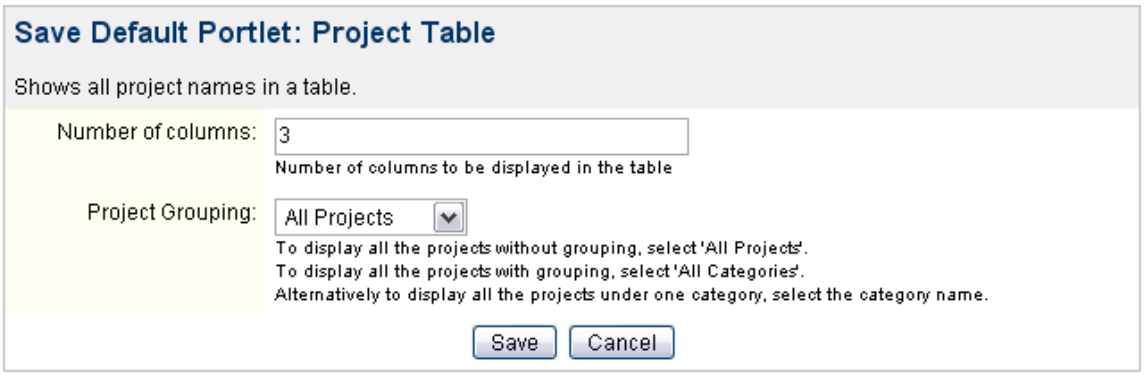
Category: **Category 1**		
Category 1 Project (ONE)		

Category: **Category 2**		
Category 2 Project (TWO)	Category 2 Project 2 (SEC)	

Projects with no category		
No Category Project (NONE)		

Adding the 'Project Table' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the '**Project Table**' gadget.
3. Click '**Add**'.
4. Select the maximum number of columns which will appear in the table.
5. Select one of:
    - *All Projects* — Displays all the projects under one table
    - *All Categories* — Displays all the projects grouped by categories
    - *Category Name* — Displays only the projects in the selected category **Note:** Project Grouping is only available in Enterprise Editions of JIRA 3.3 and later.



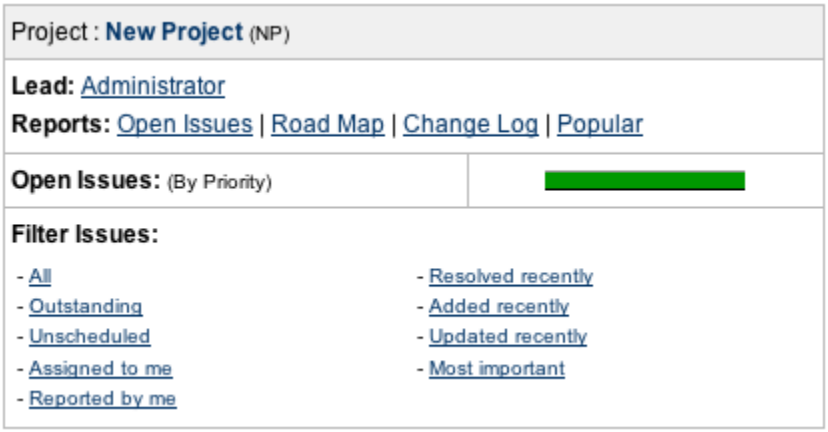
6. Click '**Save**'.
7. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Project Gadget

The **Project** gadget provides information and various filters related to a specified project on the dashboard.

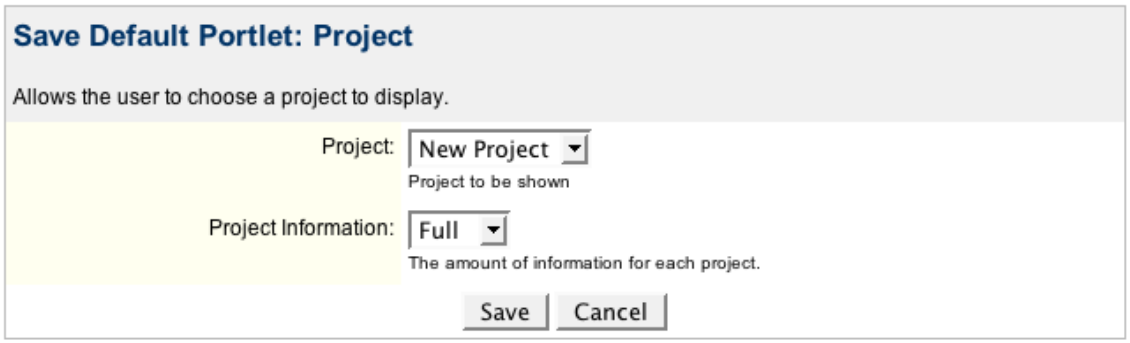
What does it look like?

The **Project** gadget should appear as follows on the dashboard:



Adding Portlet to Dashboard

1. Navigate to the dashboard configuration page.
2. Select the '**Project**' gadget.
3. Click '**Add**'.
4. Select the project on which the gadget will focus.
5. Select 'Full' or 'Brief' to specify the amount of information to be displayed.



6. Click '**Save**'.
7. The gadget can be positioned on the dashboard through the dashboard configuration page.

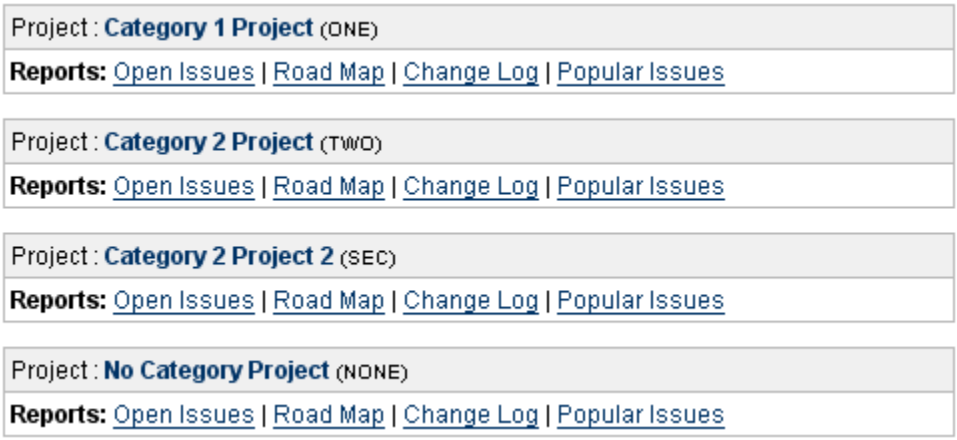
Adding the Projects Gadget

The **Projects** gadget provides information and various filters related to all projects within JIRA. If available, the gadget displays links to each project's:

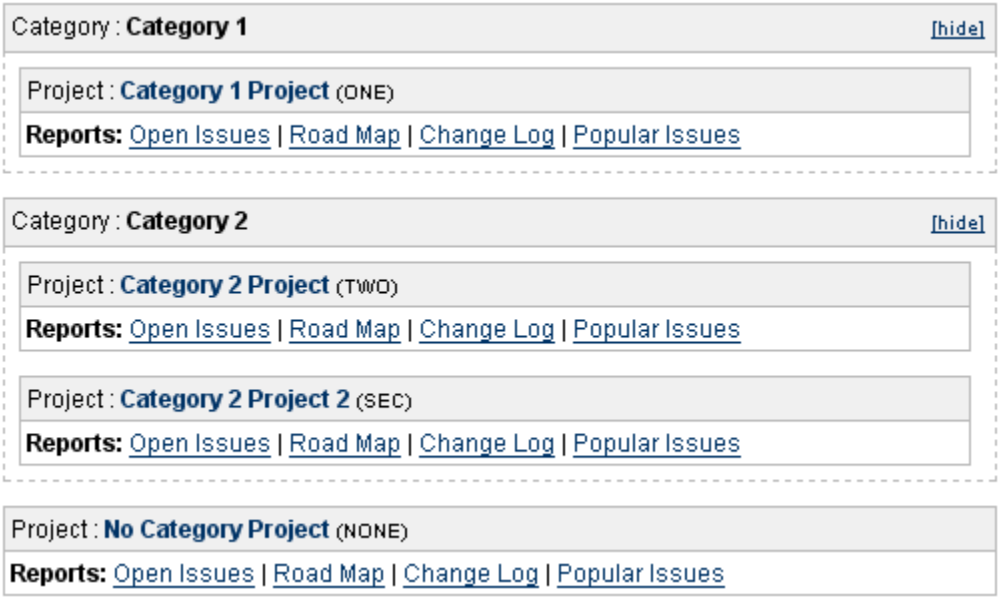
- Summary
- Issues
- Road Map
- Change Log
- Popular Issues
- Calendar
- Versions
- Components
- Bamboo Builds
- FishEye Changesets

What does it look like?

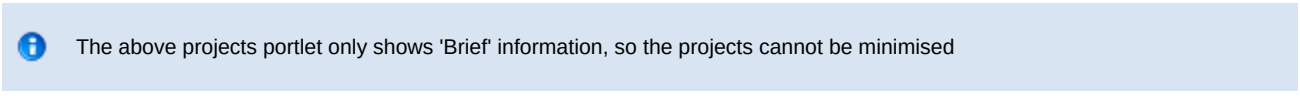
The Projects gadget should appear as follows on the dashboard for **Standard** and **Professional** editions:

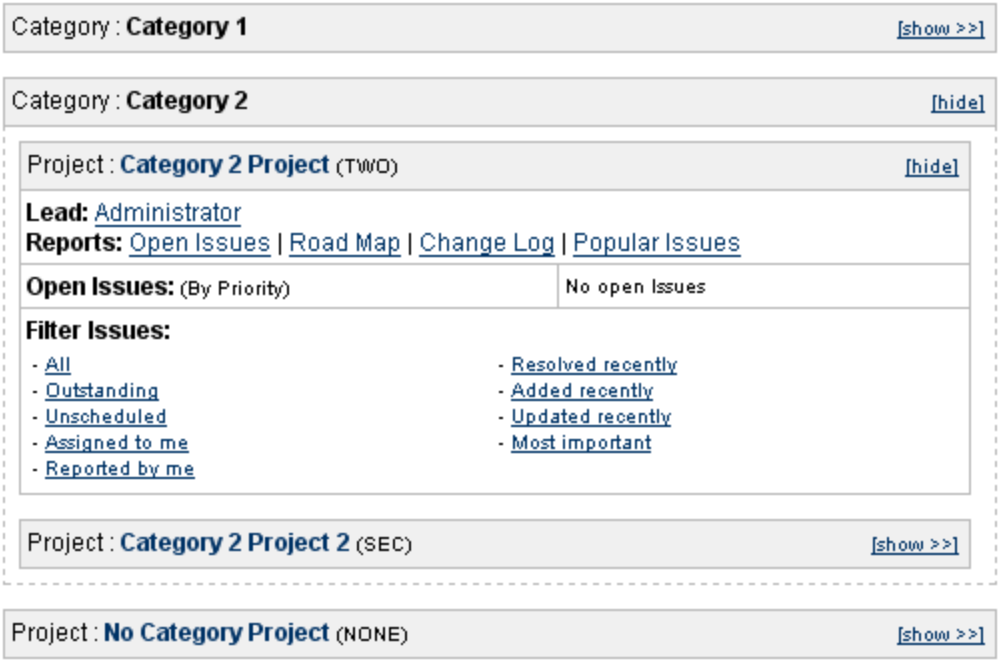


The Projects portlet should appear as follows on the dashboard for **Enterprise** edition:



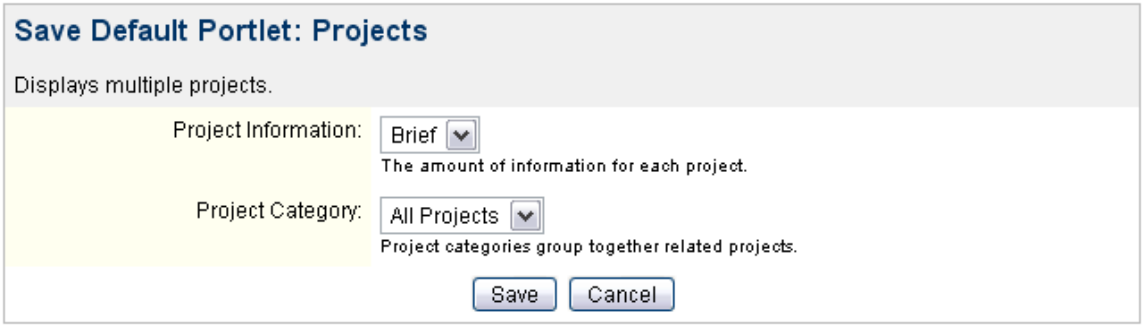
You can minimise the project or category table by clicking the 'hide' link. Likewise click the 'show >>' link to maximise the table as shown in following screenshot:





Adding the 'Projects' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the '**Projects**' gadget.
3. Click '**Add**'.
4. Select 'Full' or 'Brief' to specify the amount of information to be displayed.
5. Select 'All Projects' or a category name to filter the projects to be displayed. **Note:** Project Category selection is only available in JIRA Enterprise Edition.



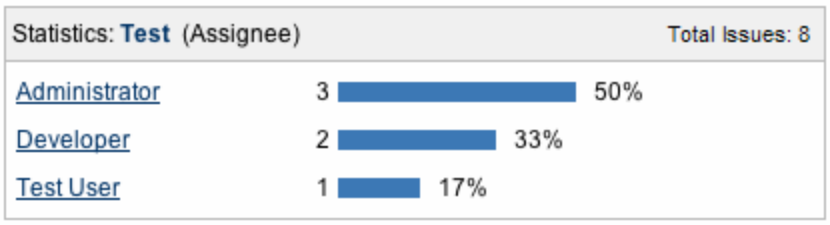
6. Click '**Save**'.
7. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Project Statistics Gadget

The **Project Statistics** portlet allows various per-project statistical data to be displayed on the dashboard. The portlet can be configured to display per-project statistical data based on various fields within JIRA (e.g. all open issues broken down by Assignee).

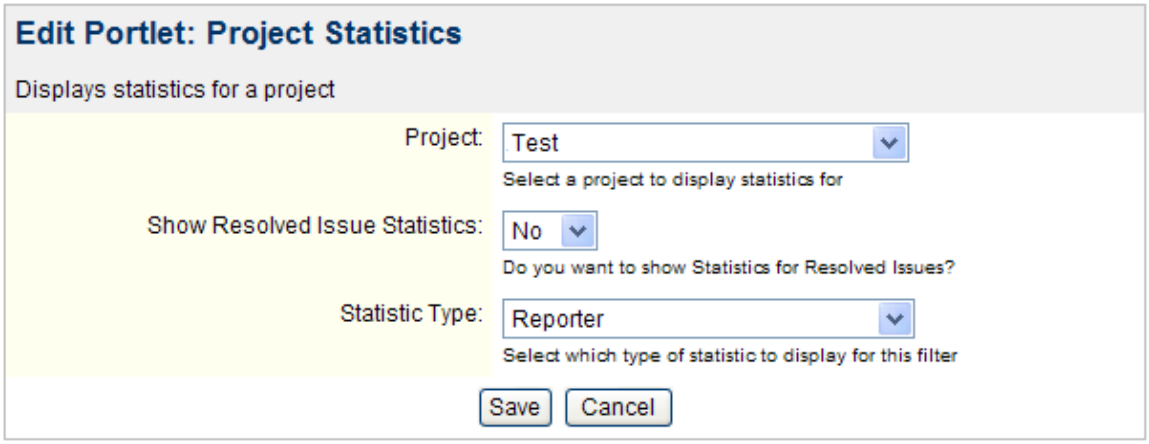
What does it look like?

The **Project Statistics** gadget should appear as follows on the dashboard:



Adding the 'Project Statistics' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the **'Project Statistics'** gadget.
3. Click **'Add'.**
4. Select the project on which the gadget will focus.
5. Choose whether or not to include resolved issues.
6. Select the field on which the statistics will focus.



7. Click the **'Save'** button.
8. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Quick Links Gadget

The **Quick Links** portlet gadget a number of useful links to issues associated with the current user. Each link takes you to the Issue Navigator, displaying the relevant issues. The gadget provides links to:

- **Reported Issues** — all issues reported by you
- **Voted Issues** — all issues for which you have voted
- **Watched Issues** — all issues that you are watching

What does it look like?

The **Quick Links** gadget should appear as follows on the dashboard:

Adding the Quick Links gadget to your Dashboard

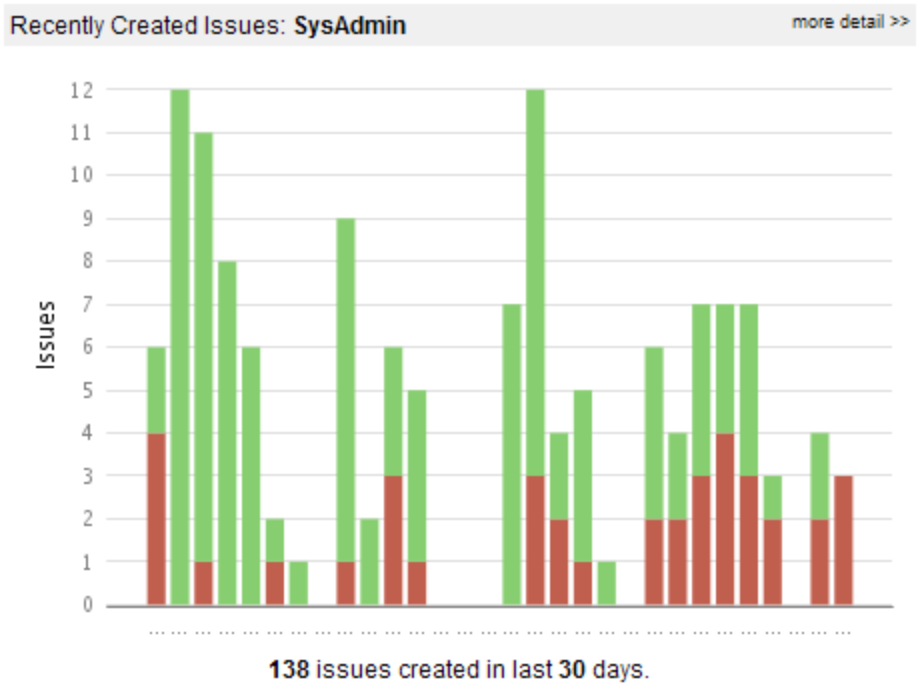
1. Navigate to the dashboard configuration page.
2. Select the 'Quick Links' gadget.
3. Click '**Add**'.
4. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Recently Created Issues Gadget

The **'Recently Created Issues'** gadget displays a bar chart showing the rate at which issues are being created, as well as how many of those created issues are resolved. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does it look like?

The **'Recently Created Issues'** gadget will appear as follows on the dashboard:

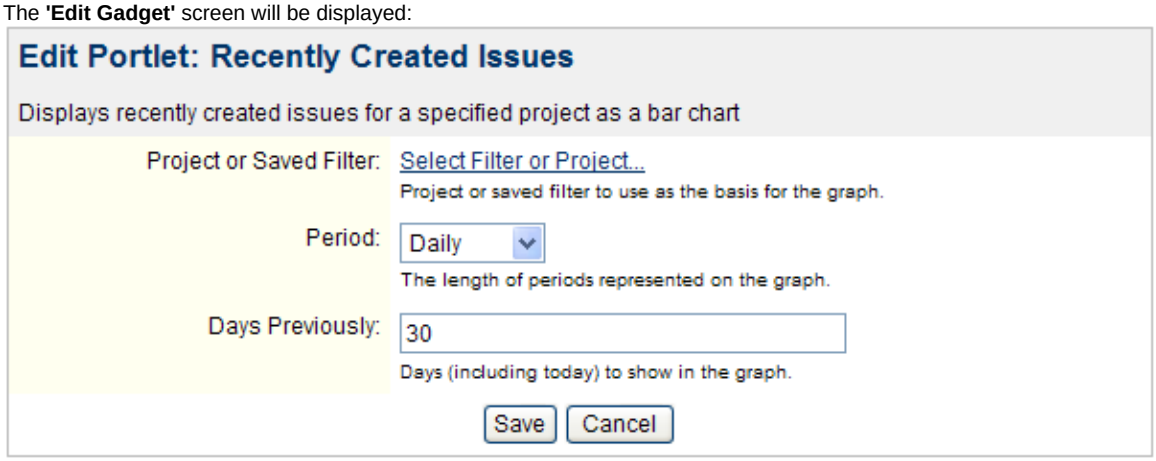


Click the **'more detail'** link to go to the full-size report and data table.

Adding the 'Recently Created Issues' gadget to your Dashboard

To add the **'Recently Created Issues'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Recently Created Issues'** gadget.
3. Click the **Add** button.
4. The **'Edit Gadget'** screen will be displayed:

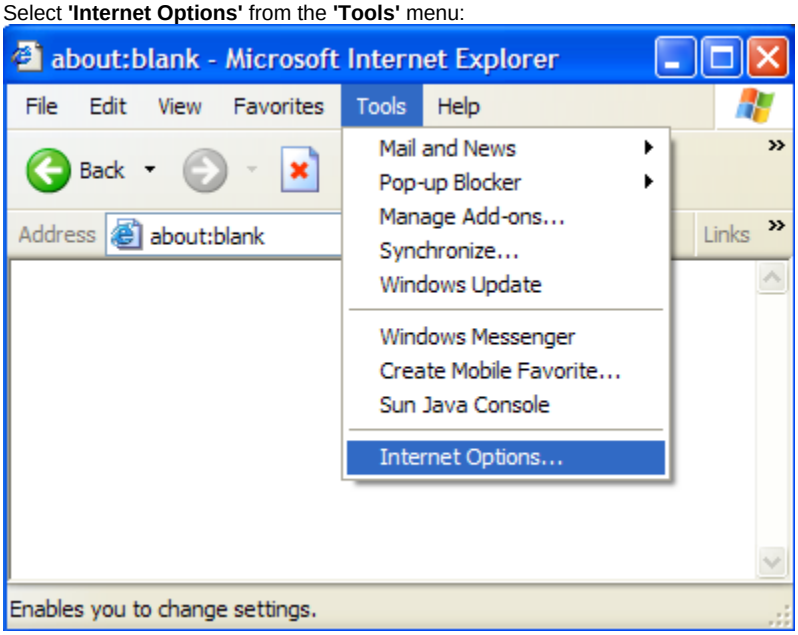


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Period'** field, select the timeframe on which the chart will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
8. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the chart.
9. Click the **'Save'** button.
10. The gadget can be positioned on the dashboard through the dashboard configuration page.

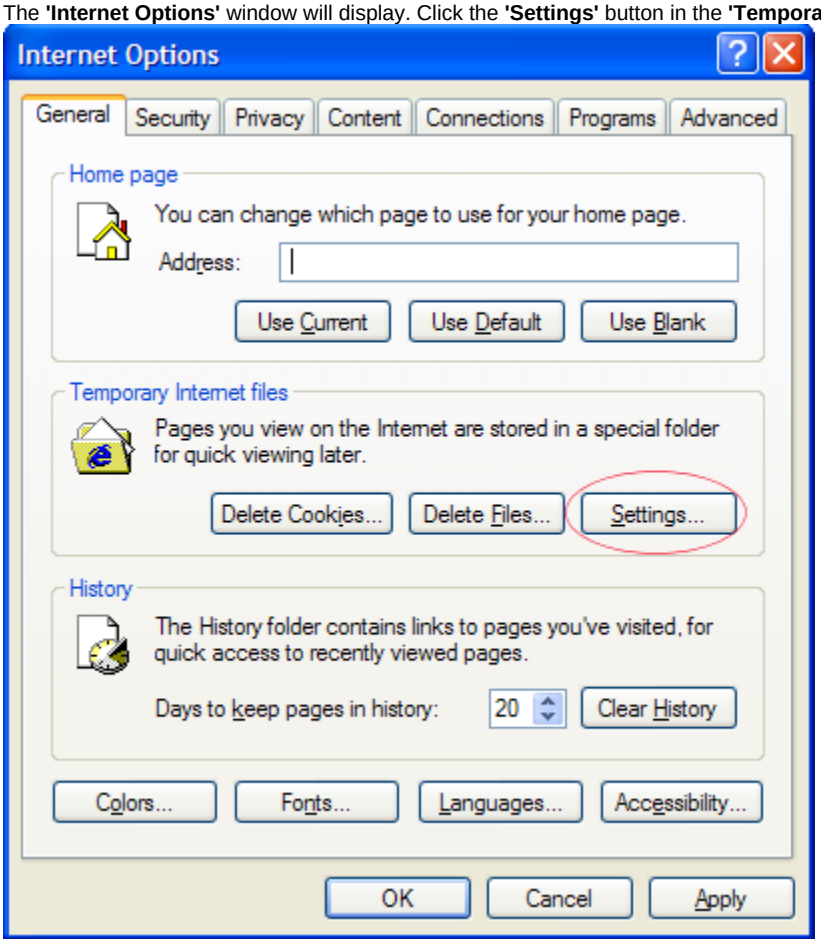
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

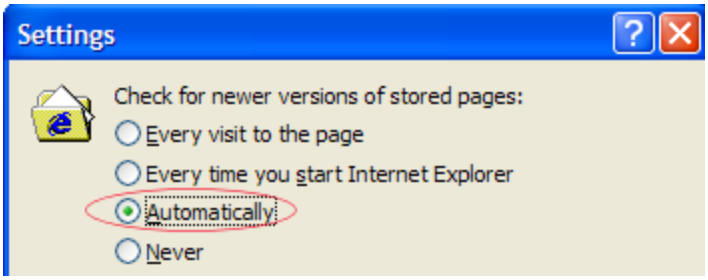
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.

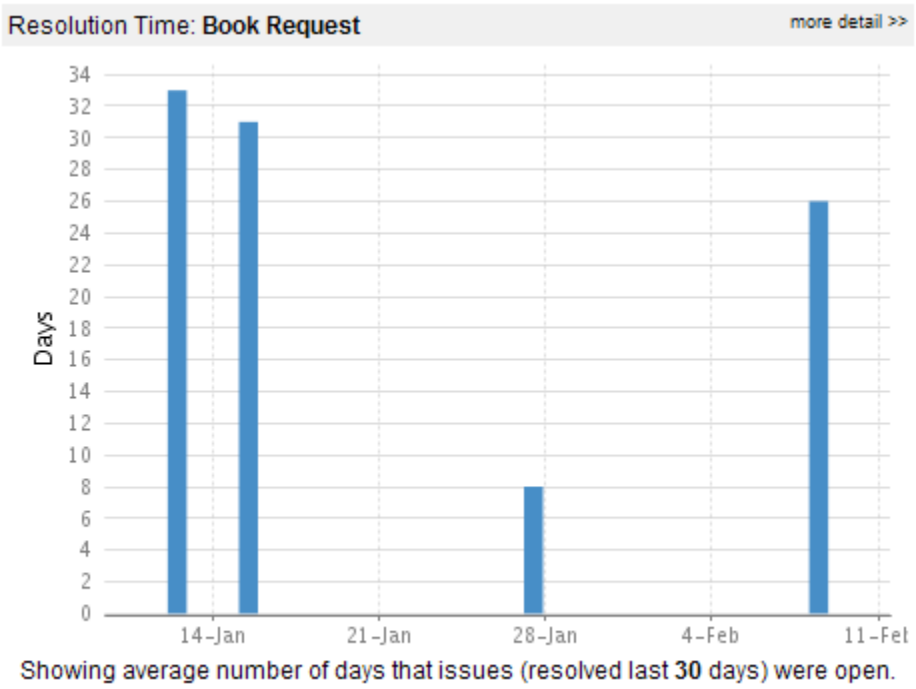


Adding the Resolution Time Gadget

The **'Resolution Time'** gadget displays a bar chart showing the average time taken to resolve issues. This is useful to show you the trends in resolution time. The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).

What does it look like?

The **'Resolution Time'** gadget will appear as follows on the dashboard:

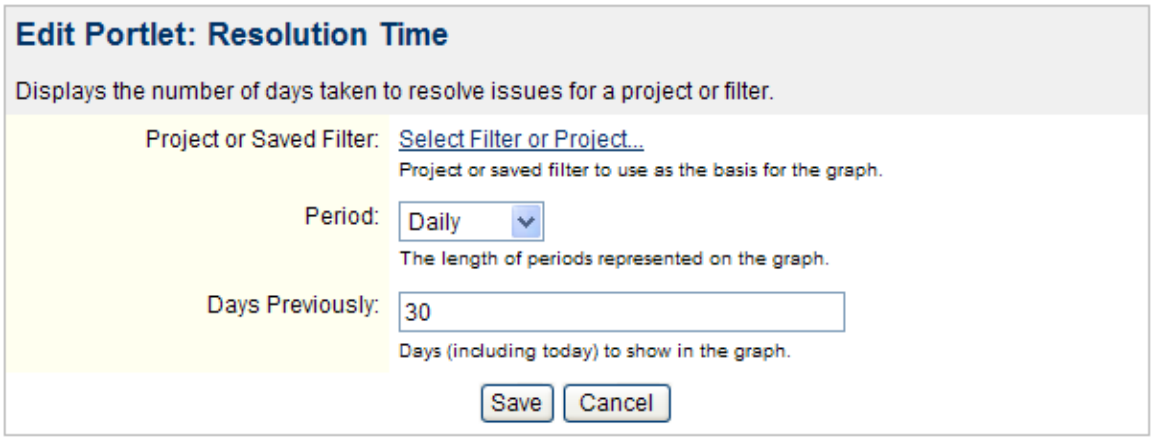


Click the **'more detail'** link to go to the full-size report and data table.

Adding the 'Resolution Time' Gadget to your Dashboard

To add the **'Resolution Time'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Resolution Time'** gadget.
3. Click the **Add** button.
4. The **'Edit Gadget'** screen will be displayed:

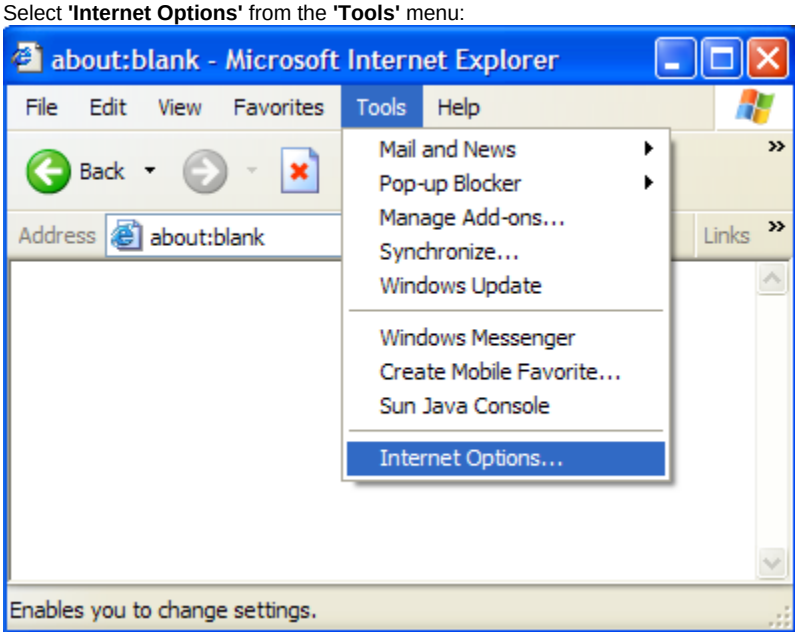


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Period'** field, select the timeframe on which the chart will be based:
    - **'Hourly'**
    - **'Daily'**
    - **'Weekly'**
    - **'Quarterly'**
    - **'Yearly'**
8. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the chart.
9. Click the **'Save'** button.
10. The gadget can be positioned on the dashboard through the dashboard configuration page.

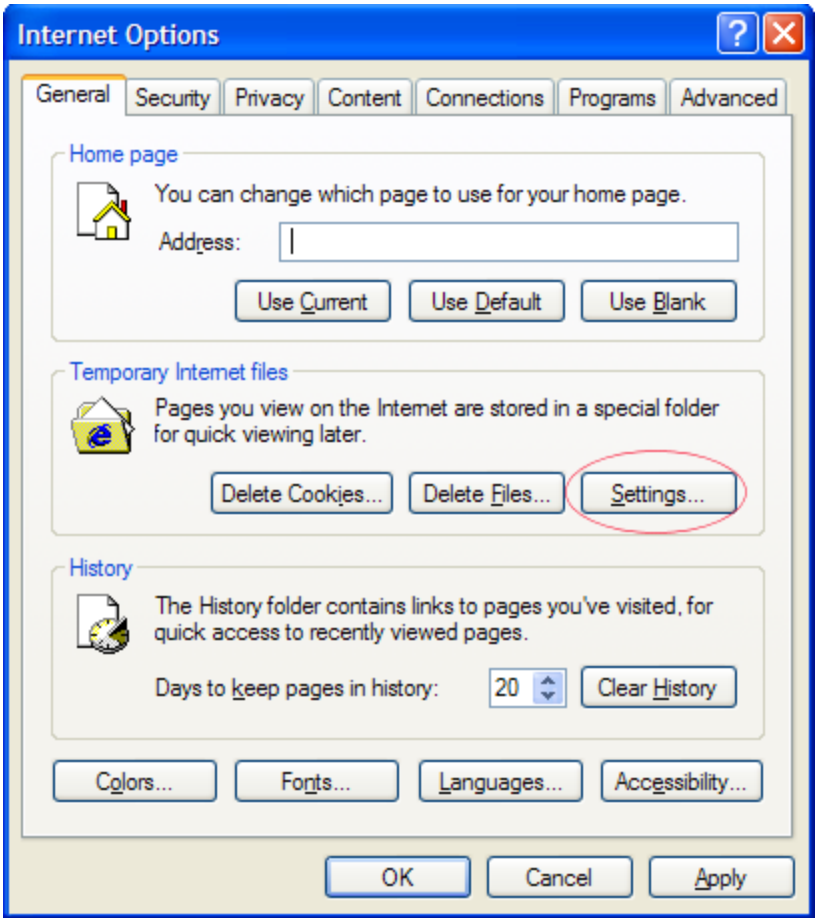
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

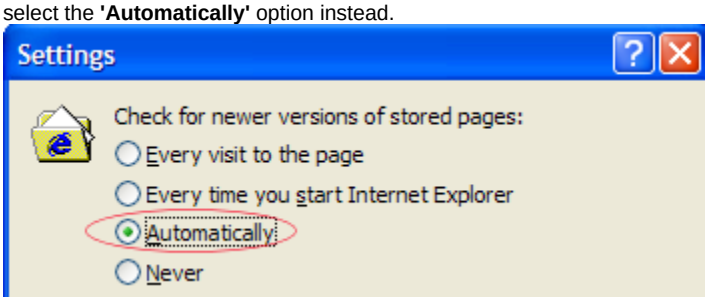
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.

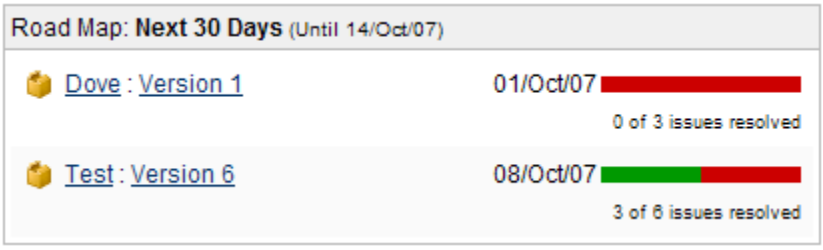


Adding the Road Map Gadget

The **Road Map** gadget shows versions which are due for release within a specified period of time, and a summary of progress made towards completing the issues in those versions.

What does it look like?

The **Road Map** gadget should appear as follows on the dashboard:

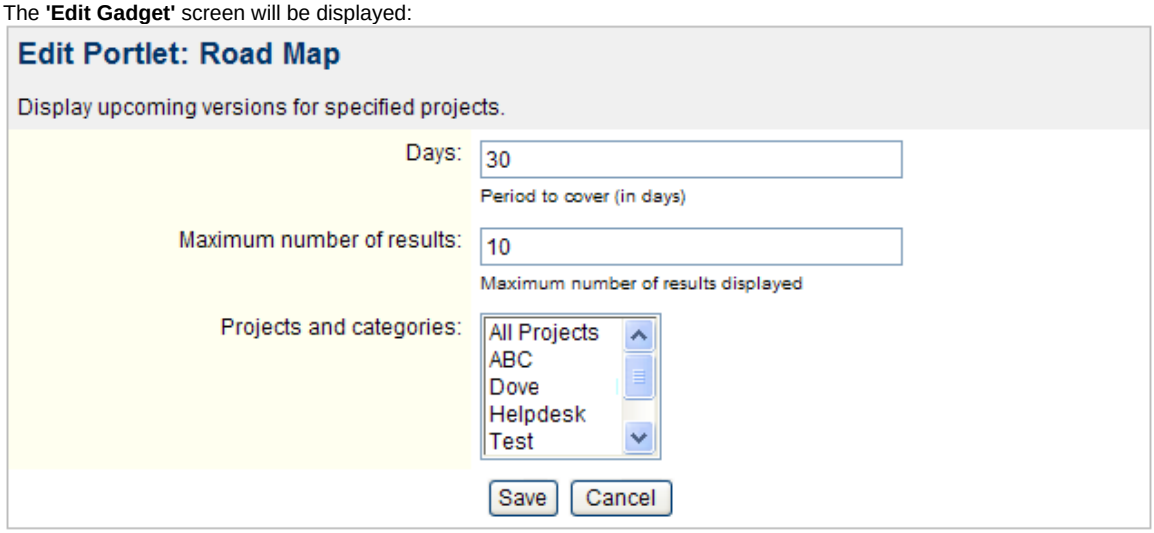


You can:

- Click the name of a project (e.g. 'Dove') to browse the project.
- Click the name of a version (e.g. 'Version 1') to browse the version.
- Click the progress bar (shown in red and/or green) to view the version's issues in the Issue Navigator.

Adding the 'Road Map' Gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the **'Road Map'** gadget.
3. Click the 'Add' button (located at the bottom of the screen).
4. The **'Edit Gadget'** screen will be displayed:



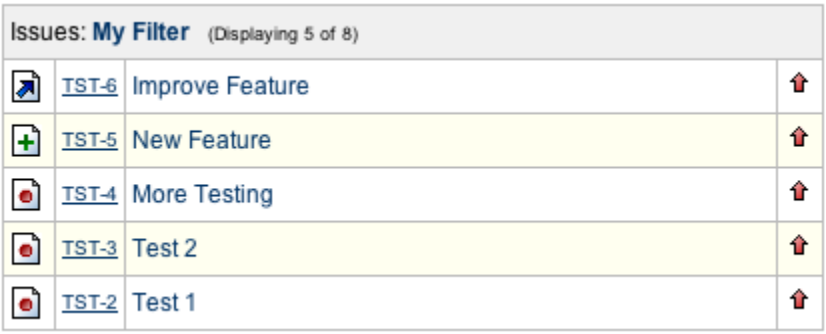
5. In the **'Days'** field, specify the period of time (in days) for which you wish to view versions due for release.
6. In the **'Maximum number of results'** field, specify the maximum number of versions you wish the gadget to display.
7. In the **'Projects and categories'** field, select the projects (or project categories) in which you are interested; or select **'All projects'** to display versions for any project in your JIRA system.
8. Click the **'Save'** button.
9. The gadget will be added to the bottom left of your dashboard. You can reposition it as described in Configuring the Dashboard.

Adding the Saved Filter Gadget

The **Saved Filter** gadget displays the results of a specified issue filter on the dashboard. It can be configured to display a maximum number of issues from the collection returned from the specified filter.

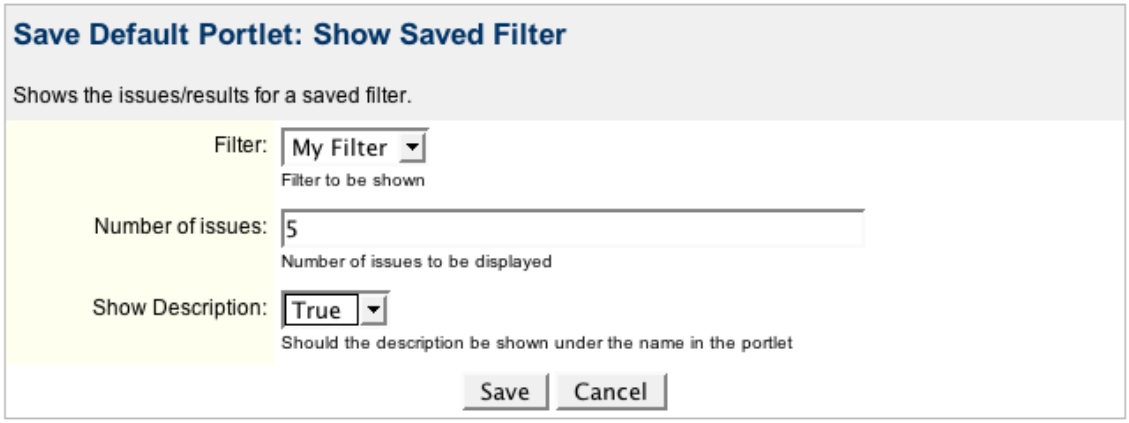
What does it look like?

The **'Saved Filter'** gadget should appear as follows on the dashboard:



Adding the 'Show Saved Filter' Gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the '**Saved Filter**' gadget.
3. Click '**Add**'.
4. Click the '**SELECT FILTER**' link and select the filter on which the gadget will focus, in the popup window that appears.
5. Select the maximum number of issues that the gadget will display.
6. Select if the filter description (if any) should be displayed within the gadget.



7. Click '**Save**'.
8. The gadget can be positioned on the dashboard through the dashboard configuration page.

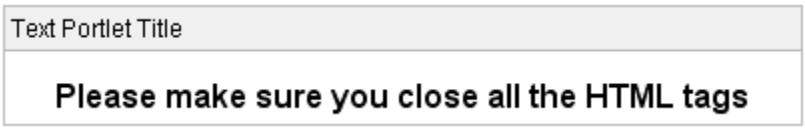
Adding the Text Gadget

The **Text** gadget displays your specified HTML text on the dashboard.

The Text gadget is disabled by default as it allows users to enter arbitary HTML. To enable the text gadget, please refer to Enabling Plugins section.

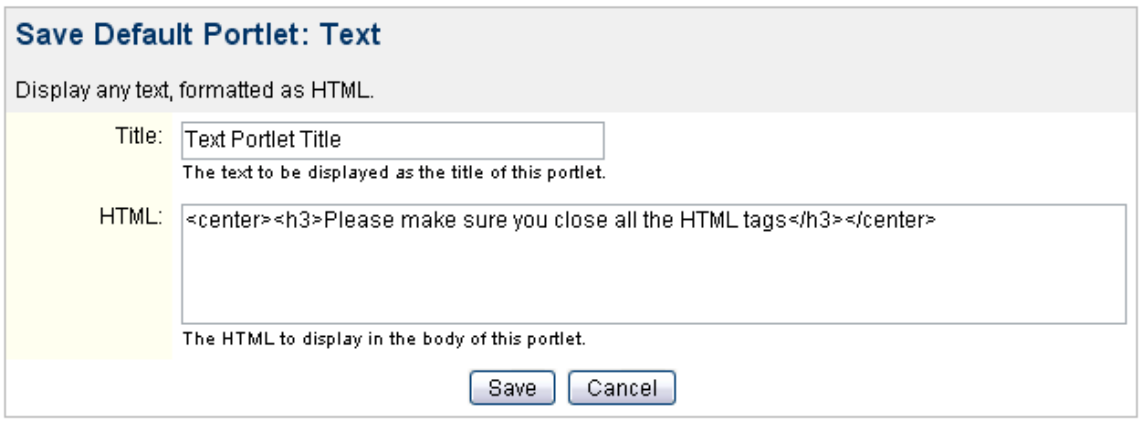
What does it look like?

The **Text** gadget should appear as follows on the dashboard:



Adding the 'Text' Gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the 'Text' gadget.
3. Click Add.
4. Enter a title for the Text gadget.
5. Enter the body text; this may include HTML.



6. Click Save.
7. The gadget can be positioned on the dashboard through the dashboard configuration page.

Adding the Time Since Issues Gadget

The **'Time Since Issues'** gadget displays a bar chart showing the number of issues for which your chosen date field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a custom field) was set on a given date. The report is based on your choice of project or issue filter, and your chosen units of

time (ie. hours, days, weeks, months, quarters or years).

What does it look like?

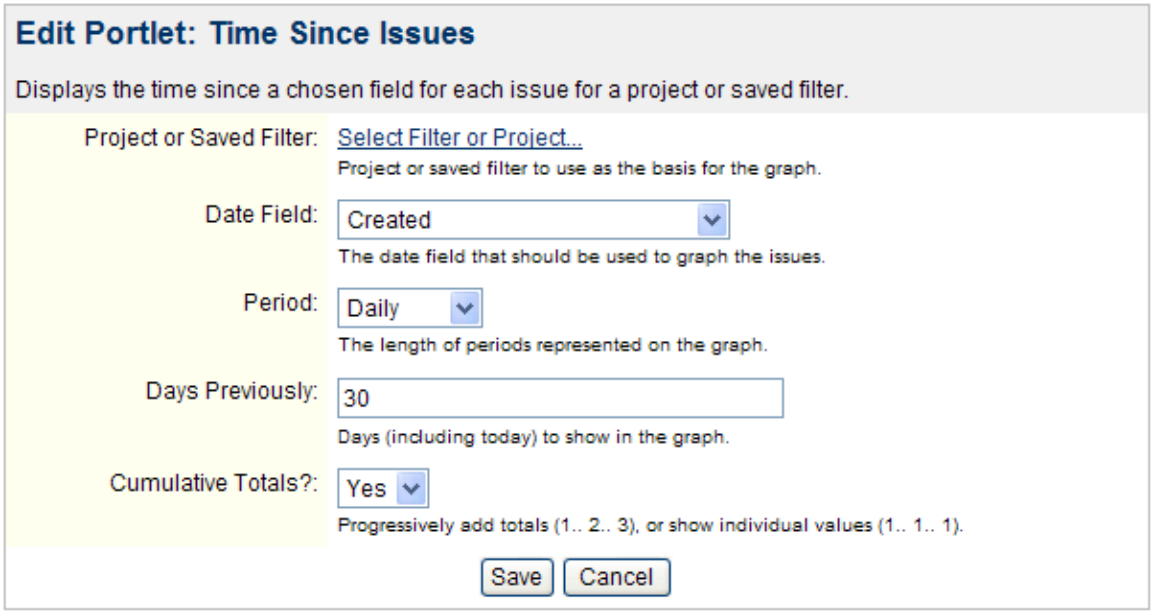
The **'Time Since Issues'** gadget will appear as follows on the dashboard:

Click the **'more detail'** link to go to the full-size report and data table.

Adding 'Time Since Issues' Gadget to your Dashboard

To add the **'Time Since Issues'** gadget to your dashboard:

1. Navigate to the dashboard configuration page.
2. Select the **'Time Since Issues'** gadget.
3. Click the '**Add**' button.
4. The **'Edit Gadget'** screen will be displayed:

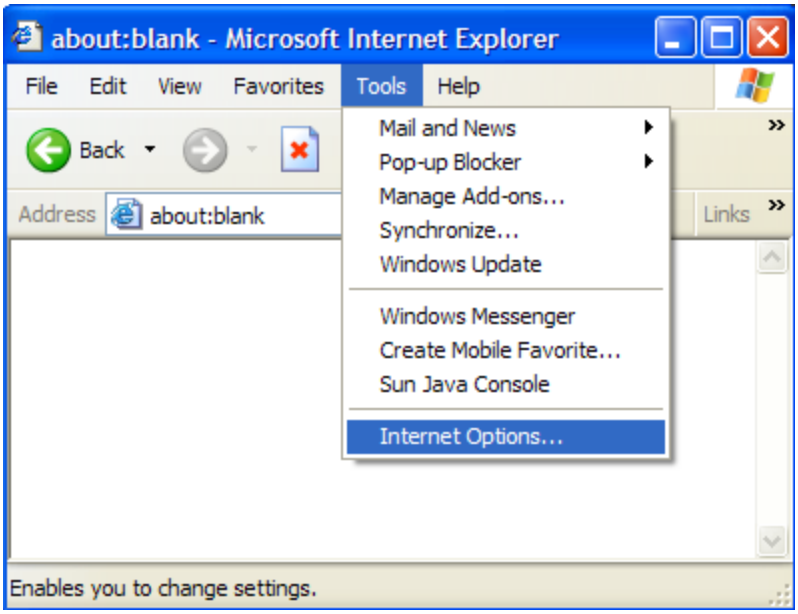


5. Click **'Select Filter or Project'.**
6. The popup **Filter or Project Picker** will appear. Select the project, or issue filter, in which you are interested. You will then be returned to the **'Edit Gadget'** screen.
7. In the **'Date Field'** field, select the date in which you are interested (e.g. **'Created', 'Updated', 'Due' , *'Resolved',** or a custom field of type 'Date').
   *Note: only available if time tracking has been enabled by your JIRA administrator.
8. In the **'Period'** field, select the timeframe on which the report will be based:
   - **'Hourly'**
   - **'Daily'**
   - **'Weekly'**
   - **'Quarterly'**
   - **'Yearly'**
9. In the **'Days Previously'** field, enter the number of days' worth of data (counting backwards from today) to be included in the report.
10. In the **'Cumulative Totals?'** field, choose either:
    - **'Yes'** to progressively add data to the preceding column; or
    - **'No'** to show just a single value in each column.
11. Click the **'Save'** button.
12. The gadget can be positioned on the dashboard through the dashboard configuration page.

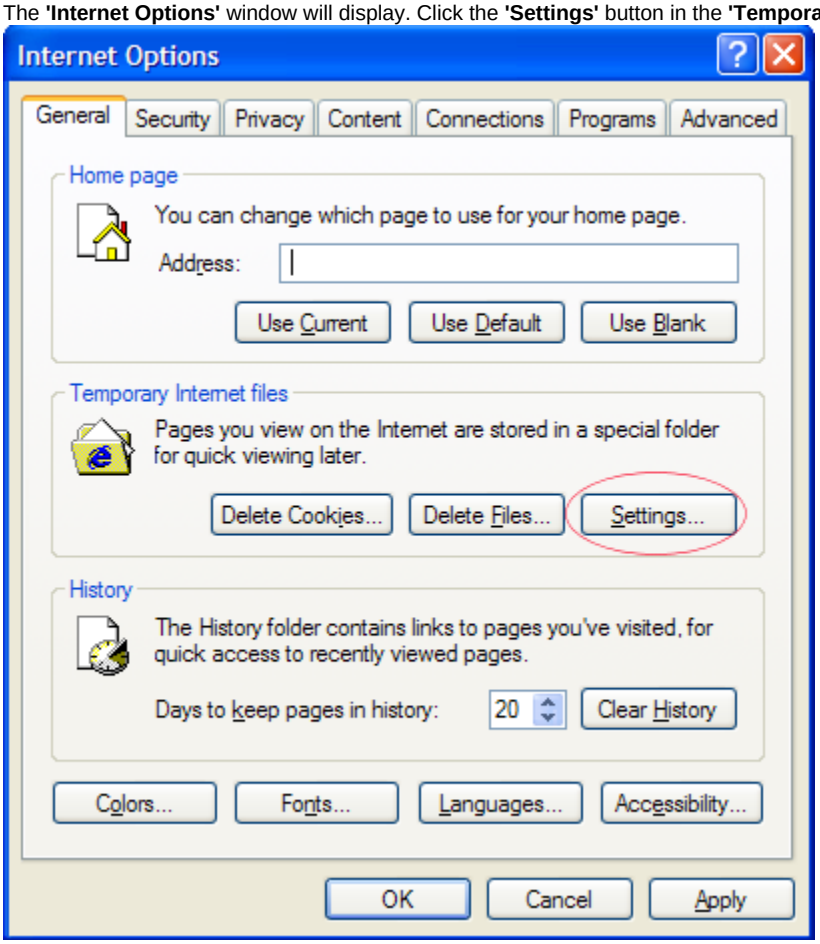
Configuring your Internet Explorer cache settings

If you use Internet Explorer, you will need to configure your browser to be able to print pages with charts correctly:

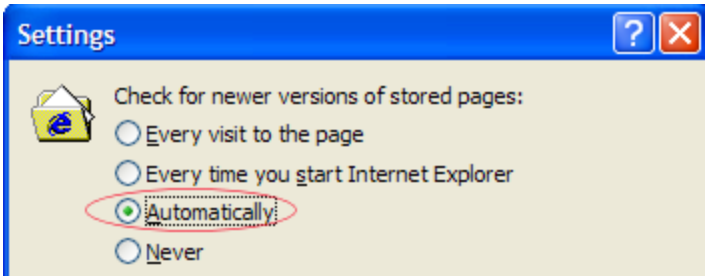
1. Select **'Internet Options'** from the **'Tools'** menu:



2. The **'Internet Options'** window will display. Click the **'Settings'** button in the **'Temporary Internet files'** (i.e. cache) section:



3. The **'Settings'** window will display. Ensure that you have do not have the **'Every visit to the page'** (i.e. no caching) option selected. If so, select the **'Automatically'** option instead.

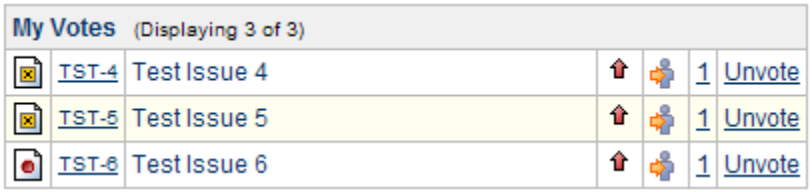


Adding the Voted Issues Gadget

The **Voted Issues** gadget shows issues for which you have voted.

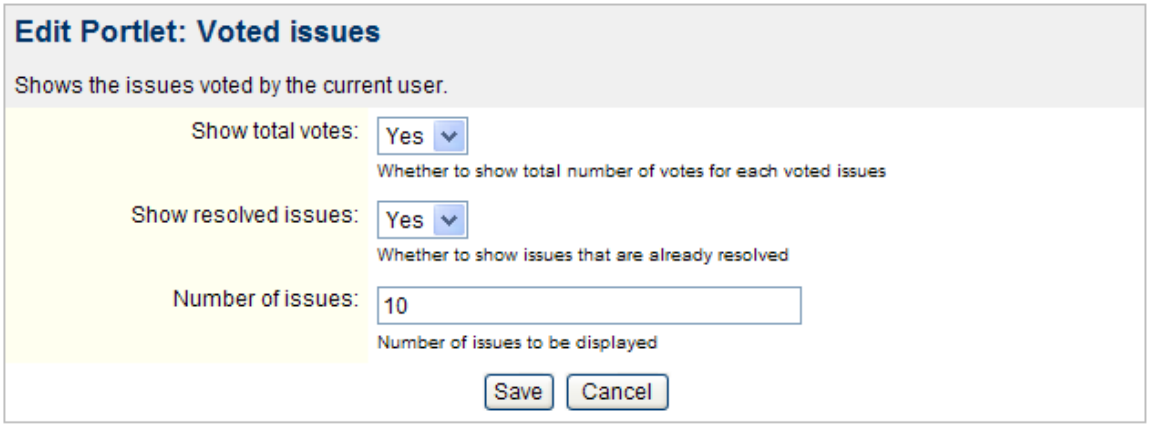
What does it look like?

The **Voted Issues** gadget should appear as follows on the dashboard:



Adding the 'Voted Issues' gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the **'Voted Issues'** gadget.
3. Click the 'Add' button (located at the bottom of the screen).
4. The **'Edit Gadget'** screen will be displayed:



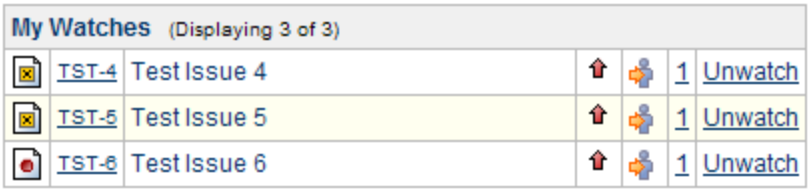
5. In the **'Show total votes'** field, select **'Yes'** if you wish the gadget to display the number of people who have voted for each issue.
6. In the **'Show resolved issues'** field, select **'Yes'** if you wish the gadget to display all issues on which you have voted, or select **'No'** if you wish the gadget to only display unresolved issues.
7. In the **'Number of issues'** field, specify the maximum number of issues you wish the gadget to display.
8. Click the **'Save'** button.
9. The gadget will be added to the bottom left of your dashboard. You can reposition it as described in Configuring the Dashboard.

Adding the Watched Issues Gadget

The **Watched Issues** gadget shows issues which you are watching.

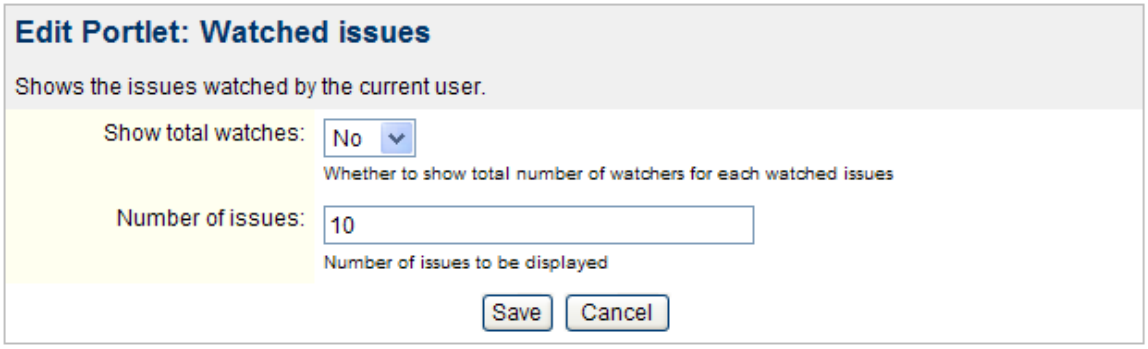
What does it look like?

The **Watched Issues** gadget should appear as follows on the dashboard:



Adding the 'Watched Issues' Gadget to your Dashboard

1. Navigate to the dashboard configuration page.
2. Select the **'Watched Issues'** gadget.
3. Click the 'Add' button (located at the bottom of the screen).
4. The **'Edit Gadget'** screen will be displayed:



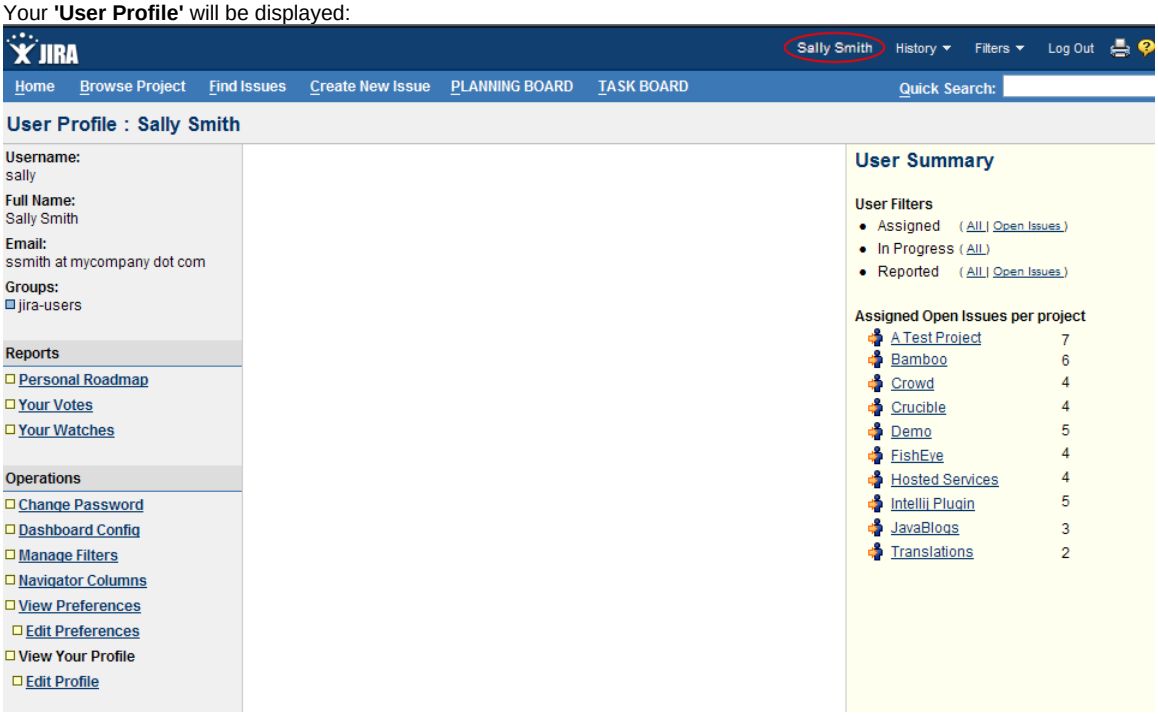
5. In the **'Show total watches'** field, select **'Yes'** if you wish the gadget to display the number of people who are watching each issue.
6. In the **'Number of issues'** field, specify the maximum number of issues you wish the gadget to display.
7. Click the **'Save'** button.
8. The gadget will be added to the bottom left of your dashboard. You can reposition it as described in Configuring the Dashboard.

Managing your User Profile

Your JIRA user profile is where you specify your JIRA settings (e.g. your email address, and the format in which you would like to receive email notifications). It also contains useful links to a number of personalised reports.

To view your JIRA user profile:

1. Click your user name at the top-right of the page.
2. Your **'User Profile'** will be displayed:



3. From here you can:

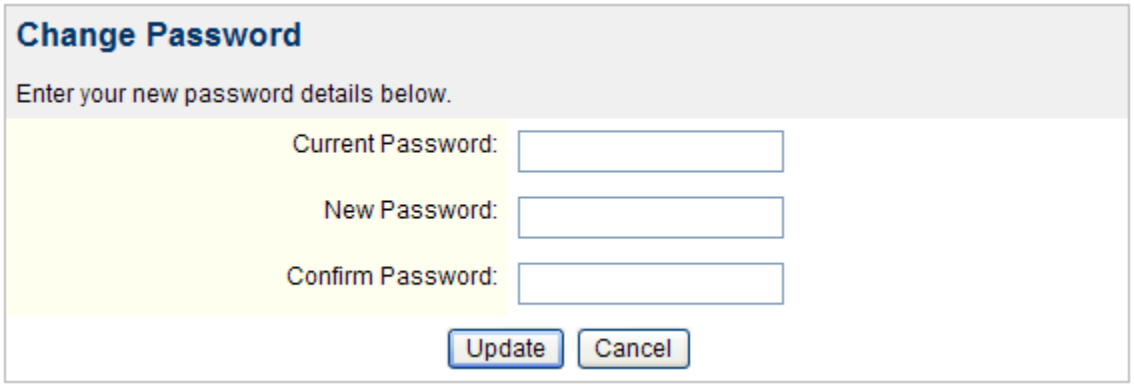
- Click **'Personal Roadmap'** in the left column to view your 'Personal Roadmap Report' (i.e. work assigned to you, across all projects).
- Click **'Your Votes'** in the left column to view the list of issues for which you have voted.
- Click **'Your Watches'** in the left column to view the list of issues that you are watching.
- Click **'Change Password'** in the left column to change your JIRA password.
- Click **'Dashboard Config'** in the left column to customise your JIRA dashboard.
- Click **'Manage Filters'** in the left column to view and edit your issue filters.
- Click **'Navigator Columns'** in the left column to choose which fields appear in your Issue Navigator.
- Click **'View Preferences'** / **'Edit Preferences'** in the left column to view/edit the following:
  - **'Number of Issues displayed per Issue Navigator page'** (the default is 50).
  - **'Outgoing email format'** (i.e. text/HTML).
  - **'Language'** (your preferred language).
  - **'Email me when I make changes'** (i.e. whether you wish to receive email notifications about issue updates that you have made, or only about issue updates made by other people).
  - **'Default sharing for filters and dashboards'** (i.e. whether you wish your filters and dashboards to be shared with all JIRA user by default at creation. You can change the sharing for individual filters and dashboards from the default).
- Click **'View Profile'** / **'Edit Profile'** in the left column to view/edit the following:
  - **'Full Name'** (your display-name, i.e. the name by which you are known in JIRA).
  - **'Email'** (the email address to which your JIRA notifications will be sent).
- Click the links under **'User Filters'** in the right column to view your:
  - **'Assigned'** issues (i.e. the list of issues that are assigned to you).
  - **'In Progress'** issues (i.e. the list of issues that are assigned to you and that currently have a status of 'In Progress').
  - **'Reported'** issues (i.e. the list of issues that were created by you).
- Click the links under **'Assigned Open Issues per project'** in the right column to view the unresolved issues (if any) that are assigned to you for each project.

Note
If your JIRA administrator has configured 'External User Management' or 'External Password Management' (e.g. if you are using your Active Directory or LDAP username to login to JIRA), the **'Edit Profile'** and **'Change Password'** links may not be available.

Changing your Password

To change your JIRA password:

1. Click your user name at the top-right of the page to open your User Profile.
2. Click the **'Change Password'** link at the left of the page.
3. Type your old password into the **'Current Password'** field, and type your new password into the **'New Password'** and **'Confirm Password'** fields:



4. Click the **'Update'** button.

If your JIRA administrator has configured 'External User Management' or 'External Password Management' (e.g. if you are using your Active Directory or LDAP username to login to JIRA), the **'Change Password'** link may not be available.

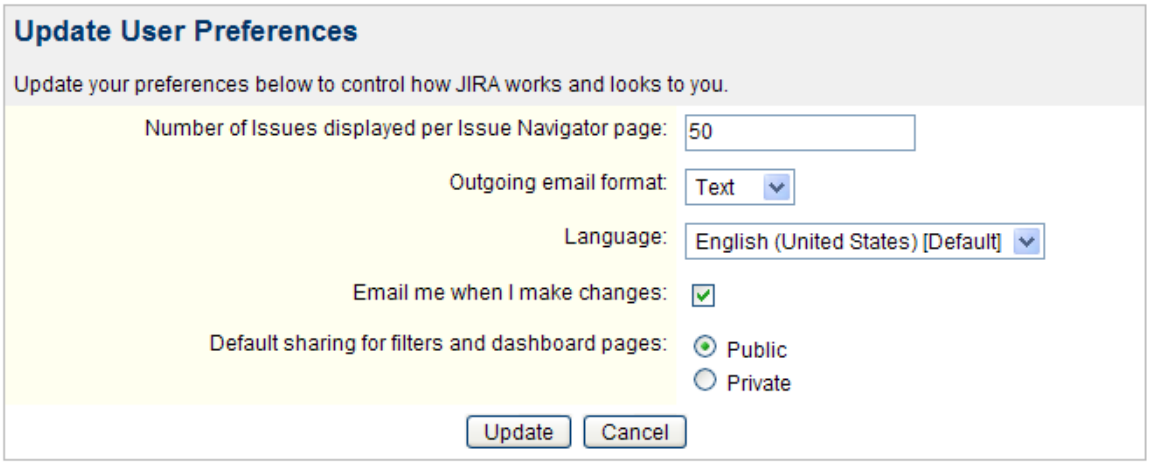
Choosing a Language

You can personalise your JIRA account to use a language of your choice.

To choose a language:

1. Click your user name at the top-right of the page to open your User Profile.

2. Click the **View Preferences** link at the left of the page.
3. Select your language from the "Language" drop down list:



4. Click the **Update** button.

JIRA Administrator's Guide

This manual contains information on administering your JIRA system:

Getting Help

Configuring the Layout and Design

- Customising the Look and Feel
- Choosing a Default Language
- Configuring the Default Issue Navigator
- Configuring the Default Dashboard
    - Using Dashboard Gadgets
    - Adding a Gadget to the Directory
- Configuring an Announcement Banner
- Enabling Logout Confirmation

User and Group Management

- Managing Users
- Managing Groups
- Managing Project Roles
- Migrating User Groups to Project Roles
- Enabling Public Signup and CAPTCHA
- Integrating JIRA with LDAP
- Integrating JIRA with Crowd
- Configuring Trusted Applications

Project Management

- Defining a Project
- Managing Project Role Membership
- Defining a Component
- Managing Versions
    - Creating Release Notes
- Configuring Project Keys

Configuring Security

- Managing Global Permissions
- Managing Project Permissions
- Configuring Issue Level Security

Configuring Fields and Screens

- Configuring Built-in Fields
  - Defining 'Issue Type' Field Values
    - Associating Issue Types with Projects
  - Defining 'Priority' Field Values
  - Defining 'Resolution' Field Values
  - Defining 'Status' Field Values
  - Translating Resolutions, Priorities, Statuses and Issue Types
- Adding a Custom Field
  - Creating Help for a Custom Field
  - Configuring a Custom Field
- Specifying Field Behaviour
  - Associating Field Behaviour with Issue Types
  - Configuring Rich-Text Renderers
- Defining a Screen
  - Associating a Screen with an Issue Operation
  - Associating a Screen with an Issue Type

Configuring Workflow

- Activating Workflow
- Adding a Custom Event

Configuring Email Notifications

- Creating a Notification Scheme
- Customising Email Content
- Creating Issues and Comments from Email
- Configuring JIRA to Send SMTP Mail

Migrating from Other Issue Trackers

- Importing Data From CSV
- Importing Data from Bugzilla
  - Migrating from Bugzilla 3.0.3
  - Modifying the Bugzilla Importer
- Importing Data From Mantis
- Importing Data From FogBugz

Moving or Archiving Individual Projects

- Archiving a Project
- Splitting a JIRA instance

Integrating with a Source Control System

- Integrating JIRA with CVS and ViewCVS
- Integrating JIRA with Subversion
- Integrating JIRA with Perforce
- Integrating JIRA with ClearCase
- Integrating JIRA with FishEye

Configuring Global Settings

- Configuring JIRA Options
- Setting properties and options on startup
- Advanced JIRA configuration with jira-application.properties
- Enabling File Attachments
  - Enabling Thumbnails for Attachments
- Enabling Sub-tasks
- Enabling Issue Linking
- Enabling Trackback
- Enabling Time Tracking
- Configuring OAuth Consumers

Server Administration

- Search Indexing
  - Re-Indexing after Major Configuration Changes
- Backing Up Data
  - Automating JIRA Backups
- Restoring Data

    - Restoring a Project from Backup
- Using the Database Integrity Checker
- Optimising Performance
- Precompiling JSP pages
- Logging and Profiling
- Increasing JIRA Memory
- Database Indexing
- Generating a Thread Dump
- Using robots.txt to hide from Search Engines

Appendix A - Extending JIRA

- Managing JIRA's Plugins
- Listeners
- Services
- Jelly Tags
- JIRA Toolkit (Customer Support Extensions)
- Developer Guides
- Building JIRA from Source
- API Documentation

Please see the *JIRA User's Guide* for an introduction to the concepts of issues and projects.

Getting Help

If you encounter any problems using or setting up JIRA, please let us know — we're here to help!

You may want to first search the following:

- the JIRA mailing list forums, where Atlassian staff and JIRA users can answer your questions.
- the JIRA Administrators FAQ.

If you need further assistance, please raise a support request (see below).

Alternatively, if you feel you have encountered a bug in JIRA, or wish to request a feature, please file an issue. It is a good idea to first scan JIRA's Popular Issues — this helps to prevent duplicates.

Looking for more helpful information? You can receive news, product information and code tips via our newsletter, blogs and forums.
Stay in touch with us here.

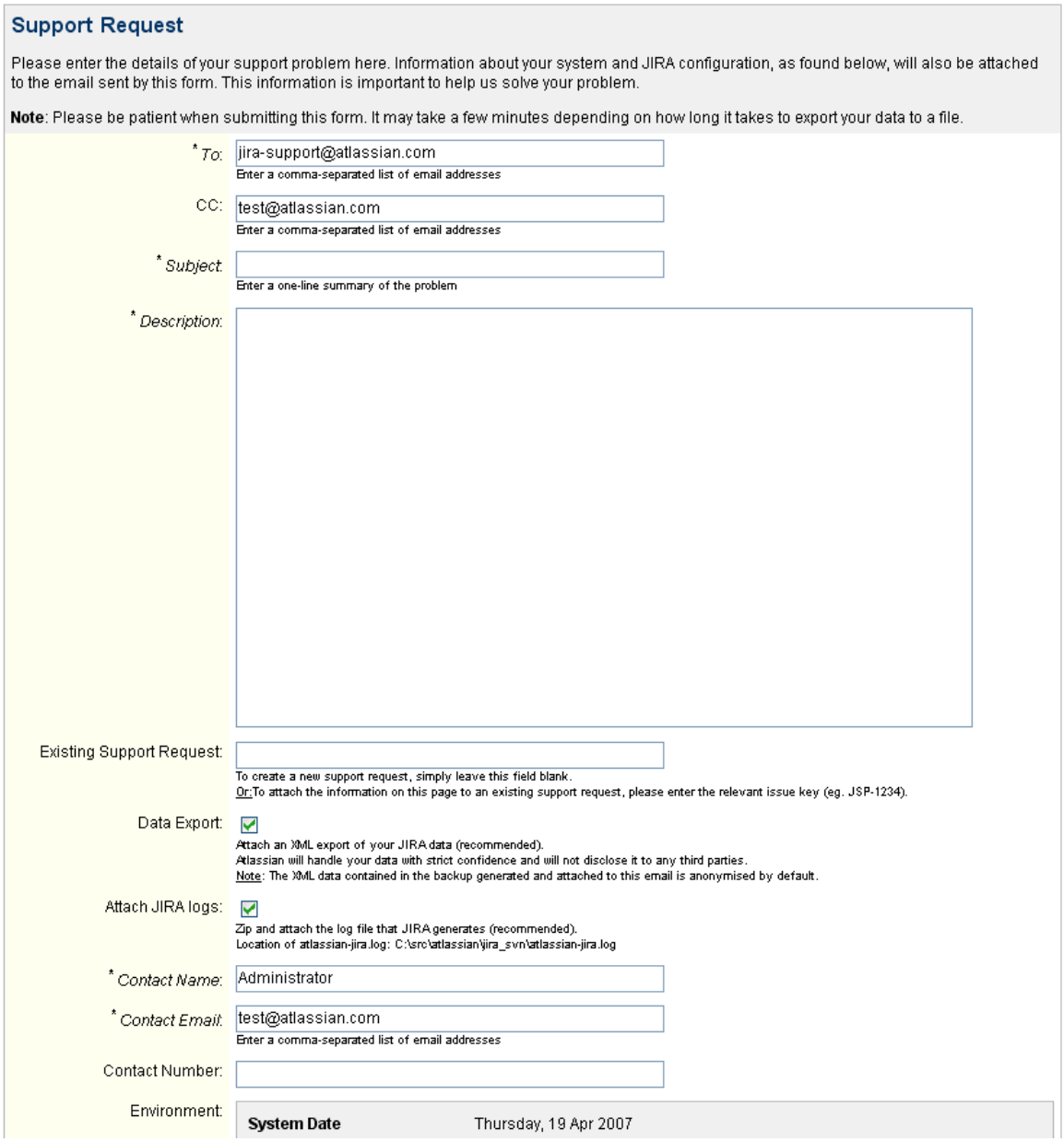
Raising a Support Request

You can raise a support request either in JIRA or via the internet, as described below:

To raise a support request via your JIRA system

(recommended, provided your SMTP email is enabled)

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'System', click the '**Support Request**' link.
4. The 'Support Request' form will be displayed:
    - Please provide as much information as possible, including any error messages that are appearing on the console or via log4j.
    - Please select '**Data Export**' and '**Attach JIRA logs**'.
    - If you have previously raised a support request for the problem, please type the issue key (e.g. **JSP-1234** into the 'Existing Support Request' field.



5. Once you have submitted your support request, you will receive email updates about its progress. You can also view the status of your support request by visiting the Atlassian Support System

OR:

To raise a support request via the internet

1. Please visit the Atlassian Support System and create a support request.
2. Please provide as much information as possible, including any error messages that are appearing on the console or via log4j. Please also mention the operating system, database and version of JIRA you are using.

Sometimes it is necessary to adjust JIRA's logging levels to get a more detailed error message or a stack trace. Please see the logging section of the documentation for information on how to do this.

Configuring the Layout and Design

The following pages contain information on configuring the layout and design of JIRA:

- Customising the Look and Feel
- Choosing a Default Language

- Configuring the Default Issue Navigator
- Configuring the Default Dashboard
- Configuring an Announcement Banner
- Enabling Logout Confirmation

You may also wish to extend JIRA's functionality by installing and/or enabling new plugins. Read the Managing JIRA's Plugins documentation for further information.

Customising the Look and Feel

You can easily customise JIRA's look and feel to suit your needs:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Global Settings'**, click **'Look and Feel'**.
4. The **'Look and Feel Configuration'** page will be displayed as follows:

Logo

URL	<default>
Preview	<N/A>
Logo Width	<default>
Logo Height	<default>

Colours

Header Background Colour	<default>
Header Text Colour	<default>
Header Highlight Background Colour	<default>
Header Text Highlight Colour	<default>
Header Separator Color	<default>
Header Menu Indicator Image	For dark backgrounds
Navigation Bar Background Colour	<default>
Navigation Bar Text Colour	<default>
Navigation Bar Highlight Background Colour	<default>
Navigation Bar Text Highlight Colour	<default>
Link Colour	<default>
Link Active Colour	<default>
Heading Colour	<default>

Gadget Colours

Colour 1	<default>
Colour 2	<default>
Colour 3	<default>
Colour 4	<default>
Colour 5	<default>
Colour 6	<default>
Colour 7	<default>
Colour 8	<default>

Date/Time Formats

	Format	Example
Time Format	hh:mm a	10:18 AM
Day Format	EEEE hh:mm a	Monday 10:18 AM
Complete Date/Time Format	dd/MMM/yy hh:mm a	25/May/09 10:18 AM
Day/Month/Year Format	dd/MMM/yy	25/May/09
Date Picker Format (javascript format)	d/MMM/yy (%e/%b/%y)	25/May/09
Date Time Picker Format (javascript format)	dd/MMM/yy hh:mm a (%e/%b/%y %I:%M %p)	25/May/09 10:18 AM

Edit Configuration

5. To edit the logo, colours or time format, click the **'Edit Configuration'** link at the bottom of the page. To reset to the default look and feel

of JIRA, simply click the **'Reset Default'** button.

Here is a list of the different configuration options available, and what they do.

Logo

The logo appears in the top left corner of every JIRA page. You can easily replace the default JIRA logo with an image of your choice.

Option	Explanation
URL	This URL points to the absolute or relative path of the image that you wish to display at the top of the page. If the URL begins with **'http://'** or **'https://'** then the URL is treated as an absolute URL. Otherwise it will be treated as a relative URL, and the image will have to be packaged in the war file when you build JIRA.
Preview	When an image is selected, a preview will be shown here.
Logo Width	The width of the image, usually in pixels. You can use any format that is valid in an image tag. e.g. *'100px', '80%'*
Logo Height	The height of the image, usually in pixels. You can use any format that is valid in an image tag. e.g. *'25px', '30%'*

Colours

The following options control the appearance of the entire JIRA user interface.

Please note:

- The colours you specify for each of the following options can be anything that is valid for both a font tag, and a stylesheet's 'color:' attribute.
- When specifying a colour, you can use the pop-up colour chooser, or specify your own (eg. *'#FFFFFF', 'red').*
- To return to the original colour scheme, just clear any values that you have set.

Option	Explanation
Header Background Colour	The background colour of the top bar (the one that includes the image).
Header Text Colour	The colour of the text that sits inside the top bar (such as your user name when you are logged in).
Header Highlight Background Colour	The background colour of the text that sits inside the top bar, when selected or when the mouse hovers over it.
Header Text Highlight Colour	The colour of the text that sits inside the top bar, when selected or when the mouse hovers over it.
Header Separator Colour	The colour of the horizontal line between the top bar and the navigation bar.
Header Menu Indicator Image	The colour of the 'drop-down' image that sits inside the top bar. Two options are available: 'For dark backgrounds' and 'For light backgrounds'.
Navigation Bar Background Colour	The background colour of the bar that contains the links to 'HOME' and 'BROWSE PROJECT'.
Navigation Bar Text Colour	The text color of the links in the menu bar (such as 'HOME').
Navigation Bar Highlight Background Colour	The background colour of the navigation bar text, when selected or when the mouse hovers over it.
Menu Bar Text Highlight Colour	The colour of the navigation bar text, when selected or when the mouse hovers over it.
Link Colour	The colour of the text links.
Link Active Colour	The colour of the text links, when selected.
Heading Colour	The colour of the text headings (such as 'Logo').

Gadget Colours

These eight colours are the eight options from which users can select when changing the colour of a gadget's frame on their JIRA dashboard. Colour 4 is the default frame colour for newly-added gadgets.

Please note:

- The colours you specify for each of the eight options can be anything that is valid for both a font tag, and a stylesheet's 'color:' attribute.
- When specifying a colour, you can use the pop-up colour chooser, or specify your own (eg. '#FFFFFF', 'red').
- To return to the original colour scheme, just clear any values that you have set.

Date/Time Formats

These time and date formats are used throughout JIRA. To specify them, use the format described at http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html.

When you are not in edit mode, the examples in the rightmost column show you how the various formats will appear.

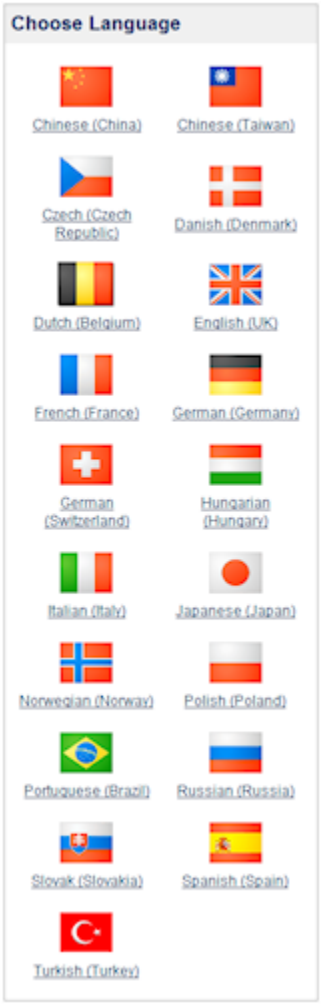
Note: for information about the **Date Picker Format** and **Date Time Picker Format,** please see 'Advanced JIRA configuration with jira-application.properties'.

Choosing a Default Language

Overview

Most user-visible pages in JIRA are now internationalised. Chinese, Czech, Danish, English, French, German, Italian, Norweigian, Polish, Portuguese (Brazilian), Russian, Slovak and Spanish translations are available (at time of writing), with more in development.

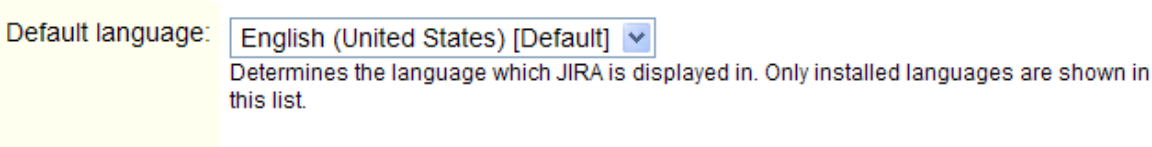
When JIRA is first installed, the default language may be chosen by clicking on a flag:



Changing the default language

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the

dashboard.
3. Click '**General Configuration**' (under the 'Global Settings' subheading on the left).
4. Click '**Edit Configuration**', then select the appropriate language in the drop-down box next to 'Default language'.



Any additional languages you have installed will appear in the list.

Per-user language selection

Individual users can choose their own language, which will override the default language (see above).

Overriding the default translations of Issue Types, Resolutions, Statuses and Priorities

Should you wish, you can easily specify your own translations for the values of the following JIRA issue fields:

- Issue Type
- Priority
- Status
- Resolution

Your specified translations will override the values specified in the JIRA translation.

Configuring the Default Issue Navigator

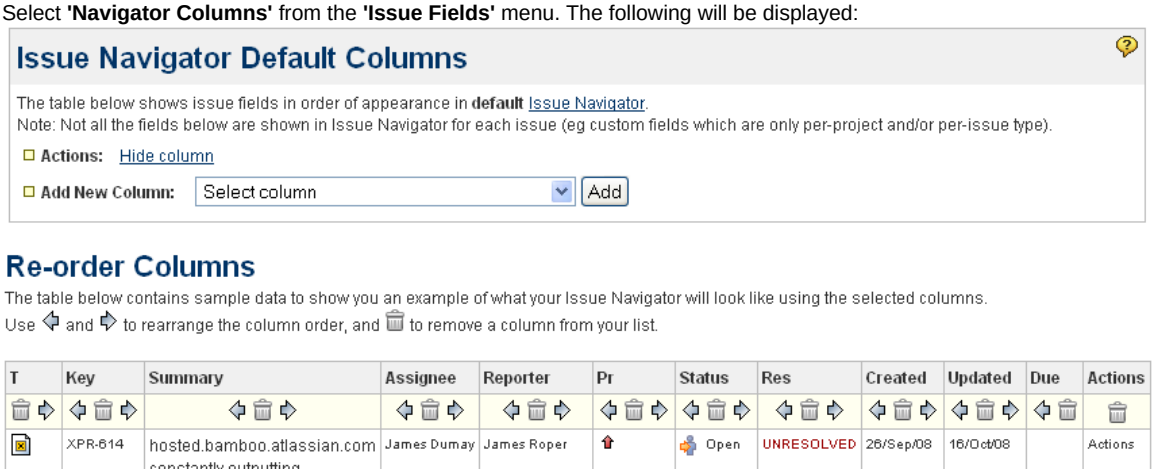
The Issue Navigator is used within JIRA to find and filter issues, and to display the search results in various formats ('views'). It is possible to select which issue fields will be displayed as columns in the Issue Navigator.

JIRA **administrators** can configure which columns appear in the Issue Navigator by default, for all users that do not have their personal navigator columns configured. Each authenticated JIRA **user** can override these defaults by configuring their own Issue Navigator columns to fit their needs. Note that only users who can see at least one issue in the JIRA system are able to configure Issue Navigator columns.

JIRA administrators can also select which views are available in the JIRA system, as views are configurable via plugins.

Configuring the default Issue Navigator Columns

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Select **'Navigator Columns'** from the **'Issue Fields'** menu. The following will be displayed:



- **To move a column left or right,** click on the left-arrow or right-arrow icon that appears under the column's heading.
- **To remove a column from the list,** click the bin icon which appears under the column's heading.
- **To add a column to the list,** select the issue field name from the drop-down box titled **'Add New Column'** and click the **'Add'** button. The column will appear as the right-most column in the list. You can then position the column where desired by using the arrow icons.

- **To hide the *'Actions'** column,* click the **'Hide Column'** link.
- If the column order has been modified from the defaults, users can restore the global defaults by clicking the **'Restore Defaults'** link (which will appear only if they have modified their Issue Navigator from the global defaults). When configuring the global defaults (only available to administrators), the link is called **'Restore System Defaults',** and when clicked restores the configuration that JIRA ships with by default.

Note:

- When configuring their personal Issue Navigator columns, a user can only see columns for issue fields that have not been hidden.
- It is possible to add any of the existing [custom fields] to the Issue Navigator column list. When configuring the columns a user can choose any custom field that they have permissions to see. That is, any custom field except those that are project-specific and apply only to a project that the user does not have permissions to browse. Some custom fields, even if selected as Issue Navigator columns, will not appear in the Issue Navigator for all issues. For example, project-specific custom fields will be shown only if the filter has been restricted to that project only. Issue type custom fields will only appear if the filter has been restricted to that issue type.
- When administrators are configuring default Issue Navigator columns, their permissions are ignored, so that they can add a project-specific custom field from a project that they do not have permissions to browse. The field would never be actually shown to users that do not have permissions to see it.

Configuring the Default Dashboard

The default dashboard is the screen that all JIRA users see the first time they login. Any users who have not added any dashboard pages as favourites also see the default dashboard.

JIRA allows Administrators to configure the default dashboard. The gadgets on the default dashboard can be re-ordered, switched between the left and right columns, additional gadgets can be added, and some gadgets can be configured.

All changes made to the default dashboard will also change the dashboards of all users currently using the default. However, gadgets that users do not have permissions to see will not be displayed to them. For example, the 'Administration' gadget, although it may exist in the default dashboard configuration, will not be visible to non-admin users.

Gadgets are the information boxes on the Dashboard. JIRA comes pre-configured with a set of standard dashboard gadgets. It is also possible to develop custom gadgets and plug them into JIRA using its flexible plugin system.

Adding and Configuring Portlets

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the 'Default Dashboard' link under 'Global Settings' on the left pane.
4. This will display the 'Configure Default Dashboard' screen, which consists of two selectable areas listing the current portlets.
5. If you wish to move and/or configure the existing portlets on the default dashboard, please see the instructions on using the configuration interface of a dashboard.
6. If you wish to add portlets to the default dashboard, please see the instructions on adding a portlet to a dashboard.

JIRA's default dashboard is limited to only one dashboard page. However, users can add multiple pages to their own dashboards if they wish.

By default, there is a limit of 20 gadgets per dashboard page. If you wish to raise this limit, edit the jira-application.properties file and set 'jira.dashboard.max.gadgets=20' to your preferred value. Then restart JIRA.

Using Dashboard Gadgets

On this page:

- About gadgets
- Preinstalled gadgets
- Extension gadgets
- Creating new gadgets

About gadgets

JIRA provides the ability to display summary information about project/issue data on the dashboard, through the use of 'gadgets'. Each gadget can be configured to display project and issue details relevant to particular users. Gadgets can be added to the dashboard — providing a central location for quick access to this information.

Preinstalled gadgets

JIRA provides a set of standard gadgets out-of-the-box:

Gadget	Description
Administration Gadget	The Administration gadget displays quick links to administrative functions conveniently on the dashboard.
Assigned To Me Gadget	The Assigned To Me gadget displays all open issues in all projects assigned to the current user viewing the dashboard.
Average Age Gadget	The Average Age gadget displays a bar chart showing the average number of days that issues have been unresolved.
Bamboo Plan Summary Gadget *	The Bamboo Plan Summary gadget displays a graphical summary of a build plan.
Bamboo Status Gadget *	The Bamboo Status gadget displays a list of all plans on the Bamboo server, and each plan's current status.
Bugzilla ID Search Gadget	The Bugzilla ID Search gadget allows the user to search all JIRA issues for references to Bugzilla IDs.
Calendar Gadget *	The Calendar gadget shows issues and versions in a calendar format based on their due date. Calendars can be based on an issue filter or on a project.
Created vs Resolved Gadget	The Created vs Resolved gadget displays a difference chart showing the issues created vs resolved over a given period.
Favourite Filters Gadget	The Favourite Filters gadget displays a list of all the issue filters that have currently been added by you as a favourite filter.
Filter Statistics Gadget	The Filter Statistics gadget displays the collection of issues returned from a specified filter broken down by a specified field.
2D Filter Statistics Gadget	The Two Dimensional Filter Statistics gadget displays statistical data based on a specified filter in a configurable table format.
In Progress Gadget	The In Progress gadget displays all issues that are currently in progress and assigned to the current user viewing the dashboard.
Introduction Gadget	The Introduction gadget displays a configurable introduction message on the dashboard.
Pie Chart Gadget	The Pie Chart gadget displays displays issues from a project or issue filter, grouped by a statistic type, in pie-chart format. The issues can be grouped by any statistic type (e.g. Status, Priority, Assignee, etc).
Project Gadget	The Project gadget provides information and various filters related to a specified project on the dashboard.
Projects Gadget	The Projects gadget provides information and various filters related to all projects within JIRA.
Project Statistics Gadget	The Project Statistics gadget allows various per-project statistical data to be displayed on the dashboard.
Project Table Gadget	The Project Table gadget displays all the project names in a table in the dashboard.
Quick Links Gadget	The Quick Links gadget displays a number of useful links to issues associated with the current user. Each link directs the user to the Issue Navigator, displaying the relevant issues such as Reported Issues, Voted Issues and Watched Issues.
Recently Created Issues Gadget	The Recently Created Issues gadget displays a bar chart showing the rate at which issues are being created, as well as how many of those created issues are resolved.
Resolution Time Gadget	The Resolution Time gadget displays a bar chart showing the average resolution time (in days) of resolved issues.
Road Map Gadget	The Road Map gadget shows versions which are due for release within a specified period of time, and a summary of progress made towards completing the issues in those versions.

Saved Filter Gadget	The Saved Filter gadget displays the results of a specified issue filter on the dashboard.
Text Gadget	The Text gadget displays a configurable HTML text on the dashboard.
Time Since Issues Gadget	The Time Since Issues gadget displays a bar chart showing the number of issues for which your chosen date field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a custom field) was set on a given date.
Voted Gadget	The Voted Issues gadget shows issues for which you have voted.
Watched Gadget	The Watched Issues gadget shows issues which you are watching.

*This gadget will only be available if you have enabled it.

Extension gadgets

Other gadgets are available as plugins on the JIRA Extensions site. These plugins include:

- Bamboo plugin
- Calendar plugin
- Timesheet plugin

Should you wish to you use these plugins, you need to first install them (using the instructions provided with each plugin) then enable them.

Creating new gadgets

New gadgets can be created by writing an XML descriptor file, packaged as an Atlassian plugin. See Writing an Atlassian Gadget for more information.

Adding a Gadget to the Directory

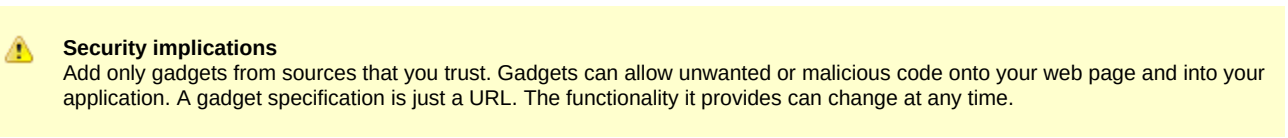
On this page:

- Overview
- Adding a Gadget that is Not a Plugin
- Adding a Gadget that must be Installed as a Plugin

Overview

The JIRA gadget directory displays all the gadgets that are available for JIRA users to add to their dashboard.

You need to have administrator privileges to add a gadget to the directory. If you have permission to add gadgets to and remove gadgets from the directory itself, you will see the '**Add Gadget to Directory**' and '**Remove**' buttons on the 'Add Gadget' screen, as shown below.



There are two types of gadgets: those that must be installed as plugins, and those that can be added as simple gadget URLs.

Adding a Gadget that is Not a Plugin

If the gadget is hosted on another server and can be added to the directory as a simple URL, then you can simply add it via your dashboard's 'Add Gadget' option.

To add a gadget to your directory,

1. First you need to find the URL for the gadget's XML specification file. A gadget's URL looks something like this:

   ```
   http://example.com/my-gadget-location/my-gadget.xml
   ```

   Gadget authors and publishers make their gadget URLs available in different ways. Below are the instructions for a Google gadget and an Atlassian gadget.

   - Follow the steps below to find the URL for a Google gadget:
     a. Go to the Google gadget directory. (You can also get there by clicking 'Add Stuff' from your iGoogle home page.)
     b. Search for the gadget you want.
     c. Click the link on the gadget to open its home page.
     d. Find the '**View source**' link near the bottom right of the page. Right-click the link and copy its location to your clipboard. This is the gadget's URL.

   - Follow the steps below to find the URL for a gadget that is published by an Atlassian application, such as JIRA or Confluence:
     a. Open the host application, such as JIRA.
        Currently, only JIRA 4.0 has a dashboard that supports gadgets.
     b. Go to the host application's dashboard by clicking the '**Dashboard**' link or the '**Home**' link at the top left of the screen.
     c. Click '**Add Gadget**' to see the list of gadgets in the host application's directory. The list will look something like screenshot 1 below.
     d. Find the gadget you want, then copy the location of the '**Gadget URL**'.
        The URL will have this format:

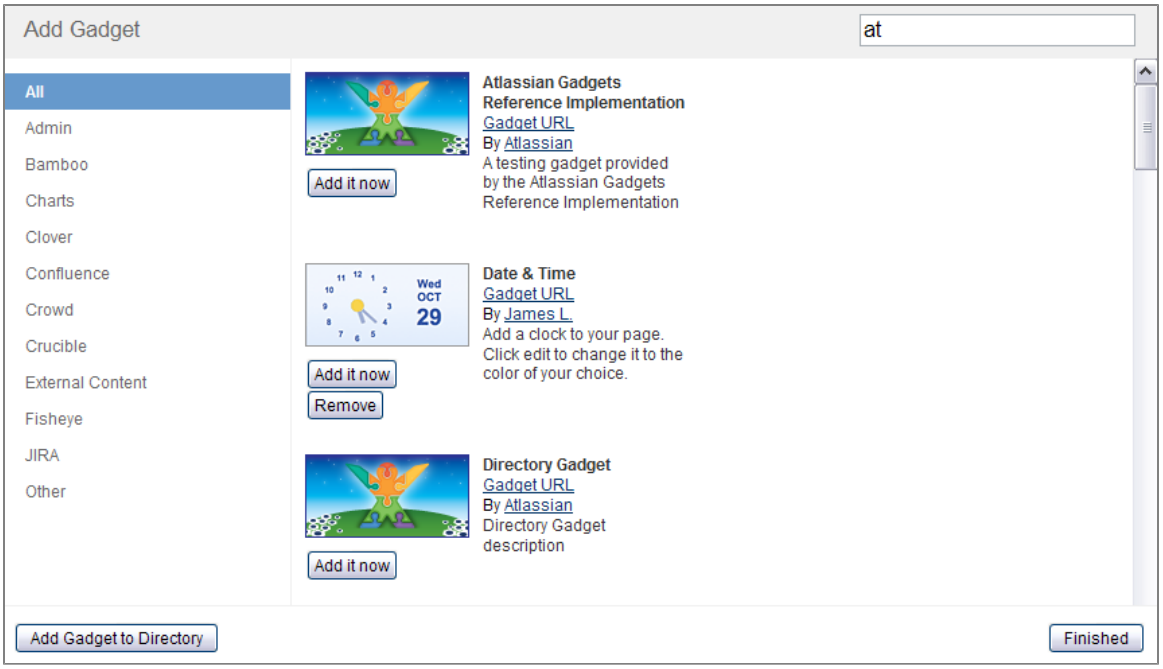
        ```
        http://my-app.my-server.com:port/rest
        /gadgets/latest/g/my-plugin.key/my-path/my-gadget.xml
        ```

        For example:

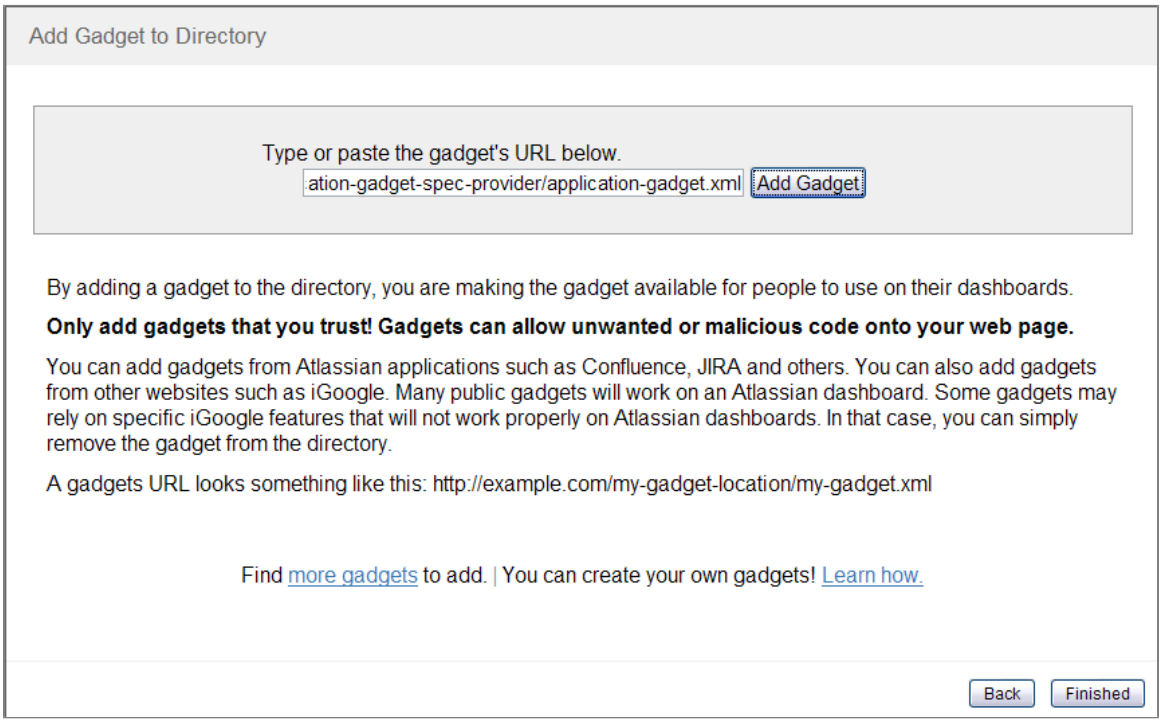
        ```
        http://mycompany.com/jira/rest
        /gadgets/latest/g/com.atlassian.streams.streams-jira-plugin/gadgets/activi
        ```

2. Now you can add the gadget to your directory. Go to the dashboard by clicking the '**Dashboard**' link or the '**Home**' link at the top left of the screen.
3. The dashboard will appear. Click '**Add Gadget**'.
4. The '**Add Gadget**' screen appears, showing the list of gadgets in your directory. See screenshot 1 below. Click '**Add Gadget to Directory**'.
   You will only see this button if you have administrator permissions for your dashboard.
5. The '**Add Gadget to Directory**' screen appears. See screenshot 2 below. Type or paste the gadget URL into the text box.
6. Click '**Add Gadget**'.
7. The gadget appears in your gadget directory. (It will be highlighted for a short time, so that you can see it easily.)

Screenshot 1: Gadget directory with 'Add Gadget to Directory' button



Screenshot 2: Adding a gadget to the directory



Adding a Gadget that must be Installed as a Plugin

If the gadget must be installed as a plugin, you cannot add it via the gadget directory user interface.

Instead, you will need to follow the instructions for adding a plugin, as described in Managing JIRA's Plugins.

Configuring an Announcement Banner

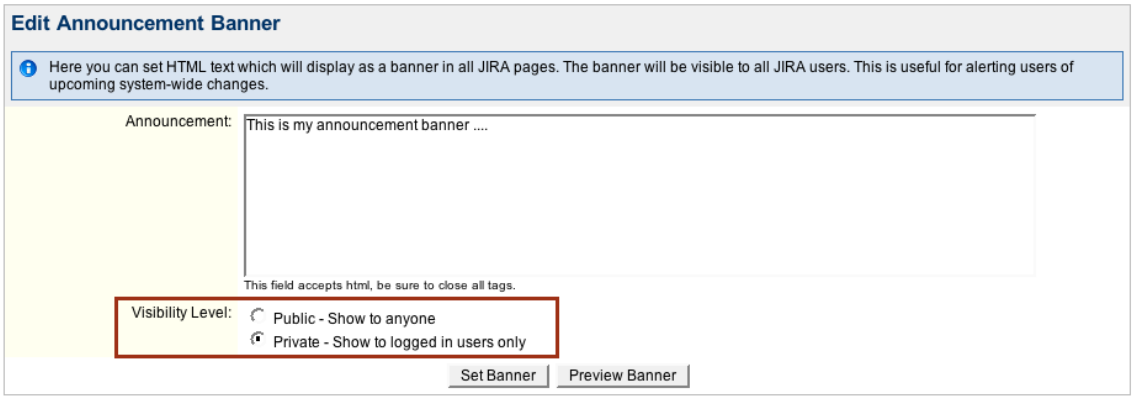
Administrators can configure an **announcement banner** to display pertinent information on all JIRA pages. The banner can be used to relate important information (e.g. scheduled server maintenance, approaching project deadlines, etc.) to all users. Further, the banner visibility level can be configured to display to all users or just logged-in users.

The banner can be configured to contain HTML text.

Configure Announcement Banner

1. Navigate to the JIRA **Administration** section.
2. Select the **Announcement Banner** under the **Options & Settings** sub-menu.
3. Enter the required text in the **Announcement** field.
4. Select the required visibility level for the banner.
5. Click **Set Banner.**

Depending on the visibilty level selected, the banner will become visible throughout JIRA.



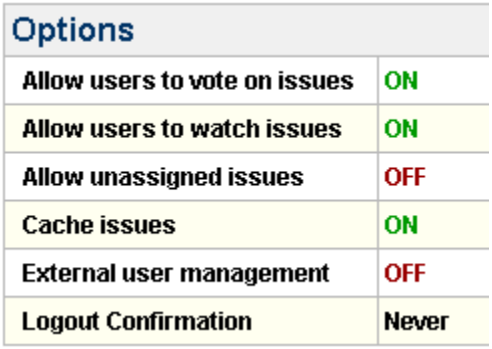
Banner Visibility Mode

The announcement banner visibility level can be configured to specify to whom the banner will be displayed. There are two modes:

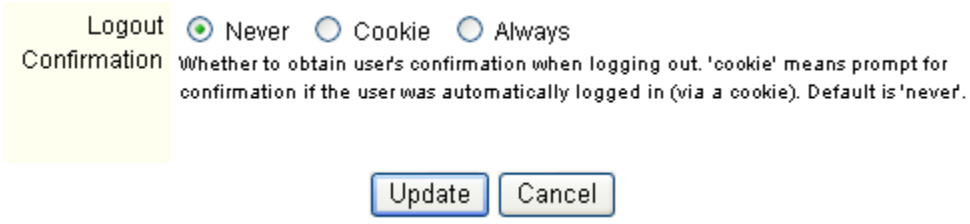
- **Public** — the banner is visible to everyone
- **Private** — the banner is visible to logged-in users **only**

Enabling Logout Confirmation

Administrators can configure JIRA to prompt users with a confirmation before logging them out. You can adjust this setting by going to the Administration page and then clicking the '**General Configuration**' link found under '**Options and Settings**' on the left pane.



As shown above, JIRA will never prompt users for logout confirmation by default. To change this, click on '**Edit Configuration**'.



The "Never" and "Always" settings are self-explanatory. When set to "Cookie", users will only be prompted if they have logged in using a cookie (i.e. checked the box reading 'Remember my login on this computer' before they logged on).

User and Group Management

The following pages contain information about user and group management in JIRA:

- Managing Users
- Managing Groups
- Managing Project Roles
- Migrating User Groups to Project Roles
- Enabling Public Signup and CAPTCHA
- Integrating JIRA with LDAP
- Integrating JIRA with Crowd
- Configuring Trusted Applications

Managing Users

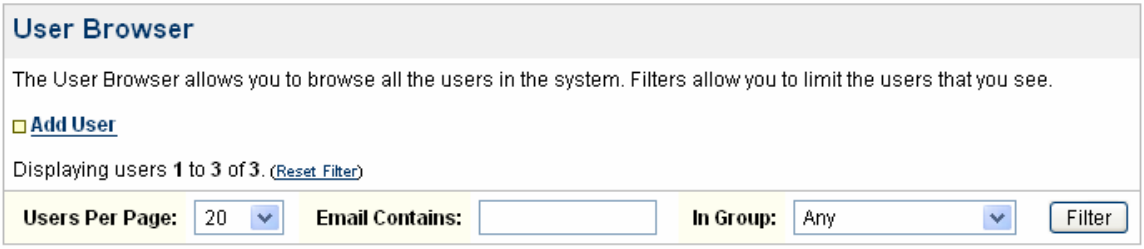
On this page:

- Viewing Users
- Adding a User
- Assigning a User to a Group
- Assigning a User to a Project Role
- Changing a User's Name or Email Address
- Changing a User's Password
- Adding a Property to a User
- Deleting a User

Viewing Users

To view a list of JIRA users:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Select **User Browser** from the **Users, Groups & Roles** section of the administration menu. This will display the **User Browser** screen:



Username	Email	Full Name	Groups	Operations
admin	admin@mycompany.com	Administrator	jira-administrators jira-developers jira-users	Groups | Project Roles | Edit | Delete
marym	marym@mycompany.com	Mary Manager	XYZ Developers jira-users	Groups | Project Roles | Edit | Delete
sallyu	sallyu@mycompany.com	Sally User	jira-users XYZ Developers	Groups | Project Roles | Edit | Delete

4. To restrict the list of users shown in the **User Browser,** use the Filter form at the top of the User Browser. Specifying (part of) the user's email and/or group, then clicking the **Filter** button, will reduce the list to only those users who match those criteria.

Adding a User

1. Open the **User Browser** (see 'Viewing Users' above) and click the **Add User** link.
2. This will display the **Create New User** form. Enter the **Username** (note that a user's **Username** cannot be changed once the user is created), **Password, Full Name** and **Email Address;** and (optionally) tick the box to send the user an email containing their account details. Then click the **Create** button.

Note
Users can also be created via:

- **Signup** - see " Enabling Public Signup "
- **Email** - e.g. you can use the CreateIssueHandler (see " Services ") to have JIRA create a user based on the sender's email address.

Note
If you have a user limited license (e.g. personal license) and have reached your user limit, any further users added will not have permission to log in to JIRA.

Assigning a User to a Group

When a user is created, they will be added to any groups that are set up to have new users automatically added to them.

To change a user's group membership:

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Groups** link in the **Operations** column.
2. This will display two lists; the one on the left shows all available groups, and the one on the right shows all groups to which the user currently belongs. Use the **Join** and **Leave** buttons to add the user to or remove them from your selected group.

Note
If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to assign any further users to groups with login permissions (i.e. jira-users permission) without first reducing the number of users with login permissions.

Assigning a User to a Project Role

Assigning a user to a project role enables them to fulfil a particular function in a particular project.

To view a user's project role membership, locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Project Roles** link in the **Operations** column. This will display a table showing all the projects and project roles that exist in JIRA, and the user's current project role membership for each project:

!usermanagement-project_role_membership.png|title="User's Project Role Membership"!E.g. this screenshot shows that, for the ABC project:

- Mary is a member of the 'Administrators' project role.
- Mary is not a member of the 'Developers' project role.
- Mary is indirectly a member of the 'Users' project role, through being a member of the 'jira-users' group.
  (Also note that, for the DEF project, May is both a direct *and* an indirect member of the 'Users' project role.)

Click the **Edit Project Roles** button. The check-boxes will then be available for you to tick (to add the user to a project role) or un-tick (to remove the user from a project role).

Changing a User's Name or Email Address

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Edit** link in the **Operations** column.
2. This displays a form where you can change the user's Full Name or Email Address. Click **Update** to confirm the change.

Changing a User's Password

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Username.**
2. This displays the user's details, below which are several links. Click the **Set Password** link.
3. This displays the **Set Password** screen. Enter and confirm the new password; then click the **Update** button.

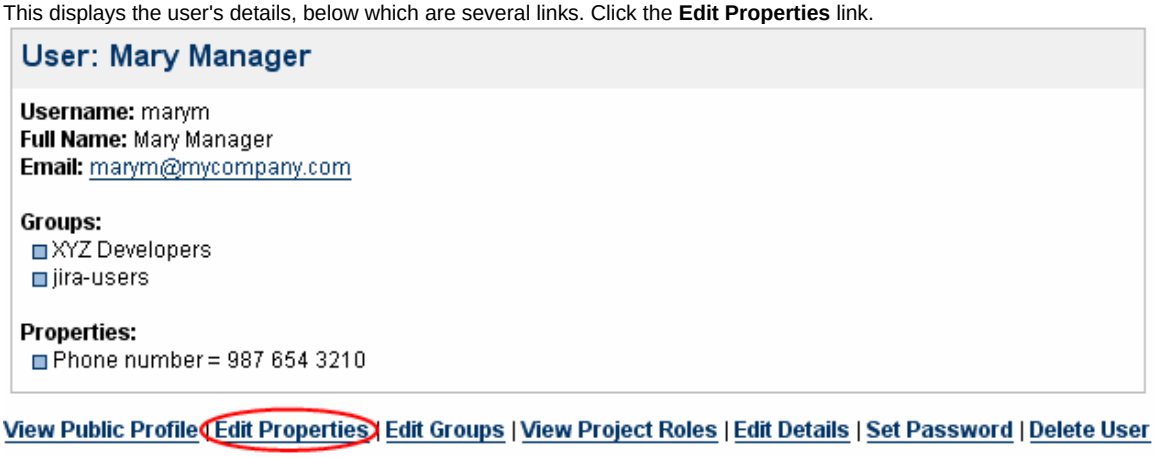
Adding a Property to a User

A **'Property'** is an extra piece of information that you can store regarding a user. A Property consists of a **Key** of your choice (eg. 'Phone number', 'Location') plus a corresponding **Value** (eg. '987 654 3210', 'Level Three').

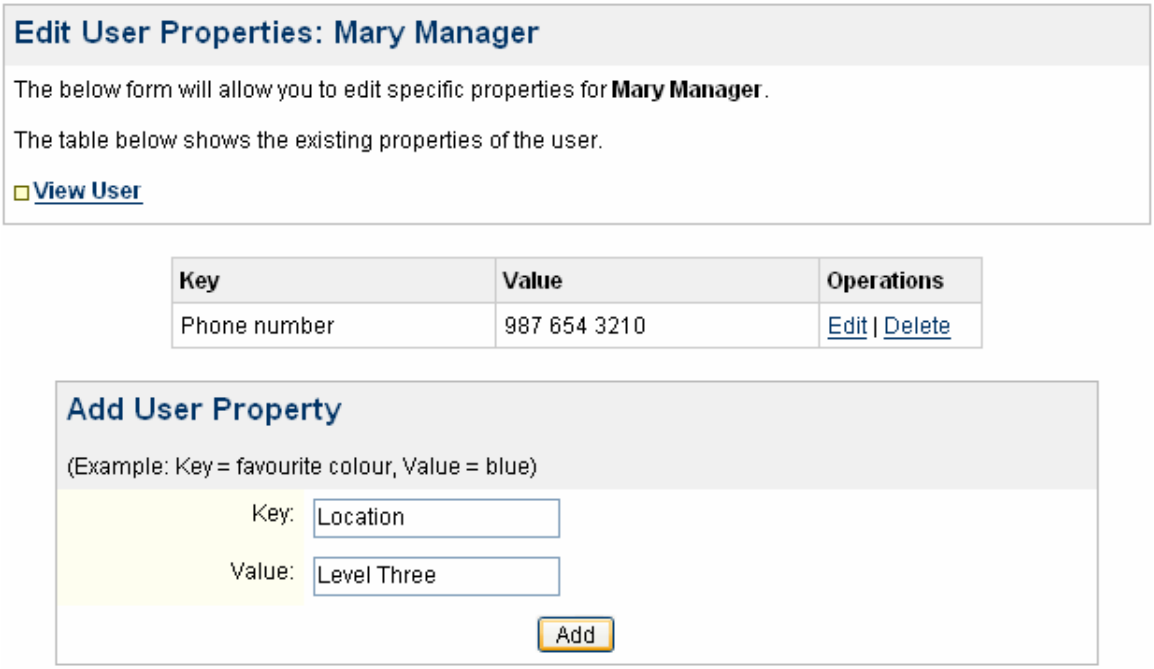
To create a new Property for a user:

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Username.**

2. This displays the user's details, below which are several links. Click the **Edit Properties** link.



3. This displays the **Edit User Properties** screen, showing any previously-created properties:



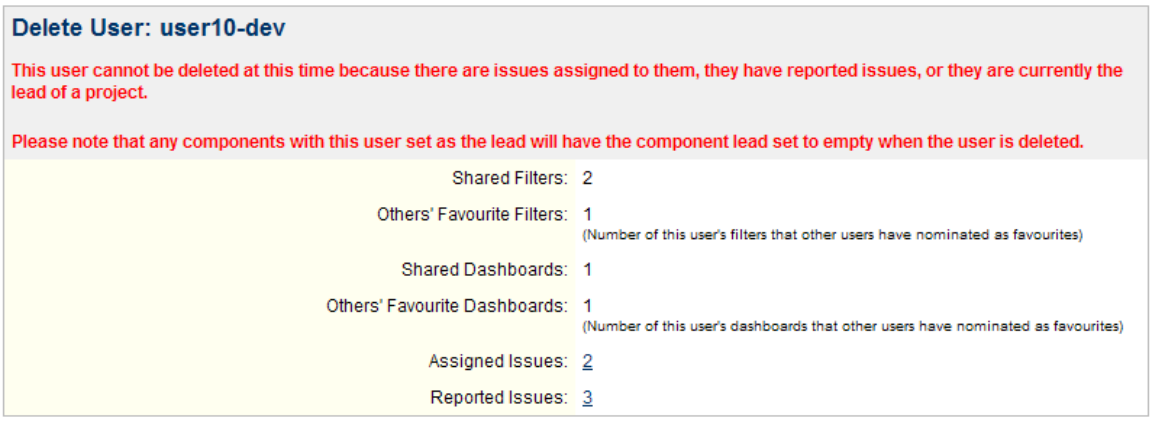
4. Enter the new **Key** and its **Value,** then click the **Add** button.

Deleting a User

> **Note**
> Rather than deleting a user, it is recommended to *disable their account* by removing them from all groups (see 'Assigning a User to a Group', above). This prevents the user's account from being used. It is still important to reassign any issues assigned to that user, but there is no need to modify the 'Reporter' as described below.

To delete a user,

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Delete** link in the **Operations** column.
2. The confirmation screen that follows will summarise any involvement of that user in the system by showing current issues assigned to and reported by that user, etc. These connections between the user and other parts of the system may prevent the deletion of that user. For example, attempting to delete a user called *bob* results in the following screen, which prevents deletion due to the presence of 10 assigned issues:



As well as reassigning any issues, you may need to bulk-edit the issues created by the user and change the 'Reporter' to someone else. You'll need the ' Modify Reporter ' permission to do this.

3. If there are no issues assigned to, or reported by the user, the confirmation screen will display a **Delete** button; click this to proceed with the deletion.

> Please note that the filters and dashboards of a user will be deleted when the user is deleted, regardless of whether the filters or dashboards are shared with other users.

> **Note**
> If you are using External User Management, you will not be able to create, edit or delete users from within JIRA; but you can still assign users to project roles, and create/edit/delete user properties.

Managing Groups

A JIRA group is a convenient way to manage a collection of users. Users can belong to many groups. Groups are used throughout JIRA; for example, they can:

- be granted global permissions.
- be used in project permission schemes.
- be used in email notification schemes.
- be used in issue security levels.
- be given access to issue filters *(in JIRA Professional and Enterprise editions).*
- be given access to dashboards *(in JIRA Professional and Enterprise editions).*
- be used in workflow conditions *(in JIRA Professional and Enterprise editions).*
- belong to project roles *.

** Project roles are somewhat similar to groups, the main difference being that group membership is global whereas project role membership is project-specific.*

On this page:

- JIRA's default groups
- Viewing groups
- Adding a group
- Deleting a group
- Editing group membership
- Automatic group membership

JIRA's default groups

When you install JIRA, three groups are automatically created:

- **jira-administrators** - typically contains people who are JIRA system administrators. By default, this group:
  - is a member of the **'Administrators'** project role.
  - has the **'JIRA Administrators'** and the **'JIRA System Administrators'** global permissions. (Note: if you need to give these permissions to separate people, you will need to create an additional group and grant the permissions separately, as described in *'About 'JIRA System Administrators' and 'JIRA Administrators''.*)
- **jira-developers** - typically contains people who perform work on issues. By default, this group:
  - is a member of the **'Developers'** project role.
  - has the **'Browse Users', 'Create Shared Filter'** and **'Manage Group Filter Subscriptions'** global permissions.
- **jira-users** - typically contains every JIRA user in your system. By default, this group:

- is a member of the **'Users'** project role.
- has the **'JIRA Users'** and **'Bulk Change'** global permissions.

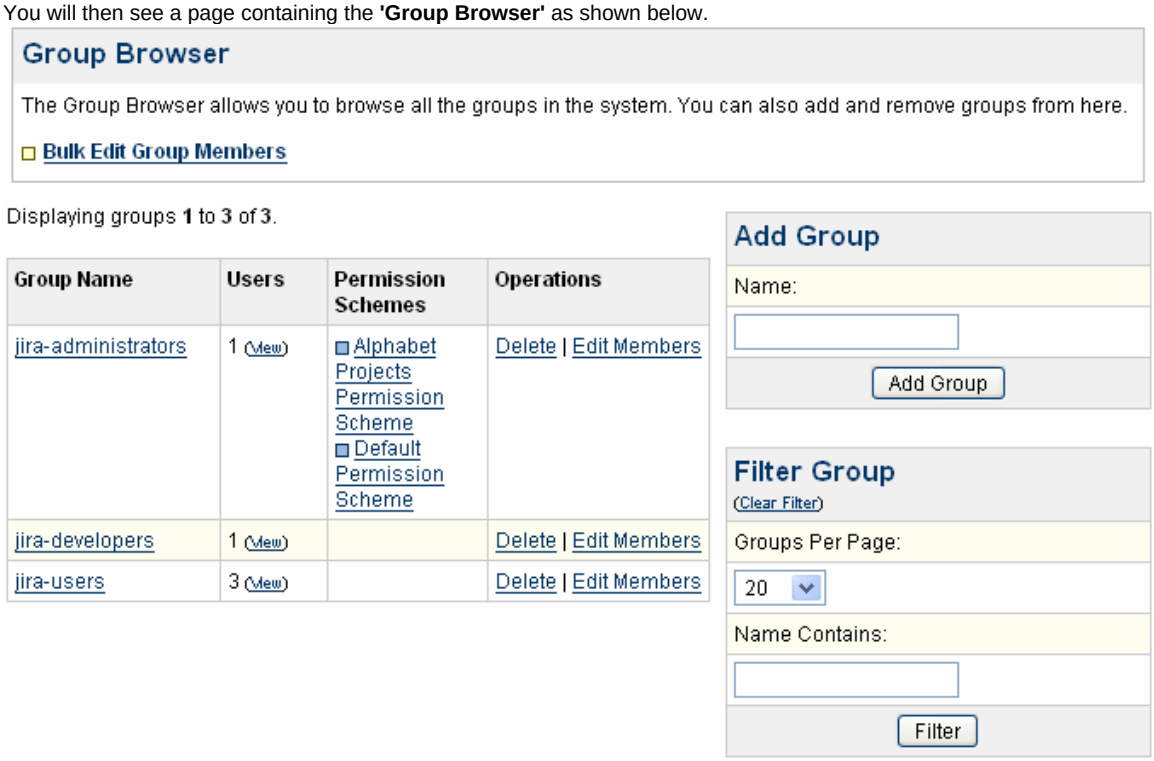
You can create and delete groups according to your organisation's requirements.

> If you are using External User Management, you will not be able to create, delete or edit groups or group membership from within JIRA; and 'Automatic Group Membership' (see below) will not apply. However, you can still assign groups to project roles.

Viewing groups

To see what groups exist, and where they are used:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Select **'Group Browser'** from the **'Users, Groups & Roles'** section of the 'Administration' menu.
4. You will then see a page containing the **'Group Browser'** as shown below.



> The **'Filter Group'** form restricts the list of groups shown to those that match the **'Name Contains',** with a specified maximum per page. Click the **'Filter'** button to refresh the list with the restricting filter.

5. To see which permission schemes, email notification schemes, issue security levels and saved filters are using this group, click the group name.

Adding a group

To create a group, enter the new group **'Name'** in the **'Add Group'** form in the **'Group Browser'** (see *'Viewing groups'* above) and click the **'Add Group'** button.

Deleting a group

To delete a group, click the **'Delete'** link for that group in the **'Group Browser'** (see *'Viewing groups'* above). The confirmation screen that follows explains that users will be removed from the group through its deletion. Be aware of the impact this may have on users in that group. For example, if that group membership was the sole conveyor of a permission for a user, then the user will no longer have that permission.

Before deleting a group it is recommended that you check whether the groups is being used by any permission schemes, email notification schemes, issue security levels or saved filters. See *'Viewing groups'* (above).

Editing group membership

To edit a group's membership, click the **'Edit Members'** link in the row for that group in the **'Group Browser'** (see *'Viewing groups'* above). This takes you to a form allowing you to add users to or remove them from the group.

If the group has the **'JIRA System Administrators'** global permission, you cannot edit its membership unless you have the **'JIRA System Administrators'** global permission.

If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to assign any further users to groups with login permissions (i.e. jira-users permission) without first reducing the number of users with login permissions.

Automatic group membership

To automatically add newly-created users to a particular group, grant the group the **'JIRA Users'** global permission.

To do this, navigate to the **'Administration'** section and select **'Global Permissions'** from the **'Global Settings'** menu. Add the **'JIRA Users'** permission to the relevant group, as described in *'Granting global permissions'*.

JIRA Permissions	
JIRA System Administrators Ability to perform all administration functions. There must be at least one group with this permission. Note: People with this permission can always log in to JIRA.	jira-administrators (View Users | Delete)
JIRA Administrators Ability to perform most administration functions (excluding Import & Export, SMTP Configuration, etc.). Note: People with this permission can always log in to JIRA.	jira-administrators (View Users | Delete)
JIRA Users Ability to login to JIRA. They are a 'user'. Any new users created will automatically join these groups. Note: All users need this permission to login to JIRA, even if they have other permissions.	jira-users (View Users | Delete)
Browse Users Ability to select a user or group from a popup window. Users with this permission will be able to see names of all users and groups in the system.	jira-developers (View Users | Delete)
Create Shared Filter Ability to share a filter globally or with group of users.	jira-developers (View Users | Delete)
Manage Group Filter Subscriptions Ability to manage (create and delete) group filter subscriptions.	jira-developers (View Users | Delete)
Bulk Change Ability to modify a collection of issues at once. For example, resolve multiple issues in one step.	jira-users (View Users | Delete)

Managing Project Roles

On this page:

- Using project roles
- JIRA's default project roles
- Viewing project roles
- Adding a project role
- Deleting a project role
- Editing a project role
- Assigning members to a project role
- Specifying 'default members' for a project role

Project roles are a flexible way to associate users and/or groups with particular projects. In JIRA Enterprise, project roles also allow for delegated administration:

- Global administrators define JIRA's project roles — that is, all projects have the same project roles available to them.
- Project administrators* assign members to project roles specifically for their project(s).

Project roles can be used in:

- permission schemes
- email notification schemes
- issue security levels
- comment visibility
- workflow conditions

Project roles can also be given access to:

- issue filters *(in JIRA Professional and Enterprise editions)*
- dashboards *(in JIRA Professional and Enterprise editions)*

Project roles are somewhat similar to groups, the main difference being that group membership is global whereas project role membership is project-specific. Additionally, group membership can only be altered by JIRA administrators, whereas project role membership can be altered by project administrators*.

* A project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'JIRA Administrator' permission. In JIRA Enterprise, a project administrator can manage project role membership. In JIRA Professional and Standard editions, only global administrators can manage project role membership.

Using project roles

Project roles enable you to associate users with particular functions. For example, if your organisation requires all software development issues to be tested by a Quality Assurance person before being closed, you could do the following:

1. Create a project role called **Quality Assurance**.
2. Create a permission scheme called **Software Development,** in which you assign the ' Close Issue ' permission to the **Quality Assurance** project role.
3. Associate the **Software Development** permission scheme with all software development projects.
4. For each software development project, add the appropriate Quality Assurance people to the **Quality Assurance** project role.

JIRA versions prior to 3.7 did not have project roles. If you previously used JIRA 3.6.x (or earlier), please see Using 'Scheme Tools' to migrate to Project Roles.

JIRA's default project roles

When you install JIRA, three project roles are automatically created:

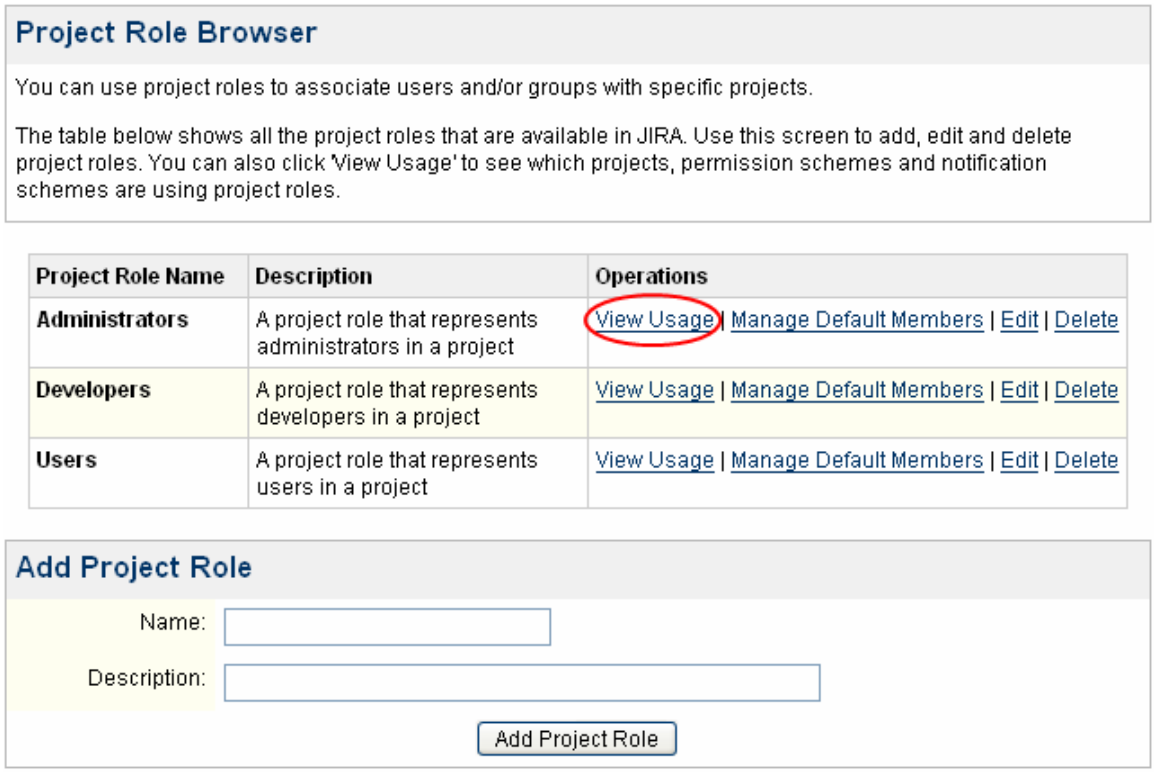
- **Administrators** — typically contains people who administer a given project.
- **Developers** — typically contains people who work on issues in a given project.
- **Users** — typically contains people who log issues in a given project.

You can create, edit and delete project roles according to your organisation's requirements.

Viewing project roles

To see what project roles exist, and where they are used:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Select **'Project Role Browser'** from the **'Users Groups & Roles'** section of the **'Administration'** menu. You will then see the Project Role Browser, which contains a list of all the project roles in your JIRA system. To see where a project role is used, click the **'View Usage'** link:



Project Role Browser

You can use project roles to associate users and/or groups with specific projects.

The table below shows all the project roles that are available in JIRA. Use this screen to add, edit and delete project roles. You can also click 'View Usage' to see which projects, permission schemes and notification schemes are using project roles.

Project Role Name	Description	Operations
Administrators	A project role that represents administrators in a project	View Usage | Manage Default Members | Edit | Delete
Developers	A project role that represents developers in a project	View Usage | Manage Default Members | Edit | Delete
Users	A project role that represents users in a project	View Usage | Manage Default Members | Edit | Delete

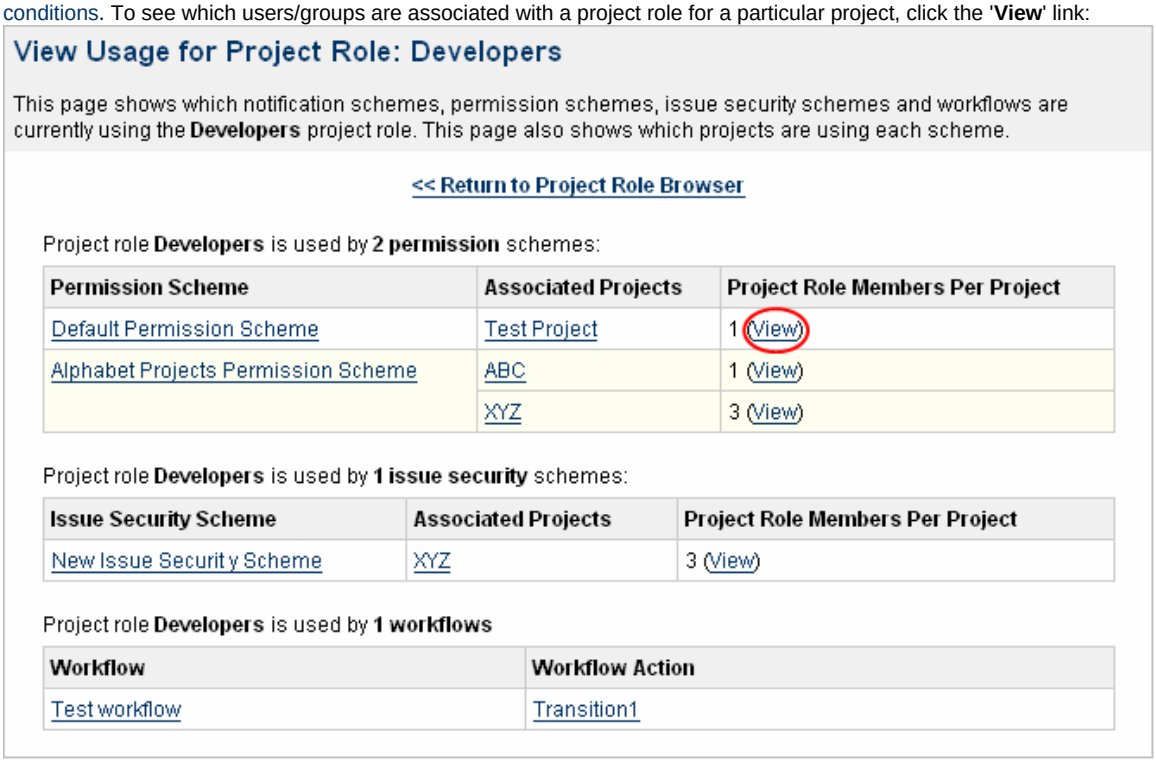
Add Project Role

Name:

Description:

Add Project Role

4. This will display a list of the project role's associated permission schemes, email notification schemes, issue security levels and workflow conditions. To see which users/groups are associated with a project role for a particular project, click the '**View**' link:



View Usage for Project Role: Developers

This page shows which notification schemes, permission schemes, issue security schemes and workflows are currently using the **Developers** project role. This page also shows which projects are using each scheme.

<< Return to Project Role Browser

Project role **Developers** is used by **2 permission** schemes:

Permission Scheme	Associated Projects	Project Role Members Per Project
Default Permission Scheme	Test Project	1 (View)
Alphabet Projects Permission Scheme	ABC	1 (View)
	XYZ	3 (View)

Project role **Developers** is used by **1 issue security** schemes:

Issue Security Scheme	Associated Projects	Project Role Members Per Project
New Issue Security Scheme	XYZ	3 (View)

Project role **Developers** is used by **1 workflows**

Workflow	Workflow Action
Test workflow	Transition1

Adding a project role

To define a new project role, enter its Name and a Description in the **'Add Project Role'** form in the Project Role Browser (see 'Viewing Project Roles' above), and click the **'Add Project Role'** button. Note that project role names must be unique.

Once a new project role is created, it is available to all projects. Project administrators can then assign members to the project role for their project (see Managing project role membership).

Deleting a project role

To delete a project role, locate the project role in the Project Role Browser (see 'Viewing Project Roles' above), and click the **'Delete'** link. The confirmation screen that follows lists any permission schemes, email notification schemes, issue security levels and workflow conditions that use the project role.

Note that deleting a project role will remove any assigned users and groups from that project role, for all projects. Be aware of the impact this may have; for example, if the project role membership was the sole conveyor of a permission for a user, then the user will no longer have that permission.

> If a project role has been used to specify who can view a comment, deleting the project role will mean that no-one can see that comment any more.

Editing a project role

To edit the **Name** and **Description** of a project role, locate the project role in the Project Role Browser (see 'Viewing Project Roles' above), and click the **'Edit'** link. This takes you to a form where you can modify the project role's **'Name'** and **'Description'**.

Assigning members to a project role

A project role's members are assigned on a project-specific basis. To assign users/groups to a project role for a particular project, please see Managing project role membership.

To see/edit *all* the project roles to which a particular user belongs, for all projects, click the '**Project Roles**' link in the User Browser.

Specifying 'default members' for a project role

The default members for a project role are users and groups that are initially assigned to the project role for all newly created projects. The actual membership for any particular project can then be modified by the project administrator.

The default members consist of the **Default Users** plus the **Default Groups** shown in the Project Role Browser (see 'Viewing Project Roles' above).

To add to the **Default Users** or the **Default Groups** for a project role, click the corresponding **'Edit'** link.

For example, if a user called Susie needs to have administration permissions for all newly created projects, you could add her to the **Default Users** for the 'Administrator' project role as follows:

1. Open the Project Role Browser.
2. Click the **'Edit'** link in the **Administrators** column (next to *'None selected')*.
3. In the 'Assign Default Users to Project Role' screen, click the '**User Picker**' icon.
4. Locate Susie in the 'User Picker' popup window, then click the '**Select**' button.
5. In the 'Assign Default Users to Project Role' screen, click the '**Add**' button.

> Changing a project role's default members does not affect the actual project role members for projects already created.

Migrating User Groups to Project Roles

Project roles are a flexible way of associating particular users and groups with a particular project.

On this page:

- Why migrate to Project Roles?
- Updating Permission Schemes and Notification Schemes to use Project Roles instead of Groups
- Minimising the number of Permission Schemes and Notification Schemes

Why migrate to Project Roles?

- **Ease of management** — JIRA versions prior to 3.7 did not have project roles. If you previously used JIRA 3.6.x (or earlier), your system may contain multiple, project-specific groups, permission schemes and notification schemes. By implementing project roles, you may be able to reduce the number of groups, permission schemes and notification schemes in your JIRA system. This can make your system easier to manage.
- **Delegated administration** — In JIRA Enterprise, a project administrator (that is, someone who has the 'Administer Project' permission, but not neceessarily the global 'JIRA Administrator' permission) can assign users and groups to project roles for their project. If their project's permission scheme and notification scheme are using project roles, the project administrator can control who may access their project and who receives email notifications.

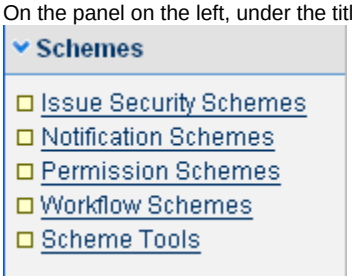
The instructions on this page will help you use Scheme Tools to:

- update your permission schemes and notification schemes so that they use project roles instead of groups; then
- minimise the number of permission schemes and notification schemes in your JIRA system.

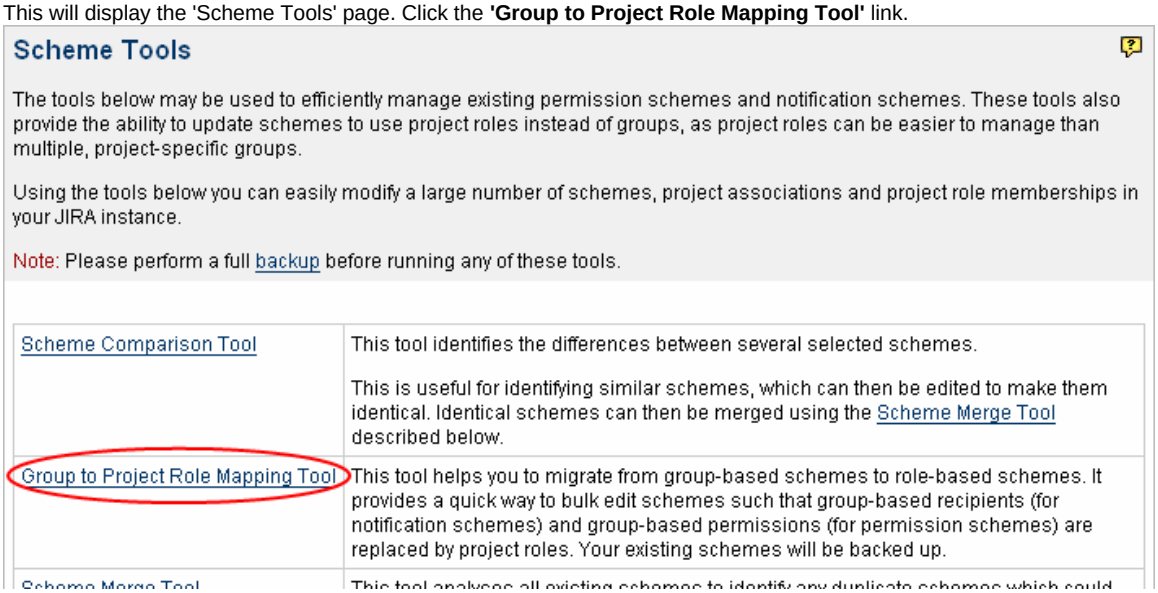
Updating Permission Schemes and Notification Schemes to use Project Roles instead of Groups

> **Backup your data**
> Before you begin, please perform a full backup.

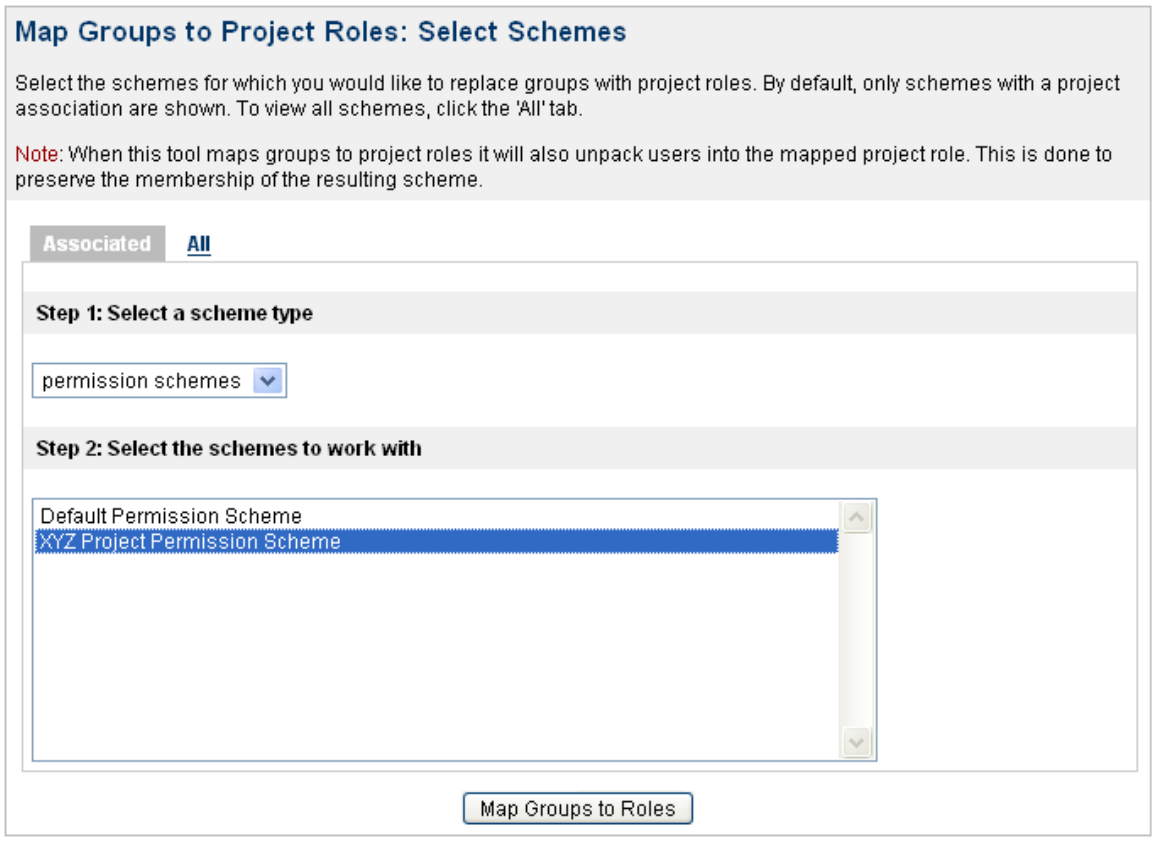
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click the link labelled '**Scheme Tools**'.



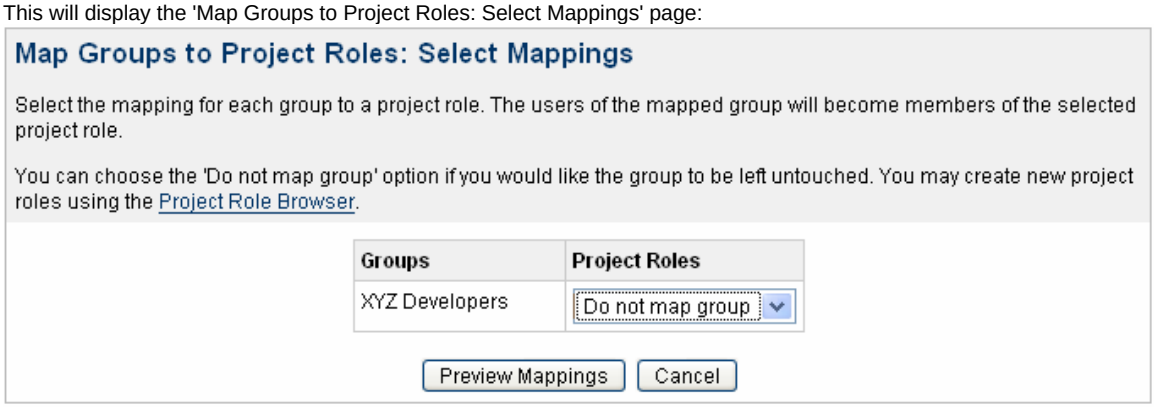
4. This will display the 'Scheme Tools' page. Click the **'Group to Project Role Mapping Tool'** link.



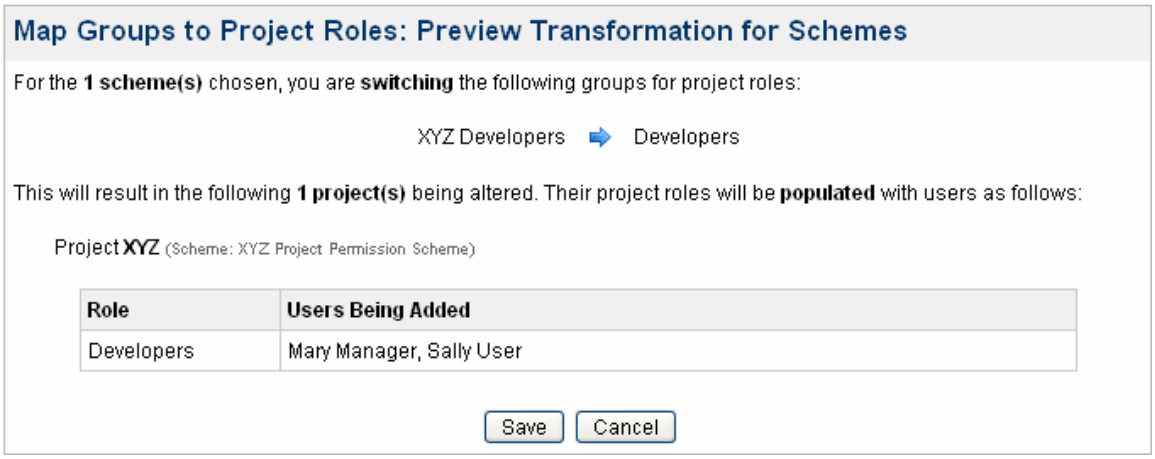
5. This will display the 'Map Groups to Project Roles: Select Schemes' page:



#* Note that schemes that are not associated with any projects need not usually be included in this process; but if you wish to select from all schemes in your system (including unused schemes), click '**All**'.
   - Under **'Step 1: Select a scheme type',** select whether you want to update permission schemes or notification schemes. (You can only do one type of scheme at a time, but you can easily come back and do the other type later).
   - Under **'Step 2: Select the schemes to work with',** select the schemes you want to update to use project roles instead of groups. You can use the **'Ctrl'** key to select multiple schemes.
   - Click the '**Map Groups to Roles**' button.
6. This will display the 'Map Groups to Project Roles: Select Mappings' page:



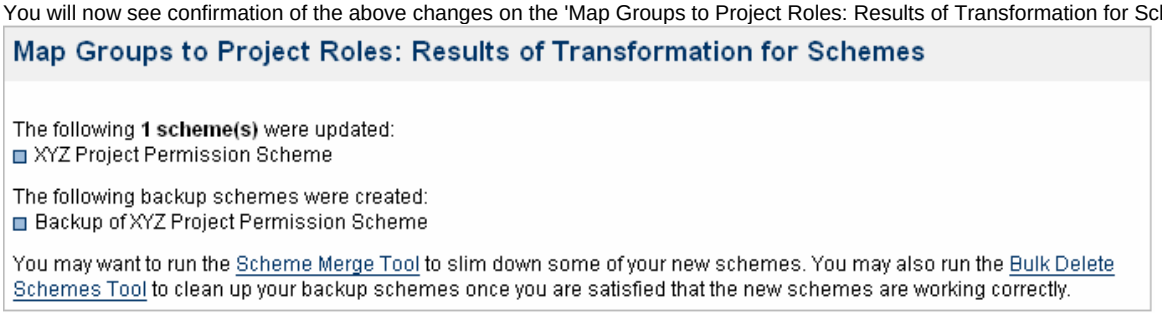
   For each group, select the project role that will replace it; or, for any groups that you do not want to migrate, choose the '**Do not map group**' option. Then click the '**"Preview Mappings**' button.
   - For ease of maintenance, it is recommended that you do not migrate any groups to which JIRA users are automatically added (that is, groups which have the 'JIRA Users' global permission). If you migrate these groups to project roles, and you still want all new users to have access to particular projects, you will need to manually add new users to the relevant project role for each project.
7. You will now see the 'Map Groups to Project Roles: Preview Transformation for Schemes' page:



If you are satisfied that the information shown on this page is correct, click the '**Save**' button to:

- create a backup of the scheme(s) that you selected in step 5 (you can later delete this backup scheme by using the 'Bulk Delete Schemes Tool', available from the 'Scheme Tools' page shown in step 4). This backup scheme will not be associated with any projects.
- update the scheme(s) that you selected in step 5 to use the role (left of the blue arrow) instead of the group (right of the blue arrow)
- add the users (in the right column of the table) to the project role (in the left column of the table) for each project that uses the scheme. This ensures that all users will continue to have the same permissions and notifications.

8. You will now see confirmation of the above changes on the 'Map Groups to Project Roles: Results of Transformation for Schemes' page:



After updating your permission schemes and notification schemes to use project roles instead of groups, you may find that many of your schemes are now very similar. To identify such schemes, merge them, and delete any redundant ones, please see Minimising the number of Permission Schemes and Notification Schemes (below).

You may also find that some groups are no longer required. You can use the Group Browser to identify and delete groups that are not used by any permission schemes or notification schemes.

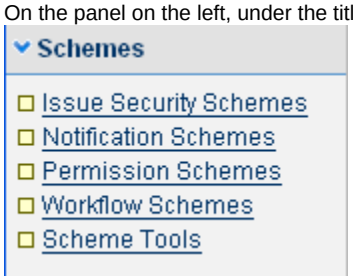
Minimising the number of Permission Schemes and Notification Schemes

Minimising the number of permissions schemes and notification schemes can make your JIRA system easier to manage. To identify and remove unnecessary schemes, follow the steps below:

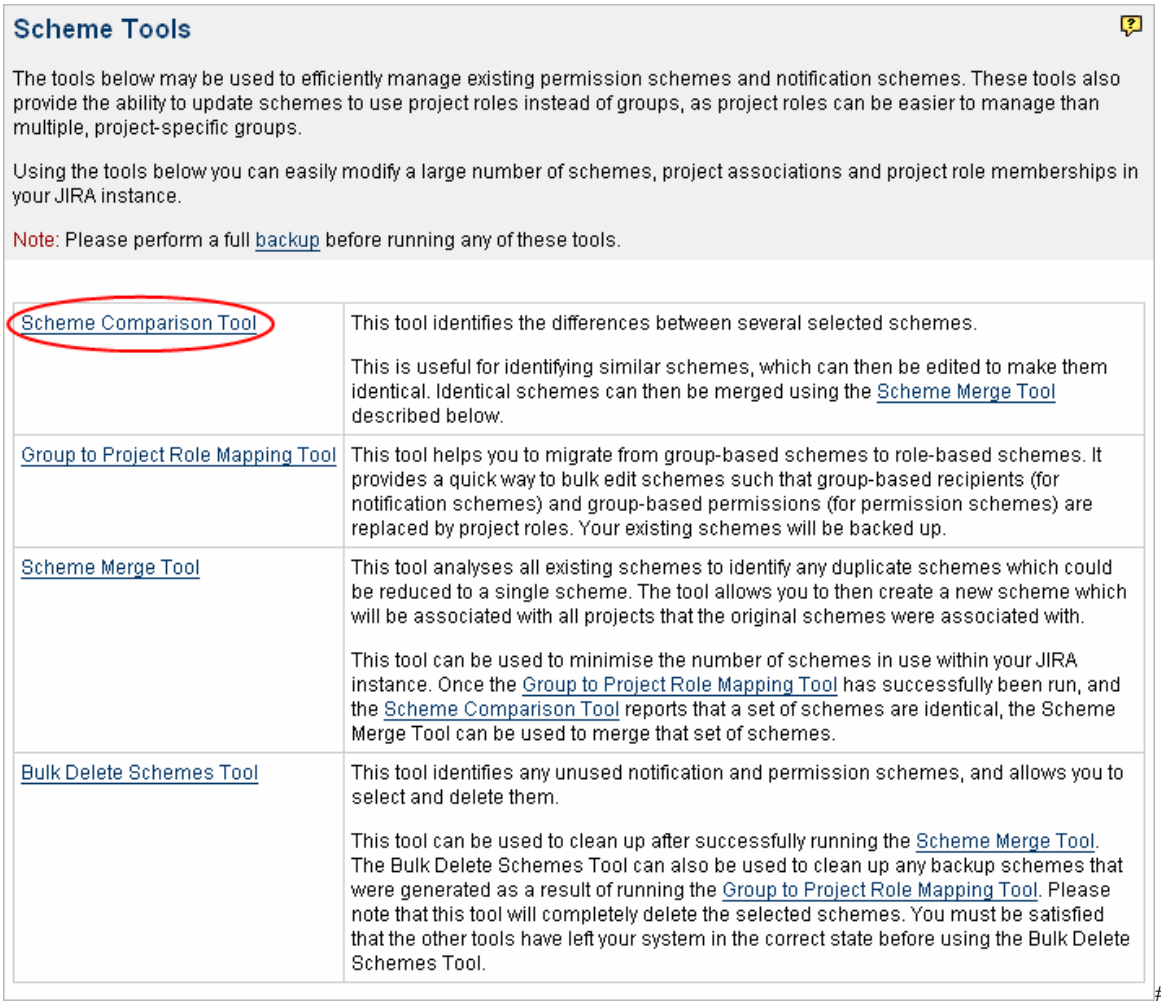
> Before you begin, please perform a full backup.

Log in as a user with the **'JIRA Administrators'** global permission.

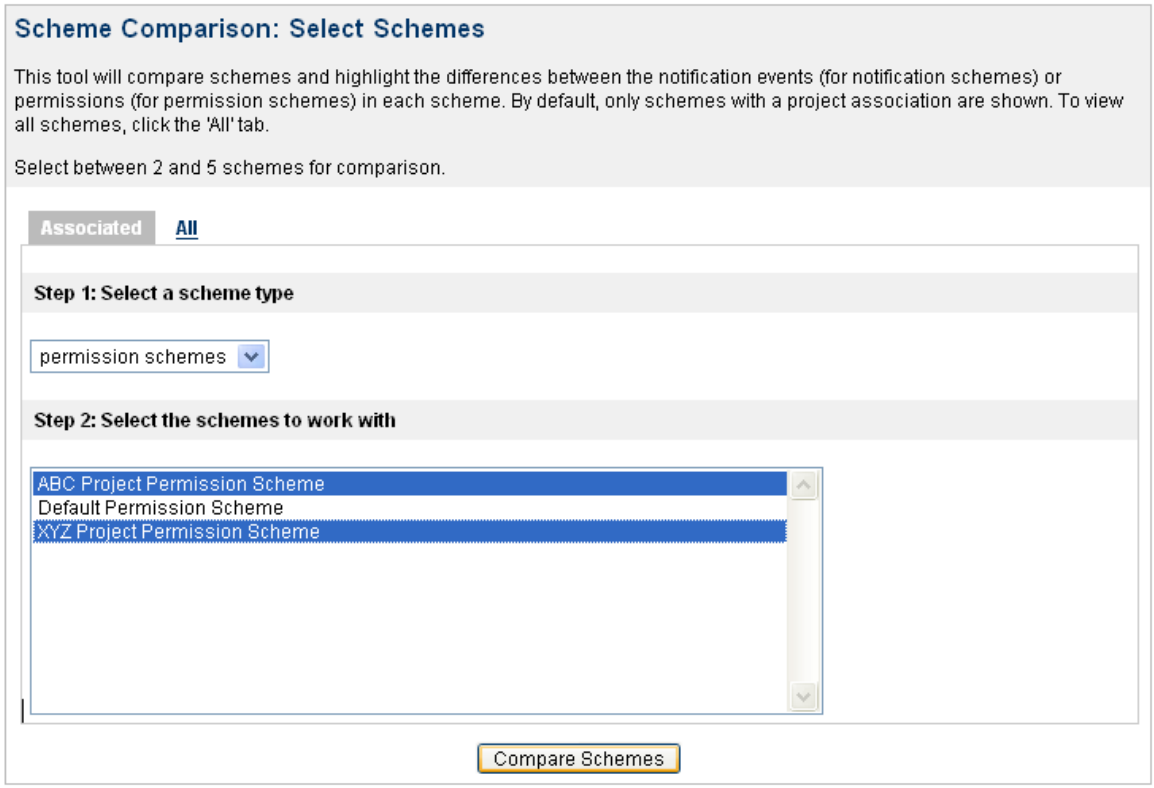
1. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
2. On the panel on the left, under the title '**Schemes**', click the link labelled '**Scheme Tools**'.



3. This will display the '**Scheme Tools**' page. Click the '**Scheme Comparison Tool**' link.



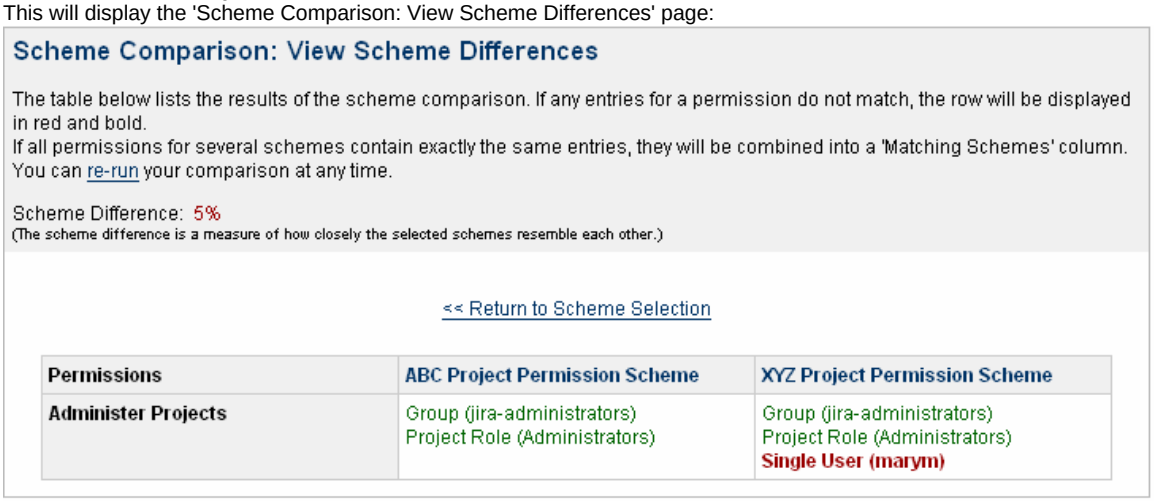
#* The Scheme Comparison Tool assists you in identifying similar schemes, and if appropriate, making them identical.
    - Identical schemes can later be merged using the Merge Duplicate Schemes Tool (see step 9 below).
4. This will display the '**"Scheme Comparison: Select Schemes**' page:



#* Note that schemes which are not associated with any projects need not usually be included in this process; but if you wish to select from all schemes in your system (including unused schemes), click '**All**'.

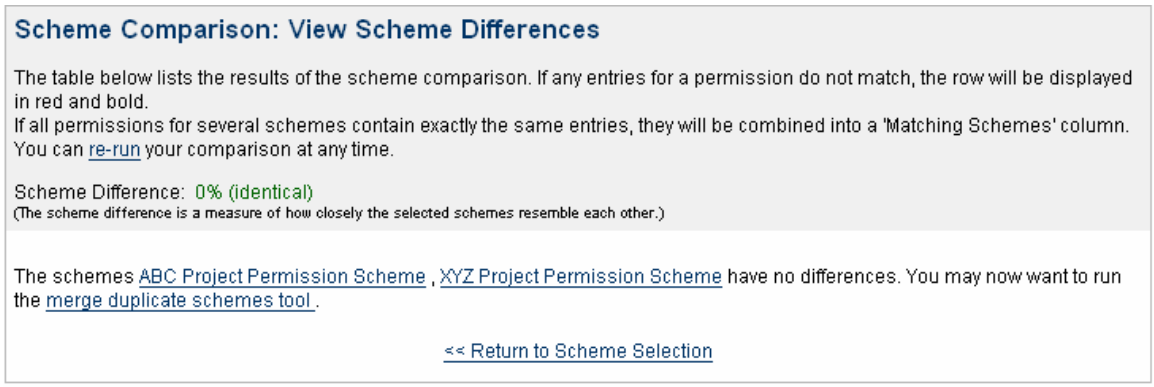
- Under '**Step 1: Select a scheme type',** select whether you want to compare permission schemes or notification schemes. (You can only do one type of scheme at a time, but you can easily come back and do the other type later.)
- Under **Step 2: Select the schemes to work with,** select the schemes you want to compare. Select at least 2 (and no more than 5) schemes, using the **'Ctrl'** key to select multiple schemes.
- Click the '**Compare Schemes**' button.

5. This will display the 'Scheme Comparison: View Scheme Differences' page:

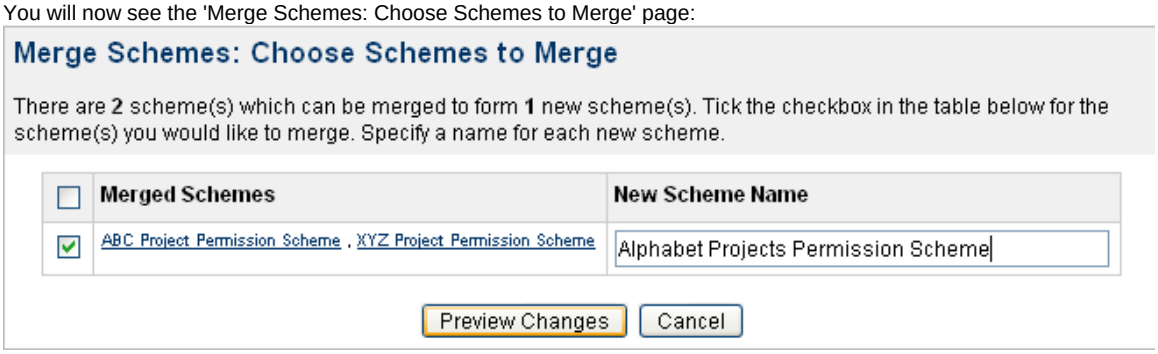


#* Only the **differences** between the selected schemes are displayed. For example, in the screenshot above, only the "Administer Projects" permission is displayed; this means that all the other permissions in these two permission schemes ("ABC Project Permission Scheme" and "XYZ Project Permission Scheme") are identical.

6. If you decide it is appropriate to edit a scheme to make it the same as another one, you can edit the scheme by clicking the scheme name. For example, it may be appropriate to delete **Single User (marym)** from the "XYZ Project Permission Scheme" if she is a member of the "Administrators" project role for the XYZ project.
7. Then repeat the steps above, and verify that you have achieved a batch of 2 or more identical permission schemes, e.g.

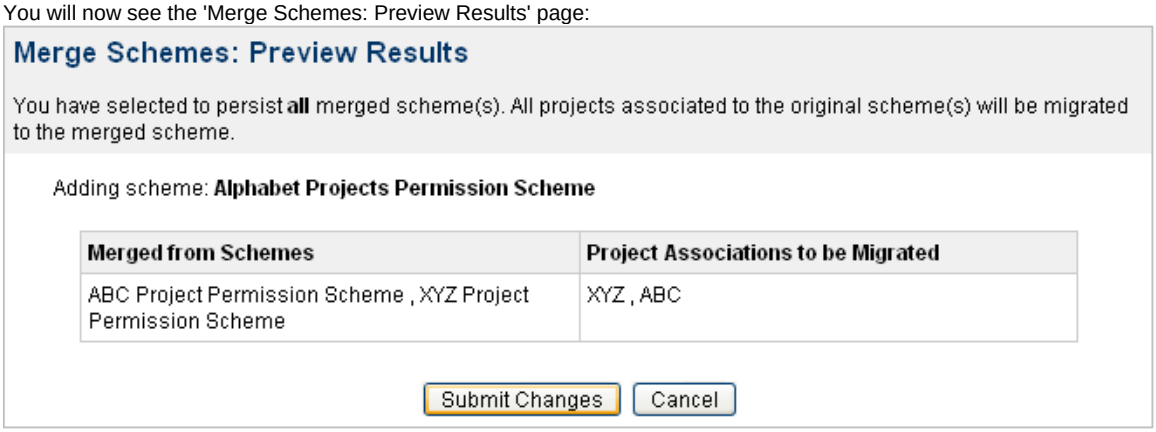


8. Click the '**Merge Duplicate Schemes Tool**' link. (Note: this link is also available from the 'Scheme Tools' page shown in step 4).
9. You will now see the 'Merge Schemes: Choose Schemes to Merge' page:



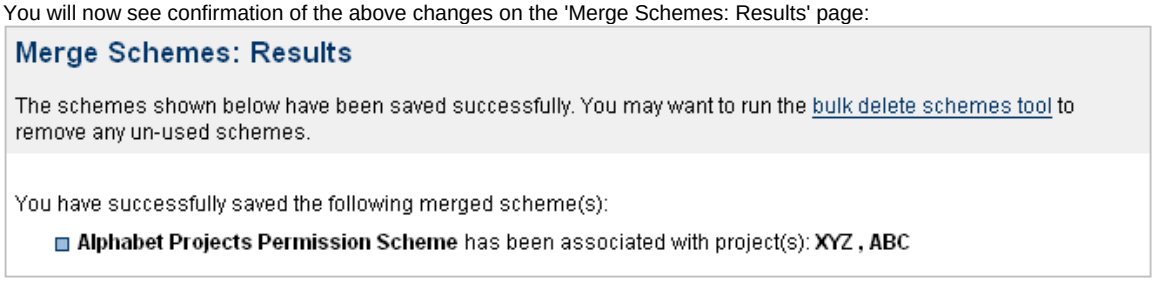
If you decide it is appropriate to merge the displayed schemes:
- Check the box next to the scheme names.
- Type a name for the new scheme in the '**New Scheme Name**' box.
- Click the '**Preview Changes**' button.

10. You will now see the 'Merge Schemes: Preview Results' page:

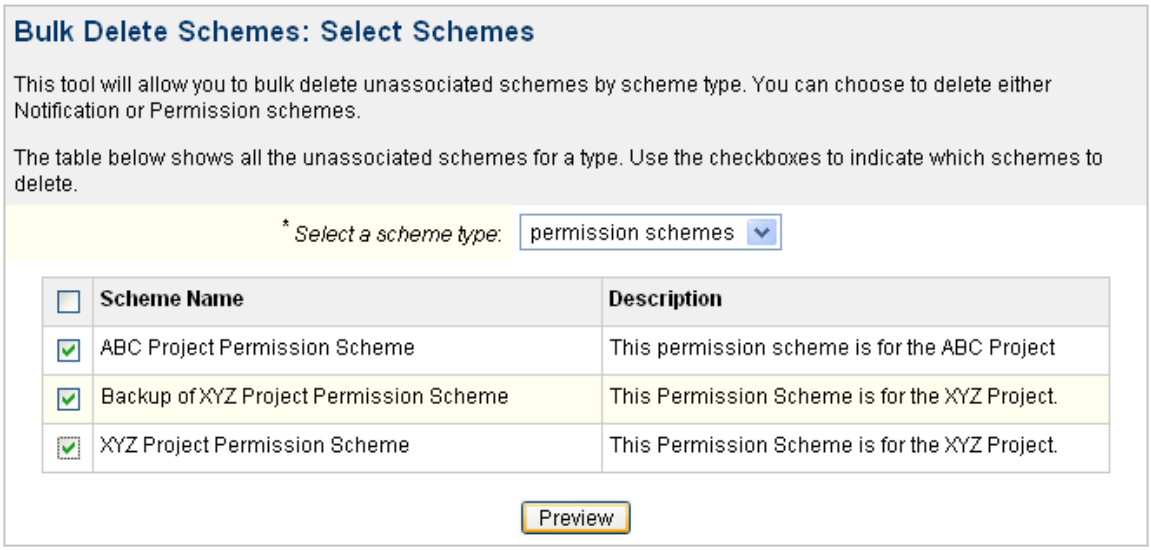


If you are satisfied that the information shown on this page is correct, click the '**Submit Changes**' button to:
- create the new scheme whose name is shown in bold.
- associate the projects (in the right column of the table) with the new scheme.
- disassociate the existing schemes (in the left column of the table) from the projects. These schemes can then be deleted using the 'Bulk Delete Schemes Tool' (see step 15).

11. You will now see confirmation of the above changes on the 'Merge Schemes: Results' page:



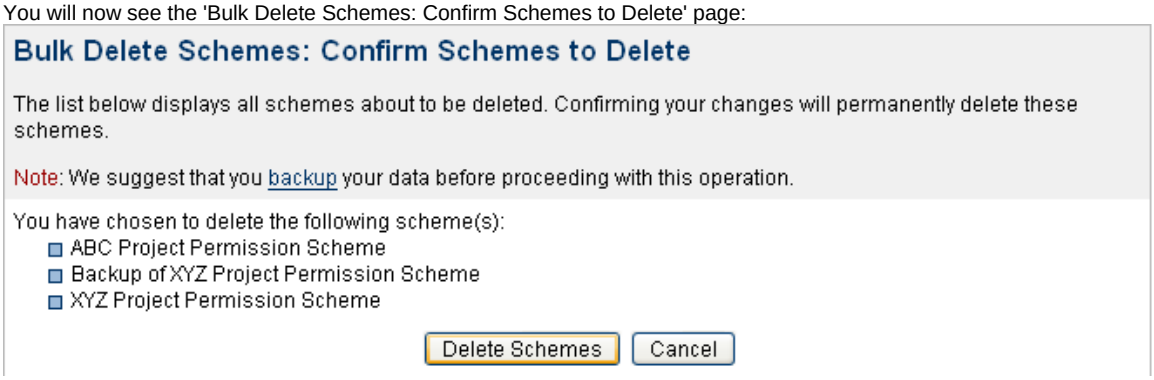
12. Click the '**Bulk Delete Schemes Tool**' link. (Note: this link is also available from the 'Scheme Tools' page shown in step 4).
13. You will now see the '**Bulk Delete Schemes: Select Schemes**' page:



If you decide it is appropriate to delete the displayed schemes:

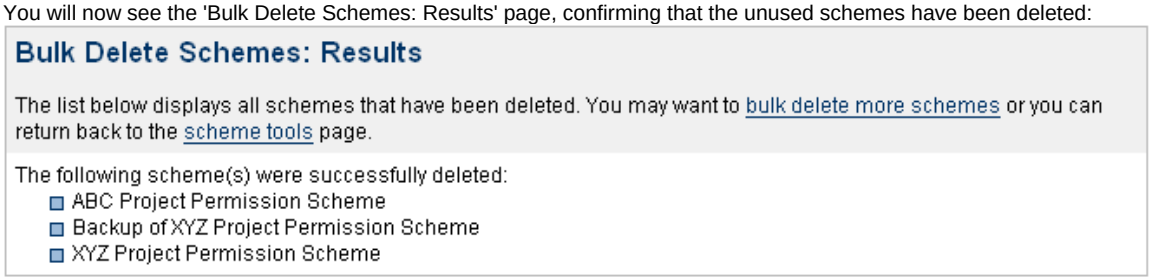
- Check the box next to the scheme names.
- Type a name for the new scheme in the "New Scheme Name" box.
- Click the '**"Preview**' button. Note that deleting these schemes will not affect any projects, as this page only displays schemes that are not associated with projects.

14. You will now see the 'Bulk Delete Schemes: Confirm Schemes to Delete' page:



If you are satisfied that the information shown on this page is correct, click the '**Delete Schemes**' button.

15. You will now see the 'Bulk Delete Schemes: Results' page, confirming that the unused schemes have been deleted:



Enabling Public Signup and CAPTCHA

For some organisations it is appropriate to enable *signup,* which allows visitors to immediately create their own JIRA user accounts. If signup is not enabled, then only a JIRA administrator can create new user accounts.

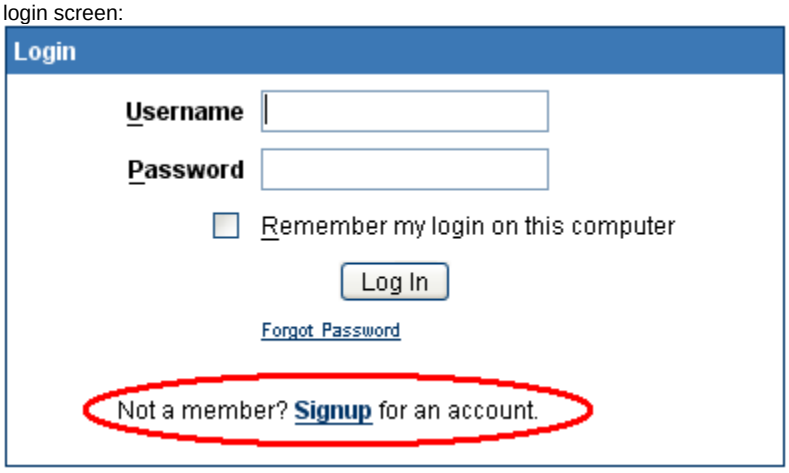
For example, enabling signup can be useful if you are using JIRA as a support system and have a very large number of potential users, of which only some will need to log support tickets.

For security reasons, even if you enable signup, it is still necessary for users to have the appropriate project permissions before they can see or create issues. Note that you can use automatic group membership to add all new users to appropriate groups.

If your JIRA server is accessible from outside your organisation's firewall, and you have enabled signup, then you may want to also enable *CAPTCHA.* CAPTCHA helps ensure that only real humans (and not automated spam systems) can sign themselves up to JIRA. When CAPTCHA is enabled, visitors will need to recognise a distorted picture of a word (see example below), and must type the word into a text field. This is easy for humans to do, but very difficult for computers.

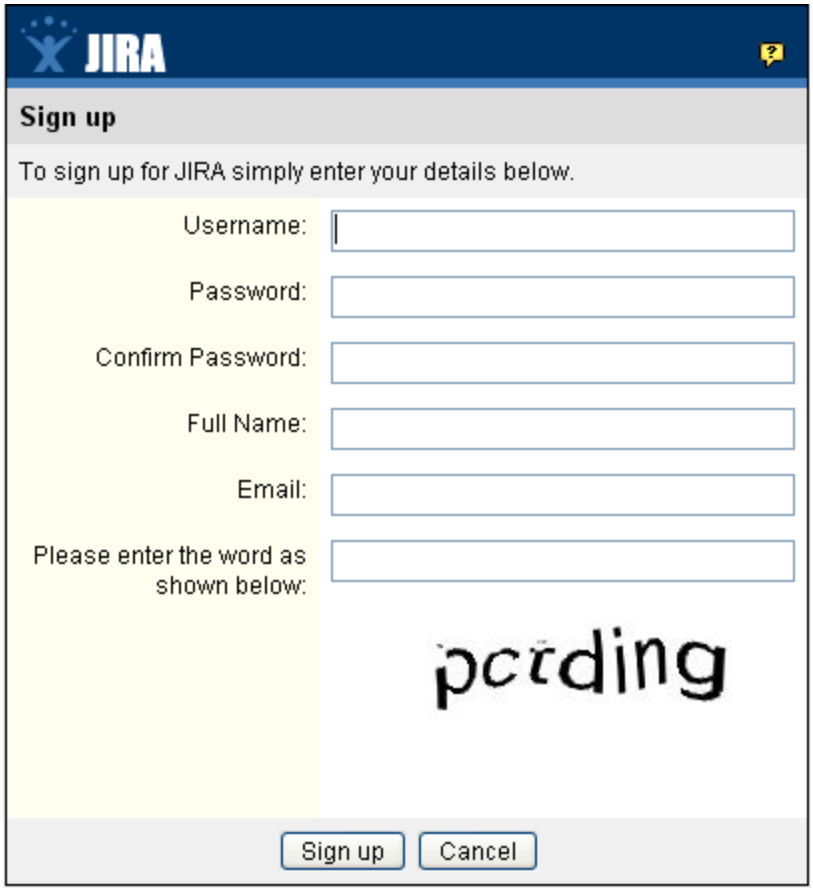
Enabling Public Signup

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left navigation column, click '**General Configuration**'.
4. This will display the 'General Configuration' screen. Click the **'Edit Configuration'** link at the bottom of the screen.
5. In the **'Mode'** drop-down, select **'Public'**.
6. Click the **'Update'** button at the bottom of the screen.
7. Log out of JIRA, then click the '**Log In**' link at the top right of the screen and verify that the **'Signup'** link is displayed at the bottom of the login screen:



Enabling CAPTCHA

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left navigation column, click '**General Configuration**'.
4. This will display the 'General Configuration' screen. Click the **'Edit Configuration'** link at the bottom of the screen.
5. Locate **'CAPTCHA on signup'** and select **'On'**.
6. Click the **'Update'** button at the bottom of the screen.
7. Log out of JIRA, click the '**Log In**' link at the top right of the screen, then click the '**Signup**' link and verify that a random sequence of letters is displayed at the bottom of the **'Signup'** screen - e.g. "pctding" in the following screenshot:



Integrating JIRA with LDAP

Many organisations have an LDAP directory acting as a centralised database of system users. JIRA is able to authenticate users against their LDAP password.

On this page:

- About JIRA's LDAP Integration
  - Caveats
    - Only password-checking for LDAP users is done in JIRA
    - Not all LDAP users have JIRA access
  - Planned Improvements
  - Atlassian Crowd
- Step 1. Configuring LDAP Integration
  - JIRA Enterprise Edition
  - JIRA Professional & Standard Editions
- Step 2. Disabling JIRA's Password Management
- Configuration Notes
  - LDAP over SSL
  - Multiple LDAP trees (eg. ActiveDirectory domains)
  - LDAP on Linux
- Debugging
  - Cannot create osuser.xml file via JIRA LDAP Configurer
  - Cannot authenticate against LDAP password

About JIRA's LDAP Integration

In JIRA, user management is handled by OSUser, a pluggable user management framework. OSUser is configured through the **WEB-INF/classes/osuser.xml** file.

Caveats

Only password-checking for LDAP users is done in JIRA

The main point to realise is that user profiles are *still managed in JIRA* (the OFBizProfileProvider in osuser.xml). Only the password lookup is done against LDAP, and only if the JIRA username coincides with a LDAP username.

If the username is not found in LDAP, then the local JIRA credentials will be used.

(Technically, this behaviour is due to credentials (password) checking being a separate operation to user-profile lookups. The profile can be loaded from the JIRA database, but the password looked up from LDAP. Furthermore, multiple credentials providers can be specified (here, LDAP and OSUser), and if one fails, the other will be used. This allows non-LDAP users to log in with their JIRA password.)

Not all LDAP users have JIRA access

Another effect of this implementation is that LDAP users *do not automatically have access to JIRA.* A JIRA account must be created for each user wishing to use JIRA. You can bulk-create users from LDAP with this LDAP user importer.

This is because each JIRA user has a set of **groups** (for example, 'jira-users') stored in their profile. Without an associated group, that user can do nothing; not even browse JIRA (they lack the 'use' permission).

Thus, for an LDAP user to be able to use JIRA, a JIRA administrator must create an account for them, and assign them to a group (typically 'jira-user'). The password in this JIRA account will be ignored, as the LDAP password will override it.

Planned Improvements

In future, we plan to more tightly integrate LDAP into JIRA, so that LDAP groups can be mapped to JIRA groups, and user management can be fully externalised (see Plans for JIRA's LDAP integration). We'd like feedback as we go further with this; please see this issue report for current status, and add your use-case as a comment. Thanks!

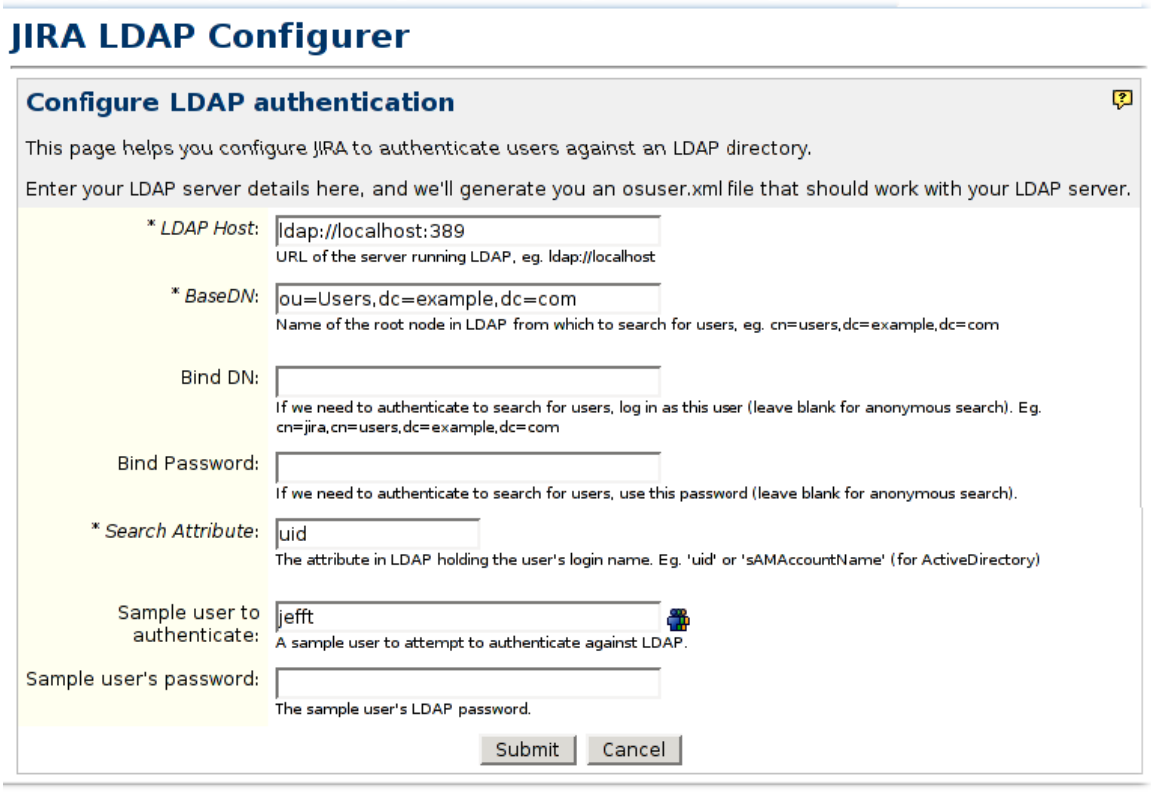
Atlassian Crowd

Atlassian Crowd is Atlassian's single sign-on product. Crowd provides an interface for managing users and groups, and provides an OSUser implementation that allows JIRA to fully delegate users and groups to Crowd, and hence LDAP. Thus while we still plan to develop tighter JIRA-LDAP integration, purchasers of Crowd can achieve this right now by using Crowd as an intermediary. See Crowd's JIRA integration documentation for details.

Step 1. Configuring LDAP Integration

JIRA Enterprise Edition

JIRA Enterprise contains a configuration utility which lets you auto-generate a valid **osuser.xml** file. This can be accessed from **Admin -> System -> LDAP:**



JIRA Professional & Standard Editions

In **JIRA Professional** and **JIRA Standard,** LDAP integration is configured by manually editing the **WEB-INF/classes/osuser.xml** file.

In the JIRA Standalone distribution, this file can be edited directly in **atlassian-jira/WEB-INF/classes/osuser.xml.** In the WAR/webapp distribution, it should be copied from **webapp/WEB-INF/classes/osuser.xml** to **edit-webapp/WEB-INF/classes/,** edited there, and then rebuilt into an updated .war file with the '`ant war`' command.

The default osuser.xml contains:

```
<opensymphony-user>
  <authenticator
      class="com.opensymphony.user.authenticator.SmartAuthenticator" />

  <provider
      class="com.atlassian.core.ofbiz.osuser.CoreOFBizCredentialsProvider">
      <property name="exclusive-access">true</property>
  </provider>

  <provider
      class="com.opensymphony.user.provider.ofbiz.OFBizProfileProvider">
    <property name="exclusive-access">true</property>
  </provider>

  <provider
      class="com.opensymphony.user.provider.ofbiz.OFBizAccessProvider">
    <property name="exclusive-access">true</property>
  </provider>

</opensymphony-user>
```

CredentialsProviders are responsible for checking usernames and passwords, which is what we are interested in here. The default **CoreOFBizCredentialsProvider** looks in the JIRA database. We are going to add a **LDAPCredentialsProvider,** so that LDAP users can also be authenticated:

```
<opensymphony-user>
  <authenticator class="com.opensymphony.user.authenticator.SmartAuthenticator" />

<provider
      class="com.opensymphony.user.provider.ldap.LDAPCredentialsProvider">
 <property name="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</property>
 <property name="java.naming.provider.url">ldap://localhost:389</property>
<property name="searchBase">dc=atlassian,dc=com</property>
 <property name="uidSearchName">uid</property>
  _<!--
 <property name="java.naming.security.principal">cn=Manager,dc=atlassian,dc=com</property>
 <property name="java.naming.security.credentials">secret</property>
 <property name="exclusive-access">true</property>
-->_
</provider>

<provider class="com.atlassian.core.ofbiz.osuser.CoreOFBizCredentialsProvider">
  <property name="exclusive-access">true</property>
</provider>

<provider class="com.opensymphony.user.provider.ofbiz.OFBizProfileProvider">
  <property name="exclusive-access">true</property>
</provider>

<provider class="com.opensymphony.user.provider.ofbiz.OFBizAccessProvider">
  <property name="exclusive-access">true</property>
</provider>

</opensymphony-user>
```

It is necessary to use *both* the LDAP and OFBiz providers, and the order must be as shown (LDAP first).

Some LDAP properties that are commonly set here are:

Property	Required	Description	Example
java.naming.factory.initial	Yes	Specifies that JNDI (the directory API) should use the LDAP implementation	Should always be **com.sun.jndi.ldap.LdapCtxFactory**
java.naming.provider.url	Yes	LDAP URL of your server	ldap://localhost:389
searchBase	Yes	The node in the LDAP tree to search below for usernames.	The root LDAP node is typically called 'dc= *companyname,* dc=com', and user account nodes are usually stored in a subtree, like 'cn=Users,dc= *companyname,* dc=com'.
uidSearchName	Yes	Attribute expected to contain username	Typically **uid,** or **sAMAccountName** for MS ActiveDirectory.
java.naming.security.principal	No	Username to initially log in to LDAP as. Not required if anonymous user lookups are allowed.	Eg. 'cn=Administrator,cn=Users,dc= *companyname,* dc=com'
java.naming.security.credentials	No	Password for initial login. Not required if anonymous lookups are allowed.	
providerName	No	unique name for this LDAP provider. Useful when you specify multiple LDAP providers (allowing fallback), and need to distinguish them in the debug logs.	
cacheTimeout	No	The value in milliseconds for duration of password caching. Password caching reduces the load JIRA will put on the LDAP server. Only successful authentication attempts are cached. The default value is 30 minutes (1,800,000 ms).	0 (to disable caching)

The full list of properties is specified in the JNDI documentation.

Once you have made this modification and restarted JIRA, JIRA users **whose username also exists in LDAP** will be authenticated against their LDAP password.

Step 2. Disabling JIRA's Password Management

Once you have LDAP-based password checking working, you should go to **Admin -> General Configuration,** and turn on **External password management** (see Configuring JIRA documentation). This will disable the "change my password" links in the JIRA interface, ensuring that passwords are now only managed via LDAP.

Configuration Notes

LDAP over SSL

With plain LDAP, passwords may be passing over the network unencrypted, which (depending on your network security) may be a security problem. If you wish to connect to LDAP over SSL, see the Connecting to SSL services guide for details on how to import the SSL server's public key. In **osuser.xml,** you would need to use **ldaps://** in the URL if you have port 636 dedicated to LDAP over SSL.

Multiple LDAP trees (eg. ActiveDirectory domains)

If you wish to authenticate users from multiple LDAP directories or different trees in the same directory, simply edit the **OSUser** file and add a **LDAPCredentialsProvider** section for each (see 'Configuring LDAP Integration' above). JIRA will query them in order, and the first one containing the requested user will be used for password checking. As soon as a user is found, the password is checked and no further processing is done (i.e. only one password will work).

If you have more than one **LDAPCredentialsProvider** it is a good idea to give each a unique **providerName** attribute for debugging purposes.

ActiveDirectory users note: a better approach to searching multiple trees is to set up an Active Directory **Global Catalog.** This is an AD instance which mirrors records in other instances. Searching the Global Catalog is thus equivalent to searching all mirrored LDAP directories. This is faster and more reliable than JIRA's LDAP fallback.

If you have a Global Catalog set up, it can be searched via LDAP on port 3268 (eg. ldap://adserver:3268) or 3269 for SSL (eg. ldaps://adserver:3269). See this guide for more information.

LDAP on Linux

See these notes on how to set up a LDAP directory on a Linux server for use with JIRA.

Debugging

Cannot create osuser.xml file via JIRA LDAP Configurer

To see exactly why the JIRA LDAP Configurer is failing, follow these steps:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu in the left-hand navigation column, click the **'Logging & Profiling'** link.
4. The **'Logging & Profiling'** page will display. Click the **'Edit'** link next to **'com.atlassian.jira.web.action.util.LDAPConfigurer'**
5. Change the logging level to 'DEBUG', then click **'Update'.** This temporarily turns up logging for the JIRA LDAP Configurer.
6. When you next try to submit your LDAP server details via the JIRA LDAP Configurer, you should see extra logs on stdout.

Cannot authenticate against LDAP password

If JIRA does not authenticate against the LDAP password, then something is probably wrong with your setup. First, ensure that the user you are trying to connect as has a JIRA account (see above). Make sure you have the LDAP connection details correct (basename, uid, username/password). These details are best discovered with the help of an LDAP browser such as Apache Directory Studio or JXplorer.

To see exactly why LDAP authentication is failing, follow these steps:

1. Edit `log4j.properties` (instructions), locate the lines:

   ```
   log4j.category.com.opensymphony = WARN, console
   log4j.additivity.com.opensymphony = false
   ```

   Duplicate these, and change the first copy to:

   ```
   log4j.category.com.opensymphony.user.provider.ldap = DEBUG, console
   log4j.additivity.com.opensymphony.user.provider.ldap = false
   ```

   Restart JIRA to apply your change. This change will turn up logging for the LDAP authentication module.

2. When next trying to log in, you should see extra logs on stdout. A successful authentication looks like this:

   ```
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] LDAPCredentialsProvider $Revision: 1.2 $
   initializing
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] Doing initial search:
    username='cn=admin,dc=atlassian,dc=com', password='secret',
   base='ou=People,dc=atlassian,dc=com', filter='uid=jturner'
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] Found users
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] Searching below
   'uid=jturner,ou=People,dc=atlassian,dc=com' for 'uid=jturner'
   DEBUG [user.provider.ldap.LDAPCredentialsProvider] User 'jturner' successfully authenticated;
   caching for 1800000 ms
   ```

   This log was generated with the following to osuser.xml:

   ```
   <provider class="com.opensymphony.user.provider.ldap.LDAPCredentialsProvider">
   <property name="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</property>
   <property name="java.naming.provider.url">ldap://localhost:389</property>
   <property name="searchBase">ou=People,dc=atlassian,dc=com</property>
   <property name="uidSearchName">uid</property>
   <property name="java.naming.security.principal">cn=admin,dc=atlassian,dc=com</property>
   <property name="java.naming.security.credentials">secret</property>
   <property name="exclusive-access">true</property>
   </provider>
   ```

3. If you have problems, try emulating the operations performed by the LDAP authentication provider. The LDAP authentication provider works by first doing an search for the specified username, searching from base **searchBase,** using query **uidSearchName** = *username,* authenticating using the principal and credentials properties if present, or doing an anonymous search otherwise. If an entry is found, it then tries to log in to LDAP using the specified matching username and specified password.

If you get an error message **javax.naming.PartialResultException: Unprocessed Continuation Reference(s),** try adding `<property name="java.naming.referral">follow</property>` to the LDAPCredentialsProvider section.

Integrating JIRA with Crowd

Atlassian's Crowd identity management system can be integrated with JIRA. For more information please see the chapter titled 'Integrating JIRA with Crowd', in the Crowd documentation.

Configuring Trusted Applications

A 'trusted application' is an application that JIRA will allow to access specified functions on behalf of any user — without the user logging in to JIRA.

For example, when Confluence is configured as a trusted application, every Confluence user will see *exactly the same list of issues* when they view the Confluence 'JIRA Issues' macro as they see when they use the JIRA Issue Navigator as a logged-in JIRA user. Likewise, the Confluence 'JIRA Portlet' macro will appear exactly the same as it does on the user's JIRA Dashboard.

At this time, Confluence (version 2.7 or later) is the only application that can be configured as a trusted application.

Trusted applications are a potential security risk. When you configure a trusted application, you are allowing the application to access JIRA as *any user.* By doing this, you are bypassing all the built-in JIRA security measures. Do not configure a trusted application unless you trust all code in this application to behave itself at all times, and are sure that the application will maintain the security of its private key.

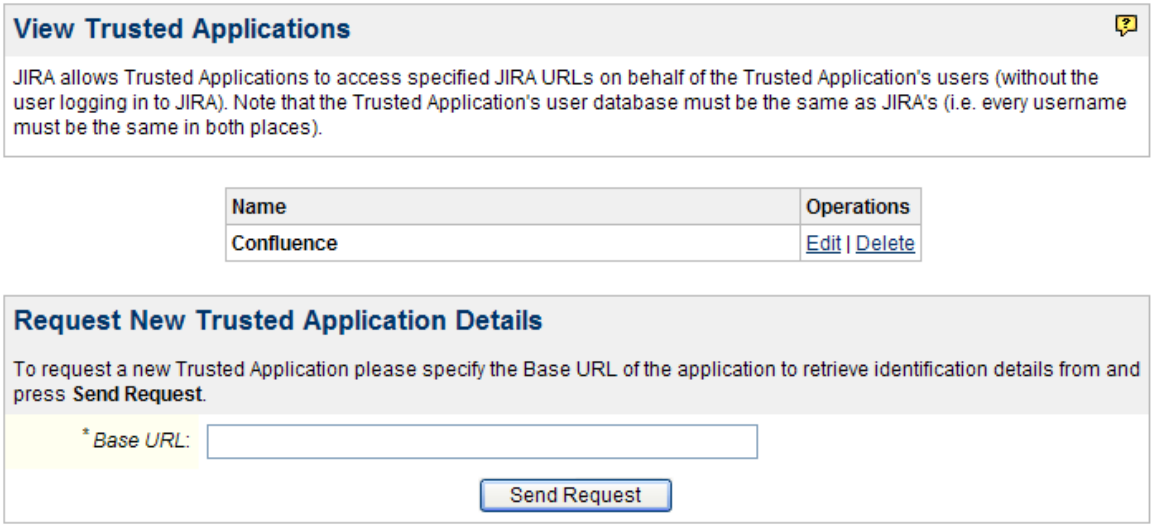
Note
All of your trusted application's users must also be JIRA users, and the usernames in both systems must be identical.

Adding a trusted application

Before you begin: Note that configuring a trusted application requires the transmission of sensitive data. To prevent 'man-in-the-middle' attacks, it is recommended that you use SSL while configuring a trusted application.

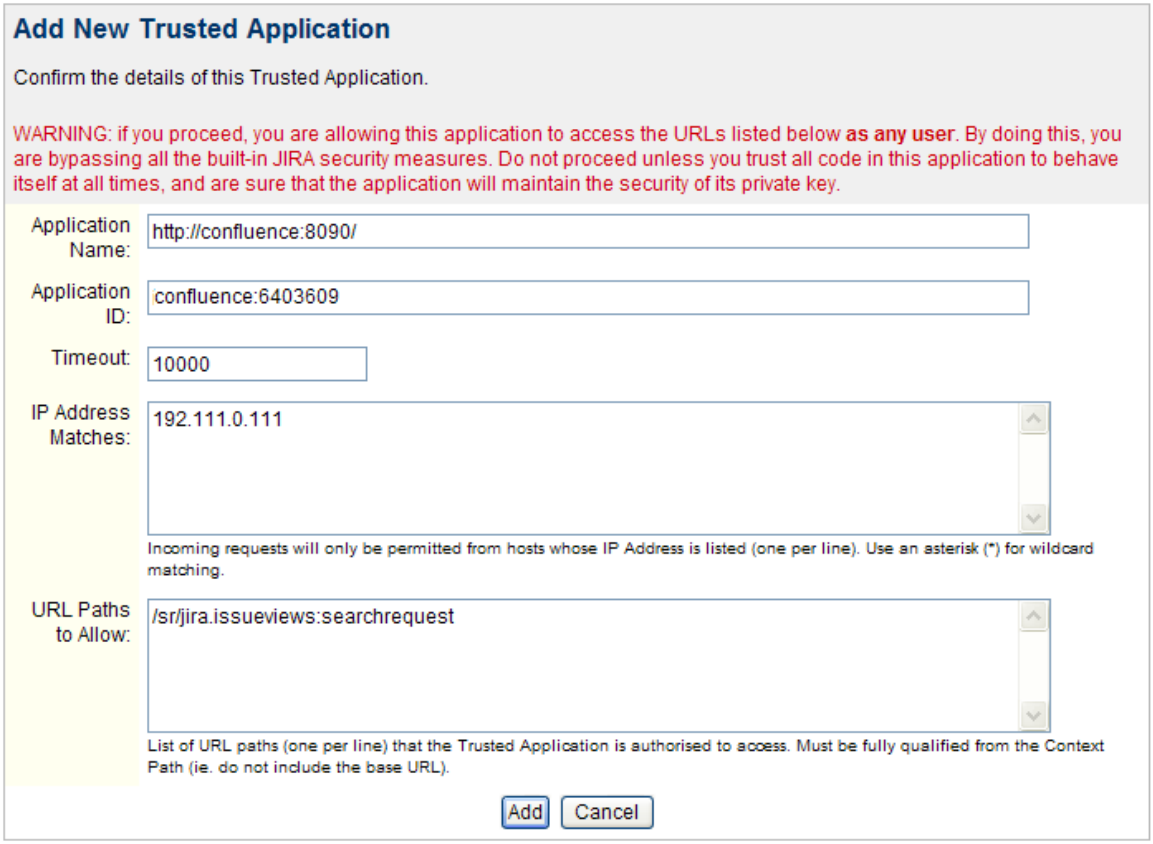
To add a trusted application,

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand panel, under the title **'System',** click the **'Trusted Applications'** link. The **'Trusted Applications'** page will be displayed, showing a list of configured trusted applications (if any). The **'Request New Trusted Application Details'** box is shown below the list.



4. In the **'Base URL'** field, type the URL that you use to access the application you wish to add (e.g. 'http://confluence.mycompany.com:8080' or 'http://www.mycompany.com/confluence').

5. Click the **'Send Request'** button to retrieve the application's ID and public key. The **'Add New Trusted Application'** screen will be displayed:



6. In the **'Application Name'** field, the URL which you typed on the previous page will be displayed. You can optionally change this if you wish (e.g. if you typed 'http://confluence.mycompany.com:8090' on the previous page, you might want to change it to just 'Confluence').
    - Note: the **'Application ID'** is generated automatically by JIRA and cannnot be edited.
7. In the **'Timeout'** field, type the number of milliseconds that JIRA will wait for a response when communicating with the trusted application.
8. In the **'IP Address Matches'** field, type the IP address (or multiple addresses, one per line) from which JIRA will accept requests on behalf of the trusted application. You can specify wildcard matches by using an asterisk , e.g. '`192.111..`'.
    - If you are using a proxy server that makes an HTTP request on the client's behalf (e.g. Squid, mod_proxy), you need to add the proxy server's IP address to this field *as well as* all the clients' IP addresses.
    - If you are using a proxy server that passes the client's IP address directly via an application server's API (e.g. AJP for Tomcat, such as mod_jk or IIS's Tomcat Connector, or mod_caucho for Resin) — or if you are not using a proxy server — then you only need to enter the clients' IP addresses.
    - If you are configuring a clustered instance of Confluence as a trusted application, you need to set up JIRA to receive requests from each Confluence node. If you do not set up each node appropriately, users may not be able to view any JIRA information in Confluence (e.g. a jiraissue macro request). You can set this up by either:
        - specifying each individual IP address for each node of the cluster separated by commas, e.g. `172.16.0.10, 172.16.0.11, 172.16.0.12` , or
        - specifying the IP address for your clustered Confluence instance using wildcards e.g. `172.16.0.*`
9. In the **'URL Paths to Allow'** field, type the JIRA URLs that the application will be allowed to access. Each URL corresponds to a particular JIRA function.By default, the following will be included:
    - '`/sr/jira.issueviews:searchrequest`' — This allows the application to search for JIRA issues.
    - '`/secure/RunPortlet`' — This allows the application to access JIRA dashboard portlets.
10. Click the **'Add'** button.
11. The **'Trusted Applications'** page will be displayed, with your new trusted application now shown in the list.

Project Management

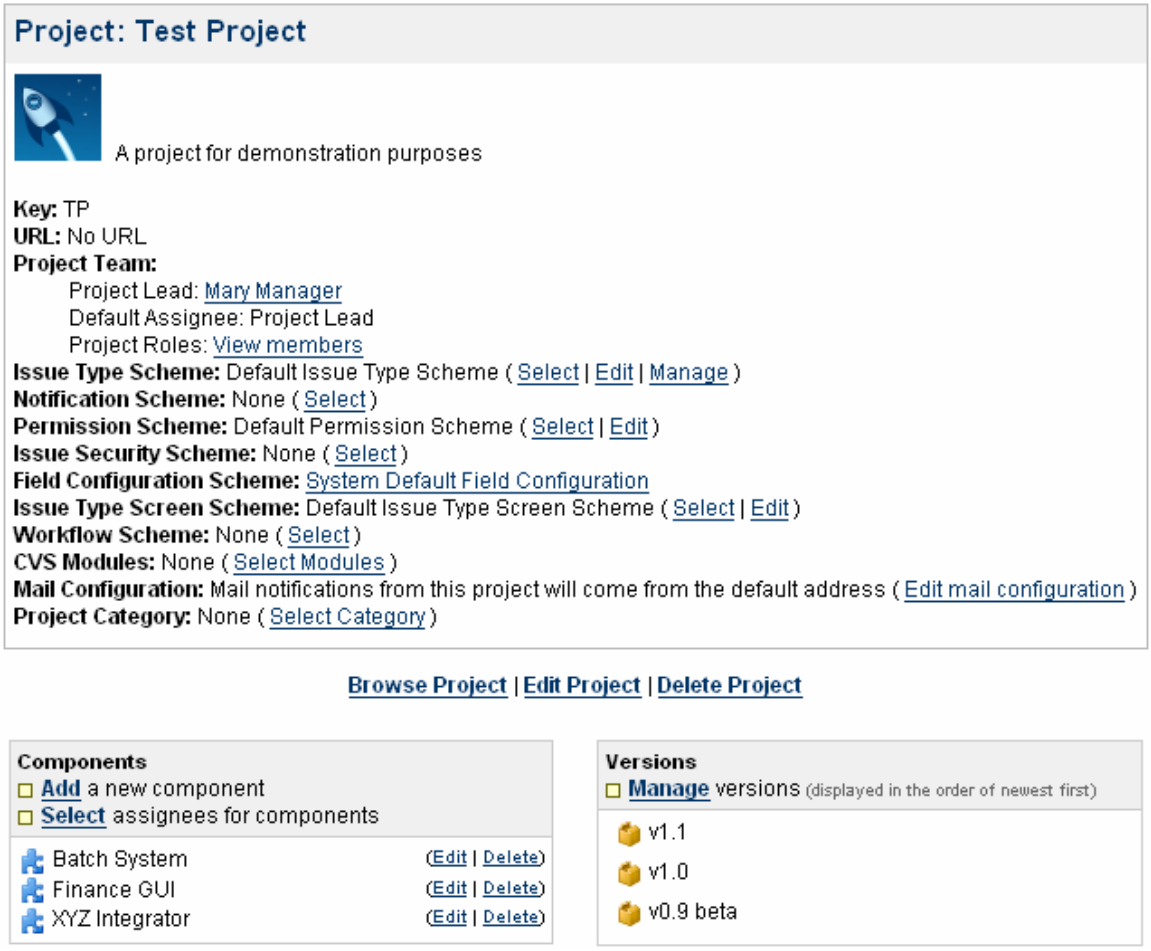
- Defining a Project
- Managing Project Role Membership
- Defining a Component
- Managing Versions
    - Creating Release Notes
- Configuring Project Keys

Defining a Project

To configure a project in JIRA:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Select an existing project, or click **'Add Project'** to add a project.

Here is what a project looks like once created:



A project's configuration settings are as follows:

- **Avatar** — an image (48x48 pixels) that represents the project. You can either use the default image, i.e.:

  or choose a different image. If you prefer not to use an image, simply upload a transparent pixel.
- **Description** — an optional description of this particular project. You can include HTML, but make sure all your tags are closed.
- **Key** — a 'key' unique to this project (eg. 'WEB'), which specifies the first few letters of this project's issue keys (e.g. 'WEB-100'). We recommend that you define a key that describes the project and is easy to type (as it prefixes each issue in the project). Please note that the key is shown to users who do not have permission to see the project and cannot be changed once the project exists.
- **URL** — an optional URL associated with this project, eg. pointing to project documentation.
- **Project Team:**
  - **Project Lead** — user fulfilling the role of project leader. Used as the 'Default Assignee' (see below), and potentially elsewhere in JIRA (e.g. in permission schemes, notification schemes, issue security schemes and workflows).
  - **Default Assignee** — the user to whom issues in this project are initially assigned when created. Can be either the 'Project Lead' (above), or, if **Allow unassigned issues** is set to 'On' in JIRA's general configuration, 'Unassigned'. Note: in JIRA Enterprise there are also default component assignees.
  - **Project Roles** — members are users/groups who fulfil particular functions for this project. Project roles are used in permission schemes, notification schemes, issue security schemes and workflows.
- **Issue Type Scheme** (Enterprise and Professional only) — the project's issue type scheme determines which issue types apply to this project.
- **Notification Scheme** — the project's notification scheme determines who receives email notifications of changes to issues in this project.
- **Permission Scheme** — the project's permission scheme determines who has permission to view or change issues in this project.
- **Issue Security Scheme** (Enterprise only) — the project's issue security scheme determines what visibility levels issues in this project can have (see issue-level security).

- **Field Configuration Scheme** (Enterprise only) — the project's field configuration scheme determines which field configuration applies to issue types in this project. (A field configuration determines each field's overall visibility, requiredness, formatting (wiki/rich-text or plain) and help-text).
- **Screen Scheme** (Professional only) — the project's screen scheme determines which screens are displayed for different issue operations (view, edit, create);
  or:
  Issue Type Screen Scheme (Enterprise only) — the project's issue type screen scheme determines which screens are displayed for different issue operations (view, edit, create), for different issue types.
- **Workflow Scheme** (Enterprise only) — the project's workflow scheme determines which workflows (issue state transitions) apply to issue types in this project.
- **CVS Modules** — configures CVS integration for this project.
- **Mail Configuration** — specifies the 'From' address for emails sent from this project. Only available if an SMTP email server has been configured in JIRA.
- **Project Category** (Enterprise only) — a logical category to group this project into. Useful for managing multiple related projects. New categories can be created via the 'Project Categories' link in the 'Administration' menu.
  As well as:
- **Components** — logical groups that this project's issues can belong to. See the component management page for details.
- **Versions** — versions defined in the project. See the version management page for details.

A note about Project Administrators

A JIRA project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'JIRA Administrator' permission.

A project administrator can:

- Edit the project name ('Test Project' in the screenshot above)
- Edit the project description ('A project for demonstration purposes' in the screenshot above)
- Edit the project avatar image
- Edit the URL
- Edit the Project Lead
- Edit project role membership *
- Define project components
- Define project versions

** Enterprise edition only. In Professional and Standard editions, only a global administrator can edit project role membership.*

Additional Resources

- Adding a Project tutorial video — Watch this short tutorial video to see how a project is added in JIRA. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Managing Project Role Membership

A JIRA project role is a flexible way to associate users and/or groups with a particular project.

Unlike groups, which have the same membership throughout JIRA, project roles have specific members for each project. Users may play different roles in different projects.

This page contains instructions for managing membership of existing project roles. For information on creating and using project roles, please see Managing project roles.

On this page:

- Viewing project role members
- Assigning a user to a project role
- Removing a user from a project role
- Assigning a group to a project role
- Removing a group from a project role

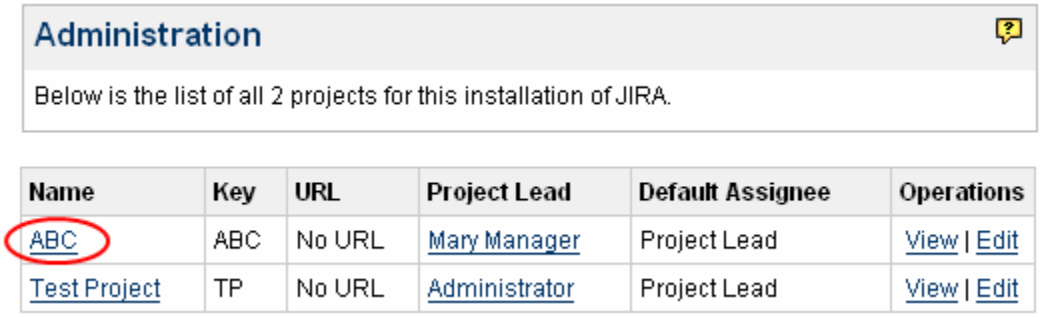
Viewing project role members

To see which users and groups belong to each project role for a particular project:

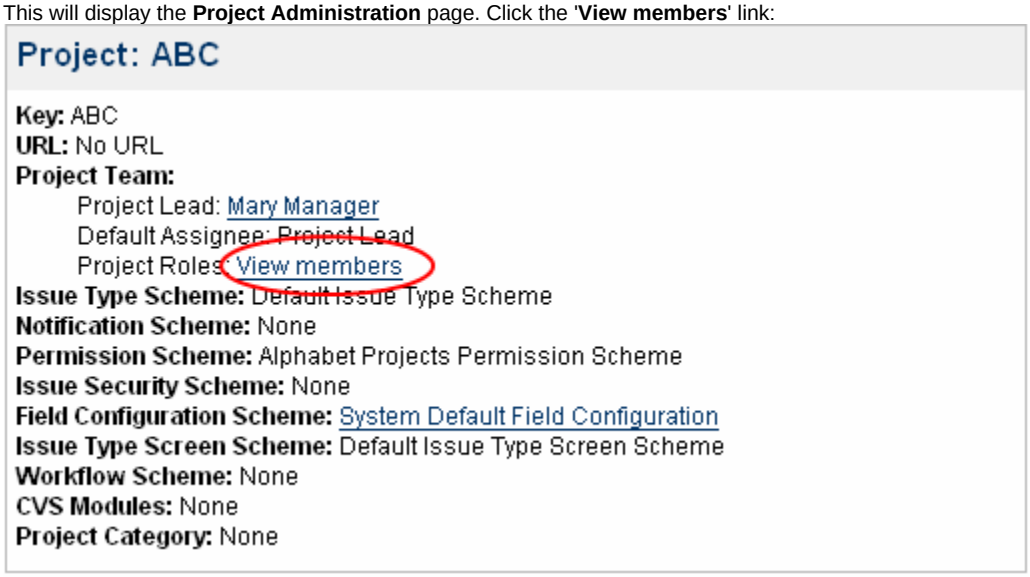
1. The first step depends on which edition of JIRA you are using:
   - *If you are using JIRA Enterprise edition,*
     a. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission **'Administer Project',** but not necessarily the global permission **'JIRA Administrators'.**)
     b. Click the **'Administration'** link on the top bar.

- *Or, if you are using JIRA Professional or Standard edition,*
    a. Log in as a user with the **'JIRA Administrators'** global permission.
    b. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.

2. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.



3. This will display the **Project Administration** page. Click the '**View members**' link:

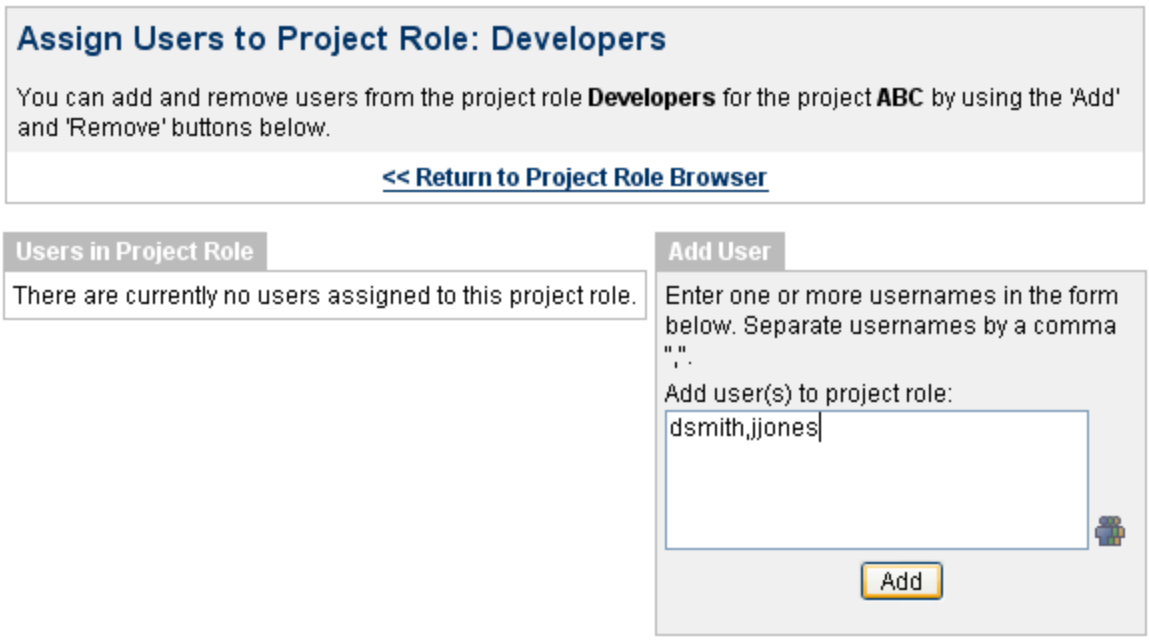


4. This will display the **Manage Project Role Membership** page, showing the project role members for this project: !project_roles-manage_roles_project.png|title="View project roles"!From this page you can assign users/groups to and remove them from project roles, as described below.

> *A project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'JIRA Administrator' permission.

Assigning a user to a project role

1. Open the '**Manage Project Role Membership**' page as described in 'Viewing project role members' (above).
2. Click the '**Edit**' link in the '**Users**' column for the project role you wish to edit.
3. This will display the '**Assign Users to Project Role**' page:



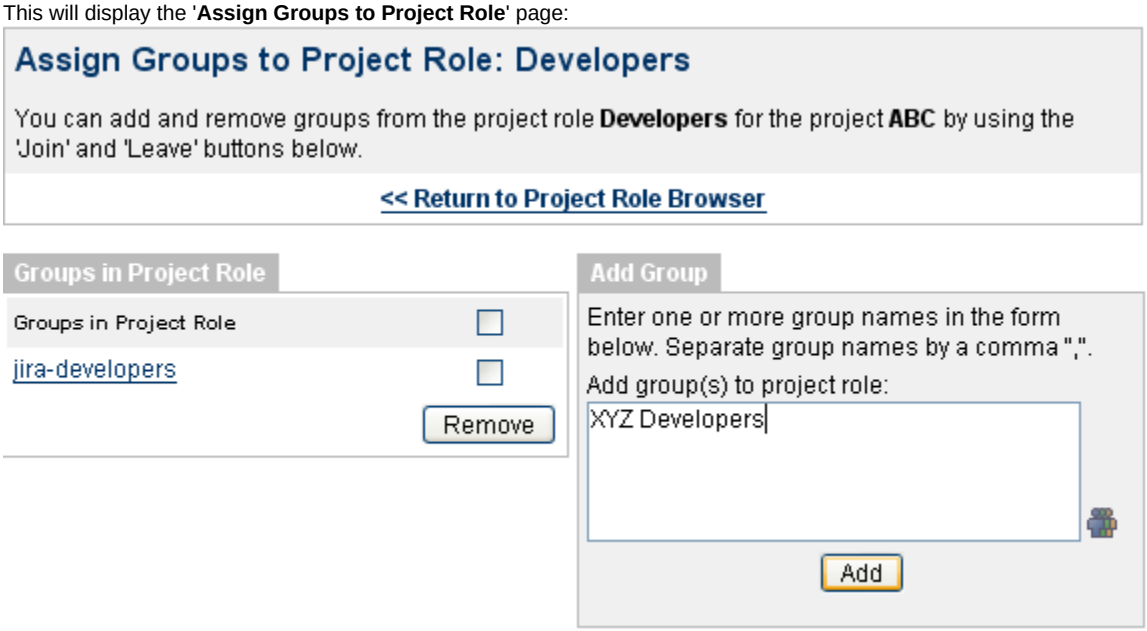
The users currently in the project role are listed on the left-hand side of the page. Type the username(s) in the '**Add User**' box on the right-hand side of the page, then click the '**Add**' button.

Removing a user from a project role

1. Open the '**Manage Project Role Membership**' page as described in 'Viewing project role members' (above).
2. Click the '**Edit**' link in the '**Users**' column for the project role you wish to edit.
3. This will display the '**Assign Users to Project Role**' page. The users currently in the project role are listed on the left-hand side of the page. Tick the user(s) you wish to remove from the project role, then click the '**Remove**' button.

Assigning a group to a project role

1. Open the '**Manage Project Role Membership**' page as described in 'Viewing project role members' (above).
2. Click the '**Edit**' link in the '**Groups**' column for the project role you wish to edit.
3. This will display the '**Assign Groups to Project Role**' page:



The groups currently in the project role are listed on the left-hand side of the page. Type the group name(s) in the '**Add Group**' box on the right-hand side of the page, then click the '**Add**' button.

Group membership can only be viewed/edited by people with the global 'JIRA Administrator' permission. Project administrators may therefore prefer to assign users, rather than groups, to their project roles.

Removing a group from a project role

1. Open the '**Manage Project Role Membership**' page as described in 'Viewing project role members' (above).
2. Click the '**Edit**' link in the '**Groups**' column for the project role you wish to edit.
3. This will display the '**Assign Groups to Project Role**' page. The groups currently in the project role are listed on the left-hand side of the page. Tick the group(s) you wish to remove from the project role, then click the '**Remove**' button.

A project role need not have any user or group assigned to it, although project administrators should be careful with this. Depending on how a project role is utilised (e.g. if the project's permission scheme is using project roles), it is possible that not having anyone in a particular project role could make some project activities unavailable.

Defining a Component

Components are sub-sections of a project. They are used to group issues within a project into smaller parts. The available operations for components are:

- Add — make new components under which issues can be classed.
- Delete — remove a component from a project.
- Edit — Update/change the details of a particular component.
- Select Default Assignee — In **Enterprise** edition you can set the default assignee for issues created in a particular component.

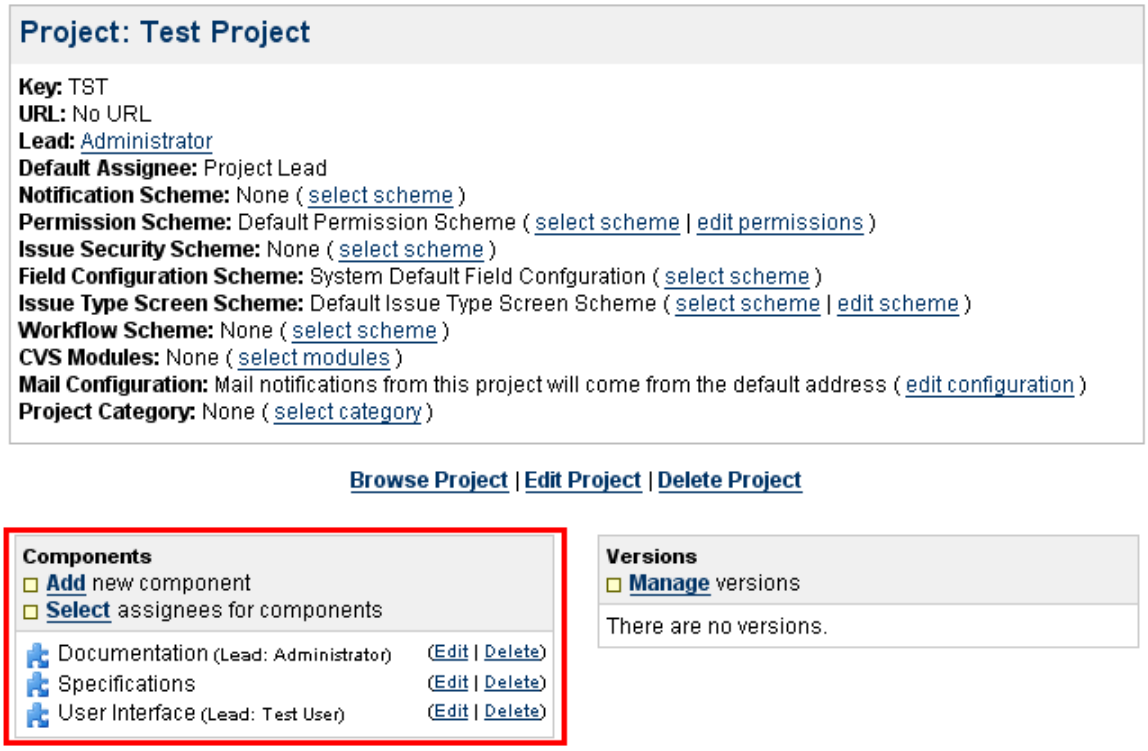
On this page:

- Managing project components
- Adding a new component
- Selecting a Default Assignee
    - Default Assignee Options
- Editing a component
- Deleting a component

Managing project components

All component management operations are available from the Component section of the Project Admin Page.

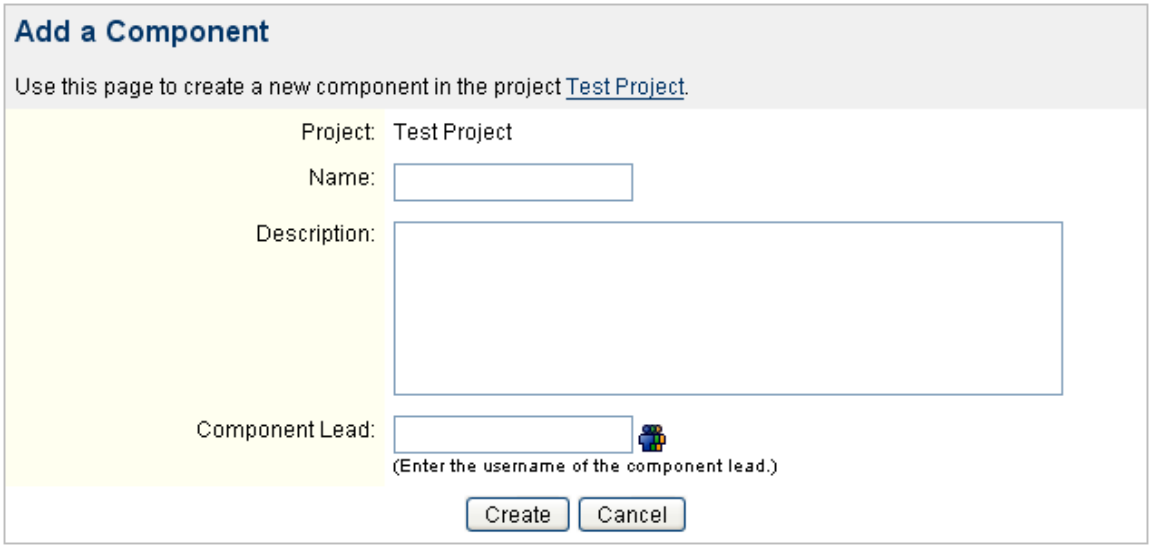
1. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission **'Administer Project',** but not necessarily the global permission **'JIRA Administrators'.**)
2. Click the **'Administration'** link on the top bar.
3. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.
4. You will now see a page displaying the project details. On the lower left, the **'Components'** section displays a summary of the project's components along with links to add, edit and remove components (as described below).



If you have created a new project and have not assigned a permission scheme with it on creation, then you will not see the above display. Instead, the **'Components'** section will say "There are no components at the moment".

Adding a new component

1. In the **'Components'** section (see above), click the 'Add' link.
2. The 'Add a Component' screen will be displayed. Type the name of your new component.
3. You can also optionally enter the component description and/or assign a user to be the component lead.
4. Click the "Add" button. The component summary list is updated immediately and you will be redirected to the project admin page.

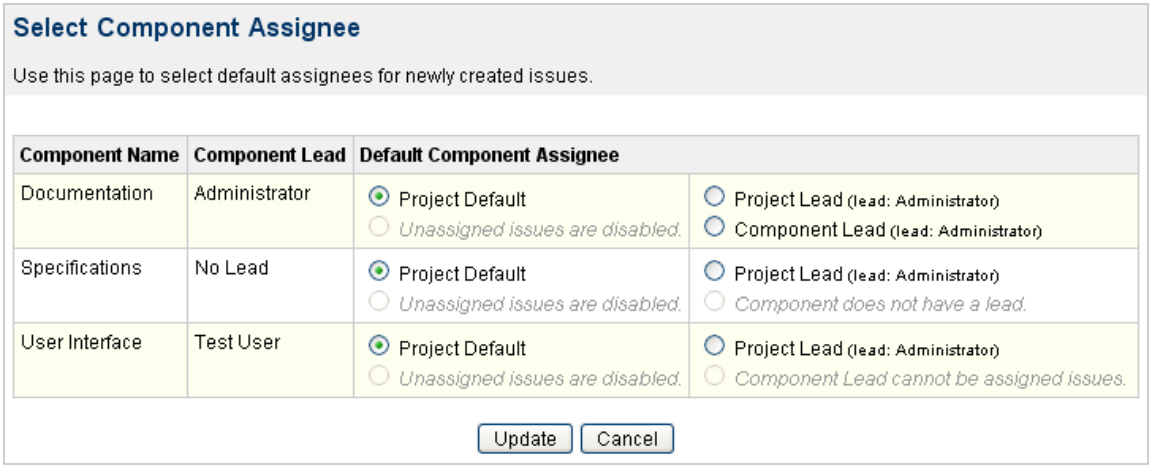


Selecting a Default Assignee

In **JIRA Enterprise edition** it is possible to extend the default assignee of an issue to be component specific instead of project specific.

1. In the **'Components'** section (see 'Managing project components' above), click the 'Select' link. A 'Select Component Assignee' pane will appear on the Project Admin Page.
2. For each component, select the assignee to whom you want to have new issues in the component assigned by default. See the list of

options below.
3. Click on the '**Update**' button.



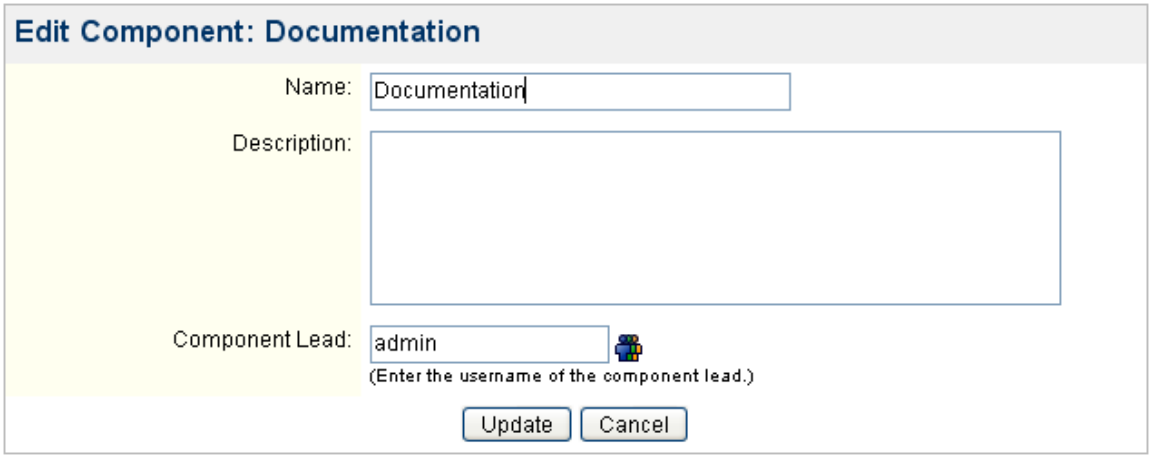
In the event that the default assignees of components clash, the assignee will be set to the default assignee of the component that is first alphabetically.

Default Assignee Options

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

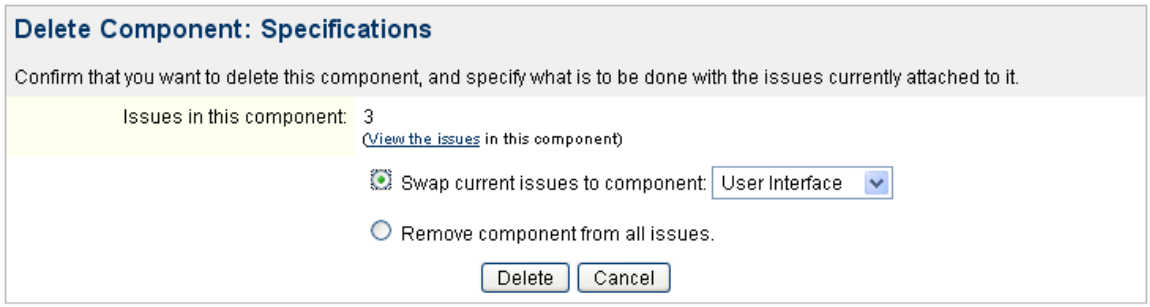
Editing a component

1. In the **'Components'** section (see 'Managing project components' above), click the 'Edit' link at the right of a the component you wish to edit.
2. This will bring you to the "Edit Component" page. Here, it is possible to edit the version name, description and lead.
3. Press the 'Update' button.
4. On completion of the update operation, you are returned to the project admin page - with an updated component list reflecting the changes made.



Deleting a component

1. In the **'Components'** section (see 'Managing project components' above), click the '**Delete**' link at the right of the component you wish to delete.
2. This will bring you to the "Delete Component" page.
3. On this page you can specify the action to be taken regarding the issues in this component. You can either associate these issues with another active component, or have the references removed.
4. Press the '**Delete**' button.
5. On completion of the delete operation, you are returned to the project admin page - with an updated component list reflecting the changes made.



Managing Versions

Versions are points-in-time for a project. They help you schedule and organise your releases. Once a version is created, the following reports are useful:

- Road Map report — gives you a view of upcoming versions
- Change Log report — gives you a review of released versions

The Change Log and Road Map reports are driven by the 'Fix For Version' field on each issue.

Versions can be:

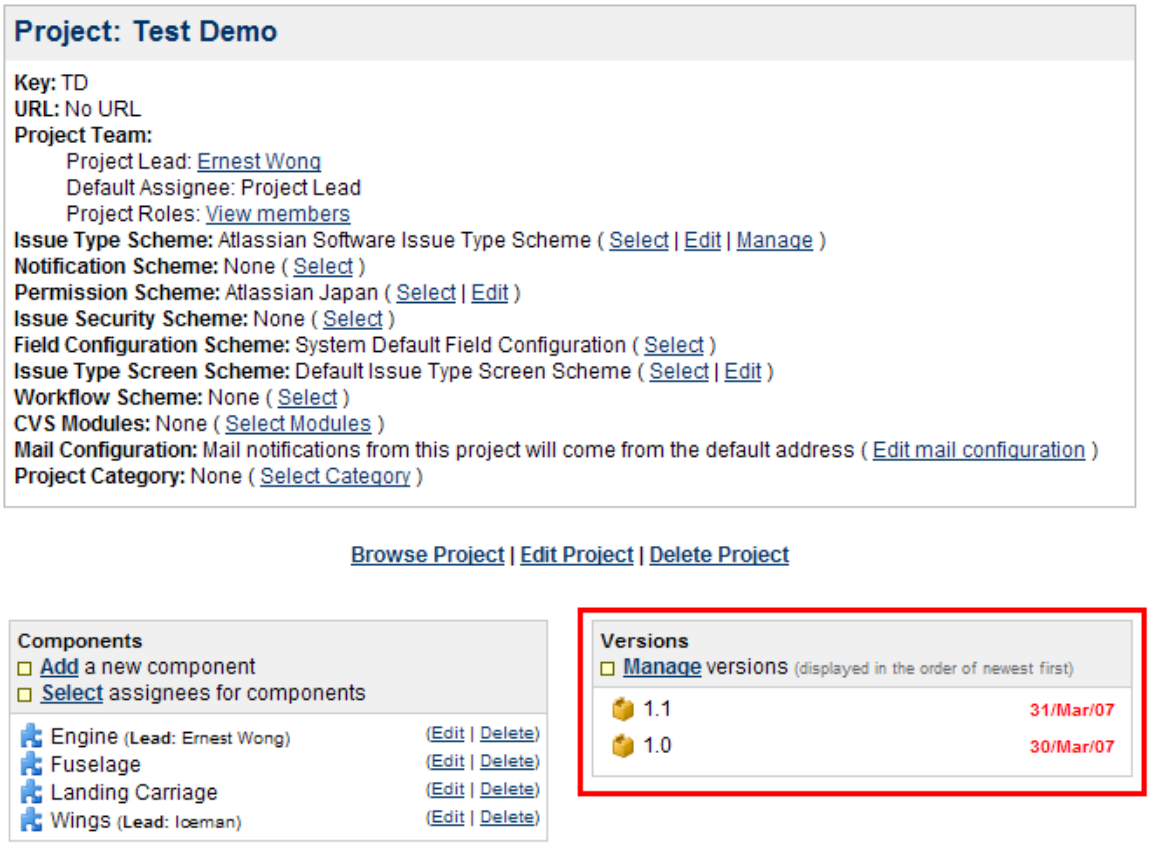
- Added — create a new version against which issues can be aligned.
- Released — mark a version as released. This changes the Road Map report, Change Log report and some issue fields' drop-downs.
- Rescheduled — re-arrange the order of versions.
- Archived — hide an old version from the Road Map and Change Log reports, and in the JIRA User Interface.
- Merged — combine multiple versions into one.

On this page:

- Managing a project's versions
  - Version status
- Adding a new version
- Releasing a version
- Archiving a version
- Merging multiple versions
- Editing a version's details
- Deleting a version
- Rescheduling a version

Managing a project's versions

1. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission **'Administer Project',** but not necessarily the global permission **'JIRA Administrators'.**)
2. Click the **'Administration'** link on the top bar.
3. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.
4. You will now see a page displaying the project details. On this page all the configurable actions available on the project are easily accessible. On the lower right, a summary of the versions is displayed along with the link to the version management interface. The summary indicates the version status and the scheduled release date for that version.



> **Note**
> If you have created a new project and have not assigned a permission scheme with it on creation, then you will not see the above display. Instead under versions it will say "There are no versions at the moment".

5. Click the 'Manage' link to display the 'Manage Versions' screen, which shows a list of versions and each version's status. From here you can perform the operations described below.

Version status

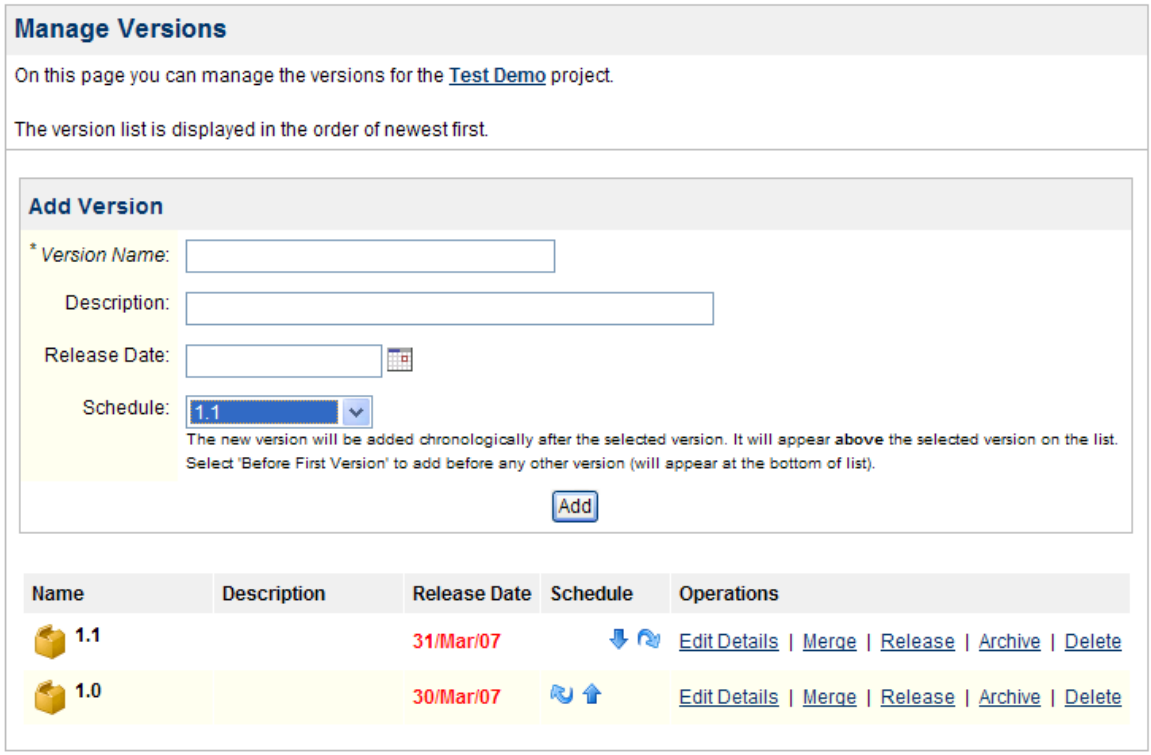
Each version can have any of the following four statuses:

- **Released** — a bundled package
- **Unreleased** — an open package
- **Archived** — a semi-transparent package
- **Overdue** — the release date is highlighted

> The status affects where the version appears in drop-down lists for version-related issue fields ('Fix For Version' and 'Affects Version').

Adding a new version

1. The "Add New Version" form is located at the bottom of the **'Manage Versions'** screen.
2. From here, you enter the name for the version. The name is treated as a plain string by JIRA, so it can be simple numeric, e.g. "2.1", it can be complicated numeric, e.g. "2.1.3", or it can be something zany like the project's internal code-name, e.g. "Memphis".
3. Optional details such as the version description and release date can be also be specified.
4. It is also possible to 'schedule' the new version by selecting its position in the version list. The new version is added after the selected version - or it can be placed at the start of the list by selecting 'First'.
5. Click on the "Add" button. The version management list is updated immediately, with the newly created version added in the specified position.



Note
At present, the version release date is only used to indicate the scheduled release date for a version. Currently JIRA only uses the release date for reporting purposes, but in future releases it is intended to add further functionality to take advantage of this data.

Releasing a version

1. On the **'Manage Versions'** screen, click the '**Release**' link available in the **'Operations'** column for the version you are interested in.
2. If there are any issues set with this version as their 'Fix For' version, JIRA allows you to choose to change the 'Fix For' version if you wish. Otherwise, the operation will complete without modifying these issues.
3. This operation immediately updates the specific version as 'released' throughout JIRA.
4. The version list indicates the version 'released' status with the bundled package icon. The **'Unrelease'** operation replaces the **'Release'** operation in the **'Operations'** column.
5. To un-release a version, simply click on the **'Unrelease'** link in the **'Operations'** column.

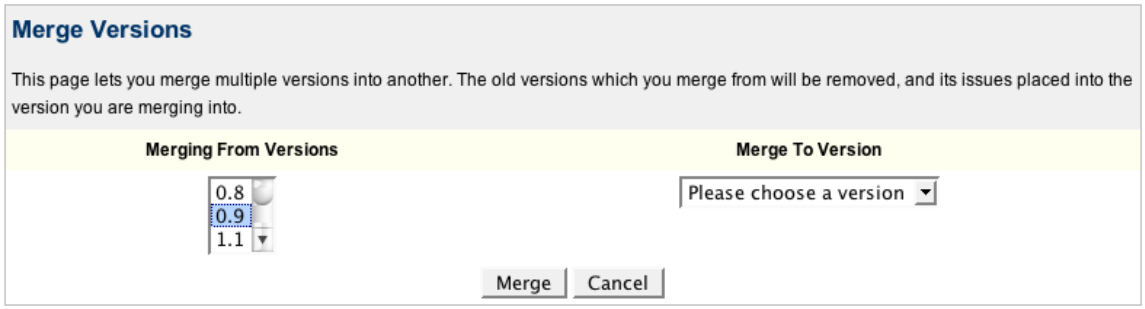
Archiving a version

1. On the **'Manage Versions'** screen, click the **'Archive'** link available in the **'Operations'** column for a version you are interested in.
2. This operation immediately updates the specific version as 'archived' throughout JIRA.
3. The version list indicates the version 'archived' status with a semi-transparent icon. The list of available operations is replaced with the 'Unarchive' operation. No further changes can be made to this version unless it is un-archived. Also it is not possible to remove any existing archived versions from an issue's affected and fix version fields or add any new archived versions.
4. To un-archive a version, simply click on the 'Unarchive' link in the operations column.

Merging multiple versions

1. On the **'Manage Versions'** screen, click the 'Merge' link available in the **'Operations'** column for a version you are interested in.
2. This will take you to the 'Merge Versions' page.On this page are two select lists - both listing all un-archived versions. The specified version is highlighted in the 'Merging From Versions' select list on the left. It is possible to select further versions you wish to merge from.

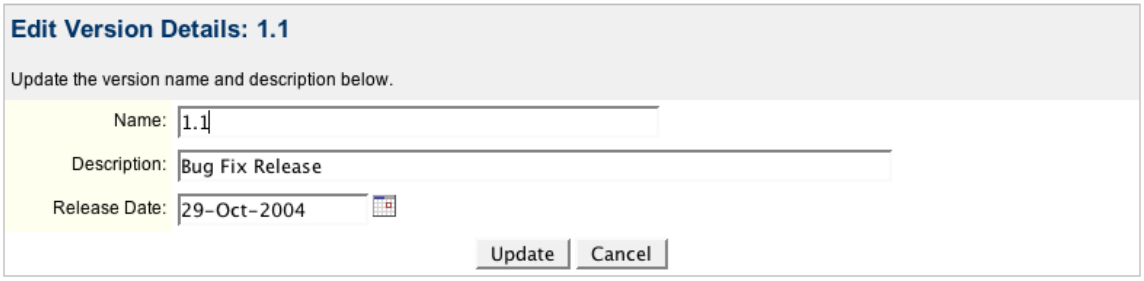
Versions selected on this list will be removed from the system. All issues associated with these versions will be updated to reflect the new version selected in the 'Merge To Version' select list on the right. It is only possible to select one version to merge to.



3. Click on the 'Merge' button. You will be shown a confirmation Page. Click on 'Merge' to complete the operation.
4. On completion of the merge operation, you are returned to the version management interface. The version list has been updated to reflect the changes that occurred in the merge operation.

Editing a version's details

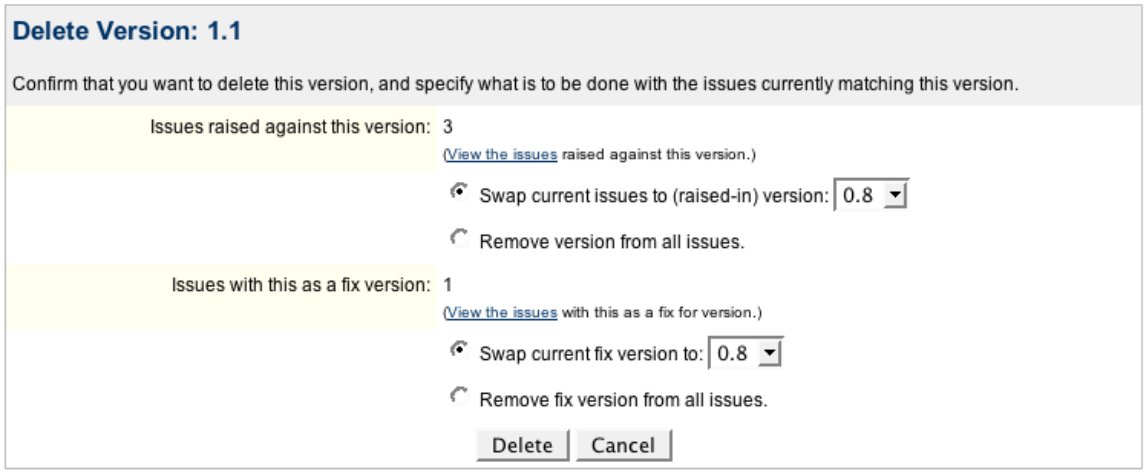
1. On the **'Manage Versions'** screen, click the **'Edit Details'** link available in the **'Operations'** column for the version you are interested in.
2. This will bring you to the **"Edit Version: <Version>"** page. Here, you can edit the version's Name, Description and Release Date:



3. Click the **'Update'** button to save your changes.
4. On completion of the update operation, you are returned to the **'Manage Versions'** screen — with an updated version list reflecting the changes made.

Deleting a version

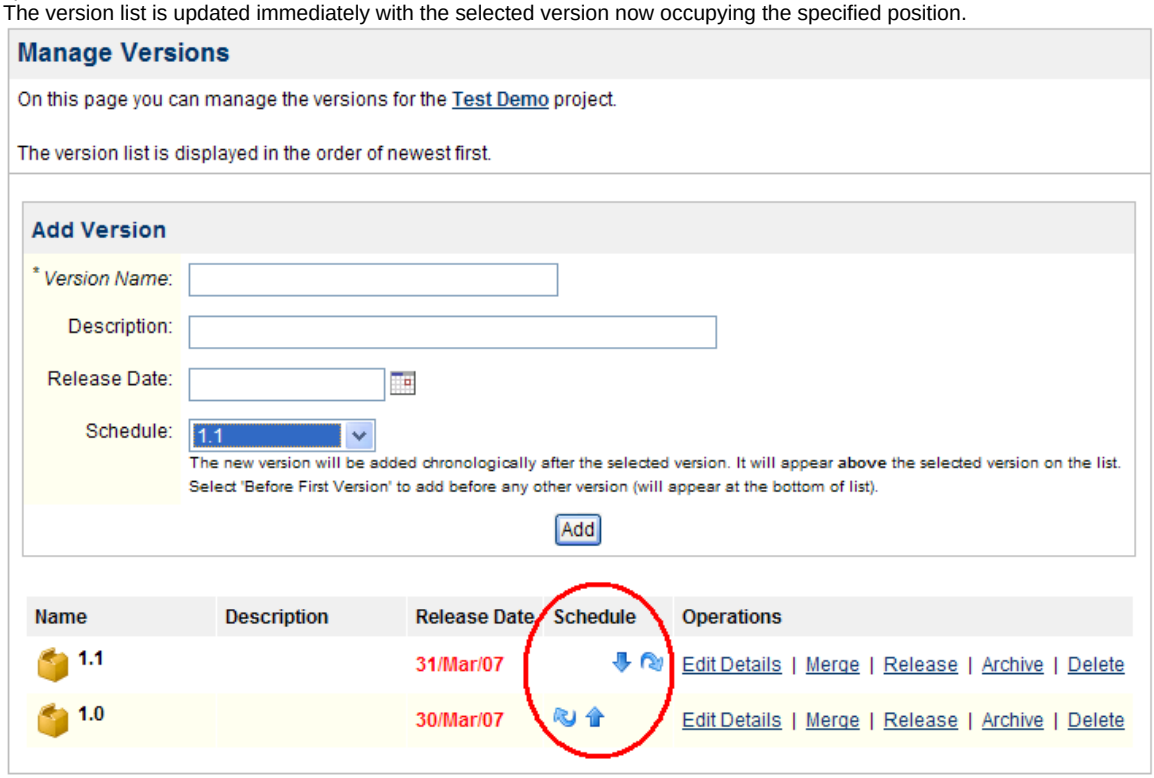
1. On the **'Manage Versions'** screen, click the **'Delete'** link in the **'Operations'** column for the version you wish to delete.
2. This will bring you to the **"Delete Version: <Version>"** confirmation page.
3. From here, you can specify the actions to be taken for issues associated with the version to be deleted. You can either associate these issues with another version, or simply remove references to the version to be deleted.



4. Click the **"Delete"** button.
5. On completion of the deletion, you are returned to the **'Manage Versions'** screen — with an updated version list reflecting the changes made.

Rescheduling a version

1. On the **'Manage Versions'** screen, re-scheduling operations are available through the 'Schedule' column.
2. It is possible to move a version up/down a position or to the start/end of the list by clicking on the specific arrow icon associated with the specific version row.
3. The version list is updated immediately with the selected version now occupying the specified position.



Creating Release Notes

JIRA provides the functionality to create release notes for a specific version of a project. The release notes contain all issues within the specified project that are marked with a specific "Fix For" version. The release notes can also be generated in a number of formats (e.g. HTML, plain text, etc.) so as they can be included in various documents.

At present, two example format templates are provided - HTML and Text - using Velocity templates. Further format templates can be created and added to the system.

Generating Release Notes

1. Select the '**Browse Projects**' option from the main menu
2. Select the project from which you wish to create release notes.
3. Select '**Road Map**'.
4. Select '**Release Notes**'.
5. Select the required format of the release notes — HTML and plain text format templates are provided.
6. Select the required project version for which the release notes will be generated.
7. Selecting the '**Create**' button will generate the release notes using the specified template in the specified format. The release notes are displayed as HTML and also within a text area in the actual format selected - allowing the contents to be copied for inclusion in another document.

It is also possible to create a default set of release notes using the HTML format template. By selecting the **Roadmap** option from the **Reports** menu, it is possible to create the HTML release notes for each un-released version by selecting the '**Release Notes**' link after each version title.

Adding a New Format Template

1. Create a Velocity template similar in content to that of the examples provided — `releasenotes-text.vm` and `releasenotes-html.vm`. Consult the JIRA API documentation and the Jakarta Velocity User Guide.
2. The title within the template should be modified along with the code within the text area. The other sections of the template do not need to be modified.
3. Add the new format template to the list within the `jira-application.properties` file. A corresponding entry must be made in both the `jira.releasenotes.templatenames` and the `jira.releasenotes.templates` lists. It is also necessary that the entries in both lists are in the same order.
4. Restart JIRA.

5. The new format template is available for selection as a release note format template.

Also see the tutorial on How to Create a Custom Release Notes Template Containing Comments.

Configuring Project Keys

JIRA provides the ability to specify the format of project keys within the system. This is achieved by defining a regular expression 'rule' that governs the valid project key format.

Project Key Pattern

Through the property `jira.projectkey.pattern`, the administrator can specify a Perl5 regular expression defining the rule for a valid project key. This property can be found in the `jira-application.properties` file. During project creation, the user must specify a project key that conforms to this rule.

JIRA prepends the regular expression specified with '^' and closes it with '$' for an exact matching rule within the system. The project key must only be allowed to contain ASCII characters, as it is used in HTTP GET requests.

Editing the key pattern

This can be done by editing jira-application.properties. You will then need to restart JIRA (JIRA Standalone) or rebuild the JIRA webapp and redeploy in your app server.

Project Key Details

The `jira-application.properties` file also contains the following properties:

- `jira.projectkey.description` — a configurable description (to match the project key pattern) displayed on project creation
- `jira.projectkey.warning` — a configurable validation warning (to match the project key pattern)

It is not possible to configure the issue key pattern as JIRA expects this key to conform to specific rules.

Further information on Perl5 is available here.

Configuring Security

When configuring security for your JIRA instance, there are two areas to address:

- security within JIRA itself
- security in the external environment

Configuring security within JIRA

JIRA has a flexible security system which allows you to configure who can access JIRA, and what they can do/see within JIRA.

There are five types of security within JIRA:

1. Global permissions — these apply to JIRA as a whole (e.g. who can log in).
2. Project permissions — organised into permission schemes, these apply to projects as a whole (e.g. who can see the project's issues ('Browse' permission), create, edit and assign them).
3. Issue security levels *(Enterprise Edition only)* — organised into security schemes, these allow the visibility of individual issues to be adjusted, within the bounds of the project's permissions.
4. Comment visibility — allows the visibility of individual comments (within an issue) to be restricted.
5. Work-log visibility — allows the visibility of individual work-log entries (within an issue) to be restricted.

Configuring security in the external environment

If your JIRA instance contains sensitive information, you may want to configure security in the environment in which your JIRA instance is running. Some of the main areas to consider are:

- Database:
  - If you are using an external database as recommended for production systems (i.e. you are not using the HSQL database that is bundled with JIRA Standalone), you should restrict access to the database that your JIRA instance uses.
  - If you are using the embedded HSQL database that is bundled with JIRA Standalone, you should restrict access to the directory

in which you installed JIRA. (Note that the user which your JIRA instance is running as will require full access to this directory.)
- File system — you should restrict access to the following directories (but note that the user which your JIRA instance is running as will require full access to these directories):
  - Index directory
  - Attachments directory
- SSL — if you are running your JIRA instance over the Internet, you may want to consider using SSL.

Managing Global Permissions

Global permissions are system wide.

See also project permissions, which apply to individual projects.

Global permissions are granted to groups of users.

This table lists the different global permissions and the functions they secure:

Global Permission	Explanation
JIRA System Administrators	Permission to perform all JIRA administration functions.
JIRA Administrators	Permission to perform most JIRA administration functions (see list of exclusions below).
JIRA Users	Permission to login to JIRA. (Note: Granting the **JIRA Users** permission to a group also means that all newly created users will be automatically added to that group.)
Browse Users	Permission to view a list of all JIRA user names and group names. Used for selecting users/groups in popup screens (such as the 'User Picker').
Create Shared Object	Permission to share a filter or dashboard globally or with groups of users. *(This applies only to JIRA Professional and Enterprise editions.)*
Manage Group Filter Subscriptions	Permission to manage (create and delete) group filter subscriptions. *(This applies only to JIRA Professional and Enterprise editions.)*
Bulk Change	Permission to execute the bulk operations within JIRA: Bulk Edit*, Bulk Move*, Bulk Workflow Transition, Bulk Delete* (*subject to project-specific permissions.) Note: The decision to grant the **Bulk Change** permission should be considered carefully. This permission grants users the ability to modify a collection of issues at once. For example, in JIRA installations configured to run in Public mode (i.e. anybody can sign up and create issues), a user with the **Bulk Change** global permission and the **Add Comments** project permission could comment on *all* accessible issues. Undoing such modifications may not be possible through the JIRA application interface and may require changes made directly against the database (which is not recommended).

About 'JIRA System Administrators' and 'JIRA Administrators'

People who have the **'JIRA System Administrators'** permission can perform all of the administration functions in JIRA, while people who have only the **'JIRA Administrators'** permission cannot perform functions which could affect the application environment or network. This is useful for organisations which need to delegate some administrative privileges (e.g. creating users, creating projects) to particular people, without granting them complete rights to administer the JIRA system.

By default, the `jira-administrators` group has both the **'JIRA Administrators'** permission *and* the **'JIRA System Administrators'** permission. If you need some people to have only the **'JIRA Administrators'** permission (and not the **'JIRA System Administrators'** permission), you will need to use two separate groups, e.g.:

1. Create a new group (e.g. called `jira-system-administrators`).
2. Add to the `jira-system-administrators` group everyone who needs to have the **'JIRA System Administrators'** permission.
3. Grant the **'JIRA System Administrators'** permission to the `jira-system-administrators` group.
4. Remove the **'JIRA System Administrators'** permission from the `jira-administrators` group.
5. *(Optional, but recommended for ease of maintenance)* Remove from the `jira-administrators` group everyone who is a member of the `jira-system-administrators` group.

People who have the **'JIRA Administrators'** permission (and not the **'JIRA System Administrators'** permission) cannot do the following:

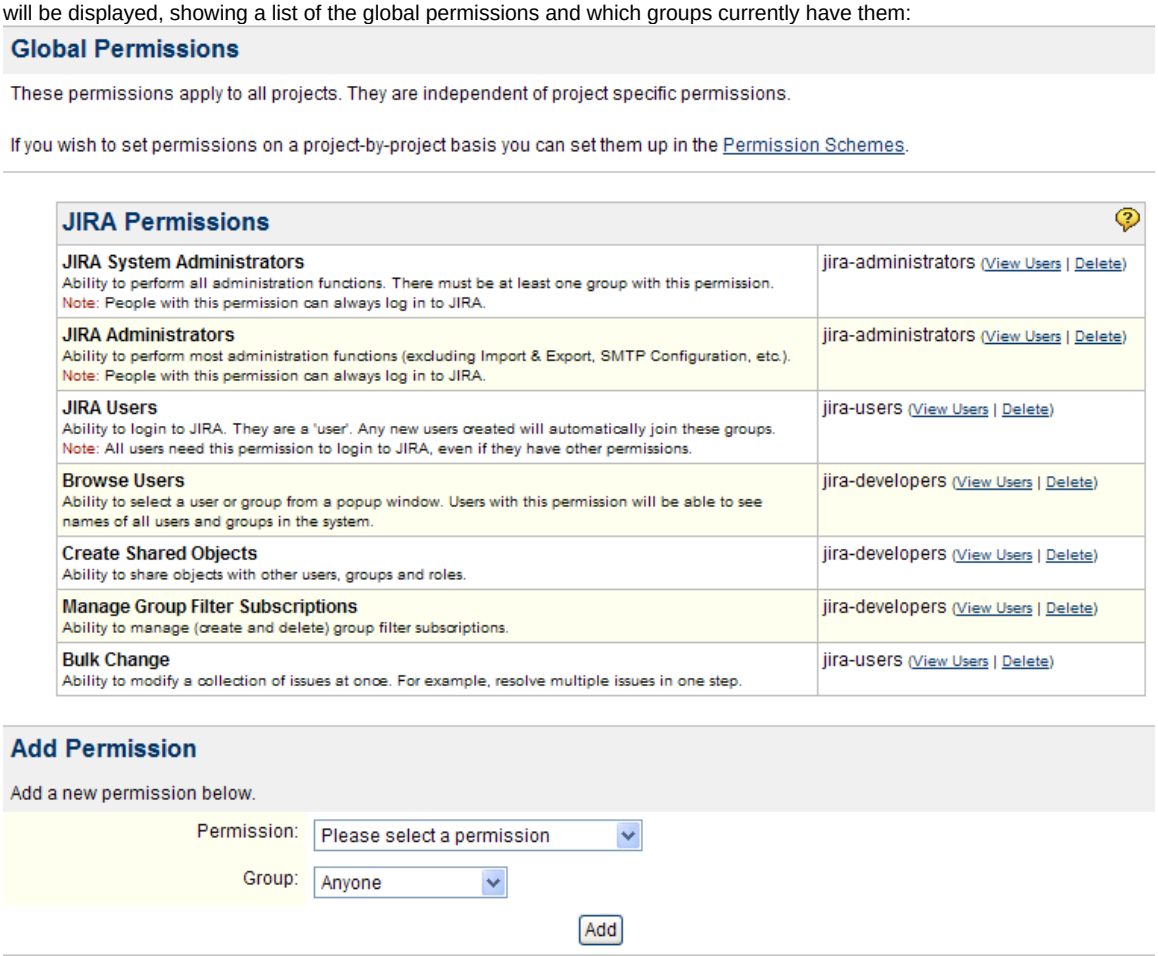
- Configure SMTP email (but note that they can fully administer email notification schemes).

- Configure a CVS source code repository (but note that they can associate a project with a configured repository).
- Configure listeners.
- Configure services.
- Change the index path (but note that they can reindex and optimise the index).
- Run the integrity checker.
- Access logging and profiling information.
- Access the scheduler.
- Export/backup JIRA data to XML.
- Import/restore JIRA data from XML.
- Import data from external systems (Bugzilla, Mantis, FogBugz, Excel/CSV).
- Import XML workflows into JIRA.
- Enable attachments or set the attachment path (but note that they can set the size limit and enable thumbnails).
- Enable trackbacks.
- Run Jelly scripts.
- Configure LDAP integration.
- Configure trusted applications.
- Access license details.
- Grant/revoke the **'JIRA System Administrators'** global permission.
- Edit (or Bulk Edit) groups that have the **'JIRA System Administrators'** global permission.
- Edit, change the password of or delete a user who has the **'JIRA System Administrators'** global permission.

It is recommended that people who have the **'JIRA Administrators'** permission (and not the **'JIRA System Administrators'** permission) are not given direct access to the JIRA filesystem or database.

Granting global permissions

1. Log in as a user with the **'JIRA Administrators'** global permission (or the **'JIRA System Administrators'** global permission, if you need to grant the **'JIRA System Administrators'** global permission).
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the **Administration** box on the dashboard.
3. In the panel on the left, under the title **'Global Settings',** click the link labelled **'Global Permissions'.** The **'Global Permissions'** page will be displayed, showing a list of the global permissions and which groups currently have them:



   The **'Add Permission'** box is shown below the list.
4. In the **'Permission'** drop-down list, select the global permission you wish to grant.
5. In the **'Group'** drop-down list, either:
   - select the group to which you wish to grant the permission; or

- if you wish to grant the permission to non logged-in users, select **'Anyone'** (not recommended for production systems). Note that the **'JIRA Users'** permission (i.e. permission to log in) cannot be granted to **'Anyone'** (i.e. to non logged-in users) since this would be contradictory.

> If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to grant login permissions (i.e. jira-users permission) to any further groups without first reducing the number of users with login permissions.

Managing Project Permissions

On this page:

- Permission Schemes
  - What is a Permission Scheme?
  - Why Permission Schemes?
- Creating a Permission Scheme
- Adding Users, Groups or Roles to a Permission Scheme
- Associating a Permission Scheme with a Project
- Deleting a Permission Scheme
- Copying a Permission Scheme
- Additional Resources

Project permissions are created within Permission Schemes, which are then assigned to specific projects.

Project permissions can be granted to:

- Individual users
- Groups
- Project roles
- Issue roles such as 'Reporter', 'Project Lead' and 'Current Assignee' (JIRA Enterprise Edition only)
- 'Anyone' (e.g. to allow anonymous access)
- A (multi-)user picker custom field.
- A (multi-)group picker custom field. This can either be an actual group picker custom field, or a (multi-)select-list whose values are group names.

The following table lists the different types of project permissions and the functions they secure. Note that, in JIRA Professional and Enterprise editions, project permissions can also be used in workflow conditions.

Project Permission	Explanation
Administer Projects	Permission to administer a project in JIRA. This includes the ability to edit project role membership (Enterprise Edition only), project components, project versions and some project details ('Project Name', 'URL', 'Project Lead', 'Project Description').
Browse Projects	Permission to browse projects, use the Issue Navigator and view individual issues (except issues that have been restricted via Issue Security). Users without this permission will not know that the project exists.
View Version Control	Permission to view the version control information (CVS, Subversion, etc) for issues. Note that for CVS, to view the Version Control information the project needs to be associated with at least one Repository.
Create Issues	Permission to create issues in the project. (Note that the Create Attachments permission is required in order to create attachments.) Includes the ability to create sub-tasks (if sub-tasks are enabled).
Edit Issues	Permission to edit issues (excluding the 'Due Date' field — see the Schedule Issues permission). Includes the ability to convert issues to sub-tasks and vice versa (if sub-tasks are enabled). Note that the Delete Issue permission is required in order to delete issues. The Edit Issue permission is usually given to any groups or project roles who have the Create Issue permission (perhaps the only exception to this is if you give **everyone** the ability to create issues — it may not be appropriate to give everyone the ability to edit too). Note that all edits are recorded in the Issue Change History for audit purposes.
Schedule Issues	Permission to set and edit the 'Due Date' of issues.
Move Issues	Permission to move issues from one project to another, or (in Enterprise Edition only) from one workflow to another workflow within the same project. Note that a user can only move issues to a project for which they have Create Issue permission.
Assign Issues	Permission to assign issues to users. (See also Assignable User permission below)
Assignable User	Permission to be assigned issues. (Note that this does not include the ability to assign issues; see Assign Issue permission above).

Resolve Issues	Permission to resolve and reopen issues. This also includes the ability to set the 'Fix For version' field for issues.
Close Issues	Permission to close issues. (This permission is useful where, for example, developers resolve issues and testers close them).
Modify Reporter	Permission to modify the 'Reporter' of an issue. This allows a user to create issues 'on behalf of' someone else.
Delete Issues	Permission to delete issues. Think carefully about which groups or project roles you assign this permission to; usually it will only be given to administrators. Note that deleting an issue will delete all of its comments and attachments, even if the user does not have the Delete Comments or Delete Attachments permissions. However, the Delete Issues permission does not include the ability to delete individual comments or attachments.
Link Issues	Permission to link issues together. (Only relevant if Issue Linking is enabled).
Set Issue Security	Permission to set the security level on an issue to control who can access the issue (Enterprise Edition only). Only relevant if issue security has been enabled.
View Voters and Watchers	Permission to view the voter list and watcher list of an issue.
Manage Watcher List	Permission to manage (i.e. view/add/remove users to/from) the watcher list of an issue.
Add Comments	Permission to add comments to issues. Note that this does not include the ability to edit or delete comments.
Edit All Comments	Permission to edit any comments, regardless of who added them.
Edit Own Comments	Permission to edit comments that were added by the user.
Delete All Comments	Permission to delete any comments, regardless of who added them.
Delete Own Comments	Permission to delete comments that were added by the user.
Create Attachments	Permission to attach files to an issue. (Only relevant if Attachments are enabled). Note that this does not include the ability to delete attachments.
Delete All Attachments	Permission to delete any attachments, regardless of who added them.
Delete Own Attachments	Permission to delete attachments that were added by the user.
Work On Issues	Permission to log work done against an issue, i.e. create a worklog entry. (Only relevant if Time Tracking is enabled).
Edit Own Worklogs	Permission to edit worklog entries that were added by the user. (Only relevant if Time Tracking is enabled).
Edit All Worklogs	Permission to edit any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled).
Delete Own Worklogs	Permission to delete worklog entries that were added by the user. (Only relevant if Time Tracking is enabled).
Delete All Worklogs	Permission to delete any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled).

Permission Schemes

What is a Permission Scheme?

A permission scheme is a set of user/group/role assignments for the project permissions listed above. Every project has a permission scheme. One permission scheme can be associated with multiple projects.

Why Permission Schemes?

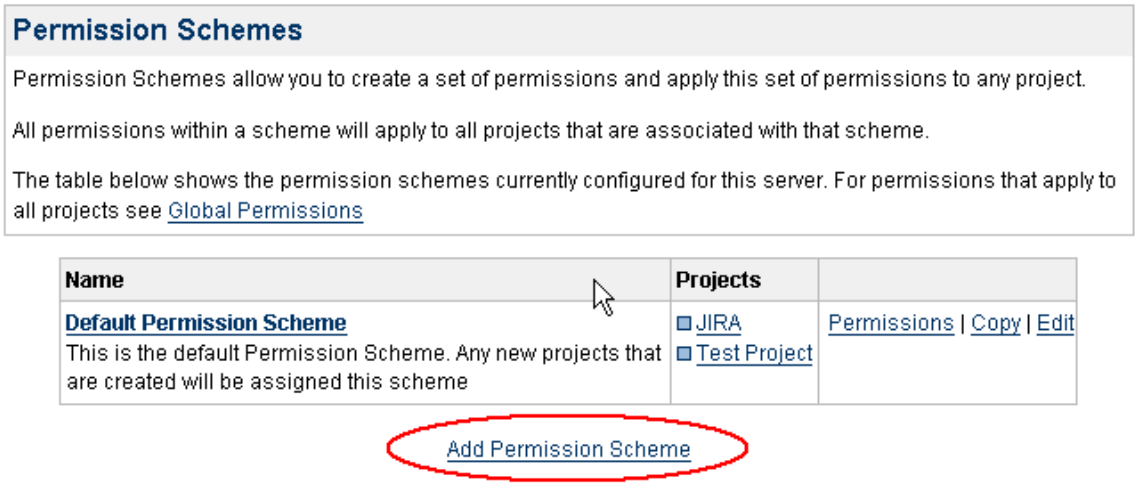
In many organisations, multiple projects have the same needs regarding access rights. (For example, only the specified project team may be authorised to assign and work on issues).

Permission schemes prevent having to set up permissions individually for every project. Once a permission scheme is set up it can be applied to all projects that have the same type of access requirements.

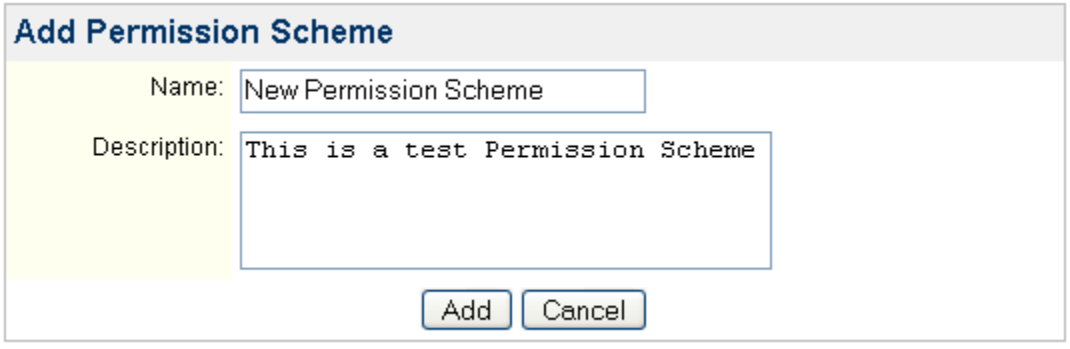
Creating a Permission Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title '**Schemes**', click the link labelled '**Permission Schemes**'.

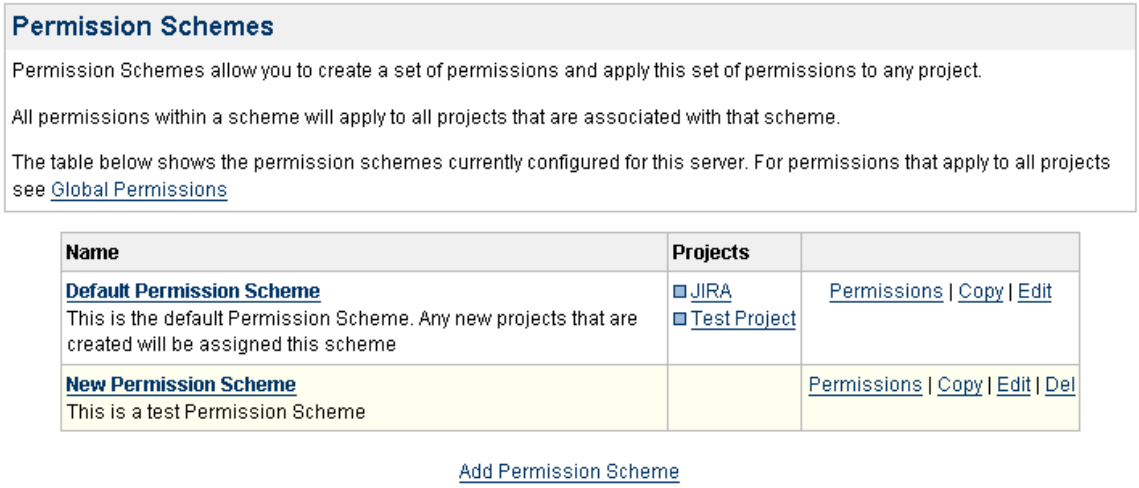
4. This will display the 'Permission Schemes' page. This page lists all of the Permission Schemes that JIRA currently has. Click the '**Add Permission Scheme**' link.



5. In the 'Add Permission Scheme' form, enter a name for the scheme, and a short description of the scheme. Click the '**Add**' button.



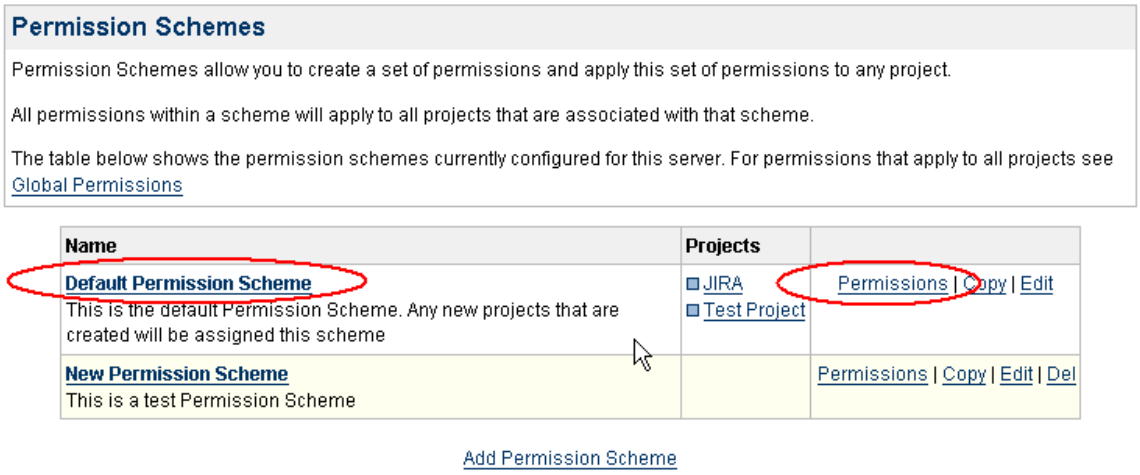
6. You will return to the 'Permission Schemes' page which now contains the newly added scheme.



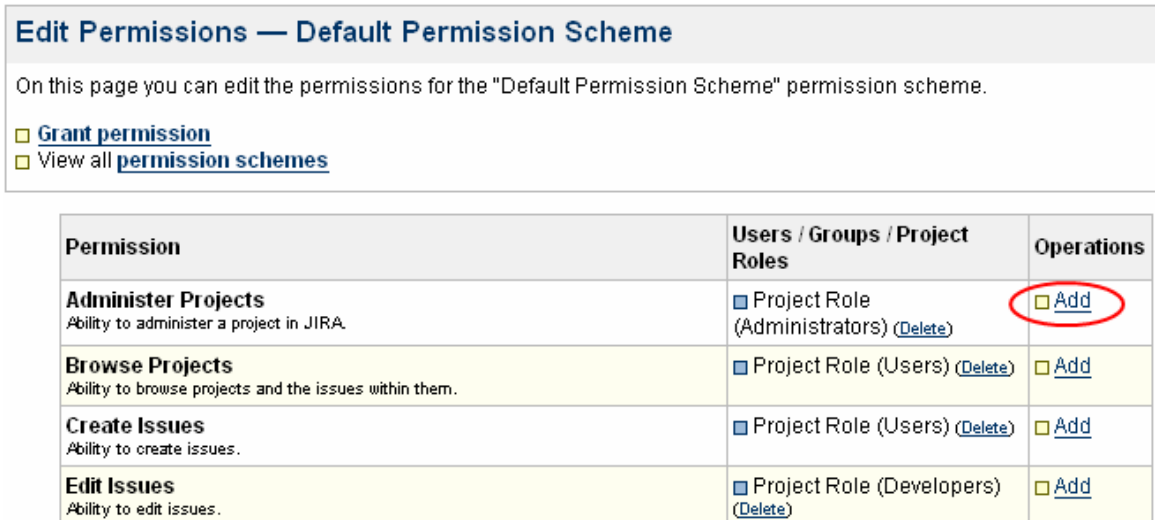
Adding Users, Groups or Roles to a Permission Scheme

1. On the panel on the left, under the title 'Features', click the link labelled '**Permission Schemes**'.

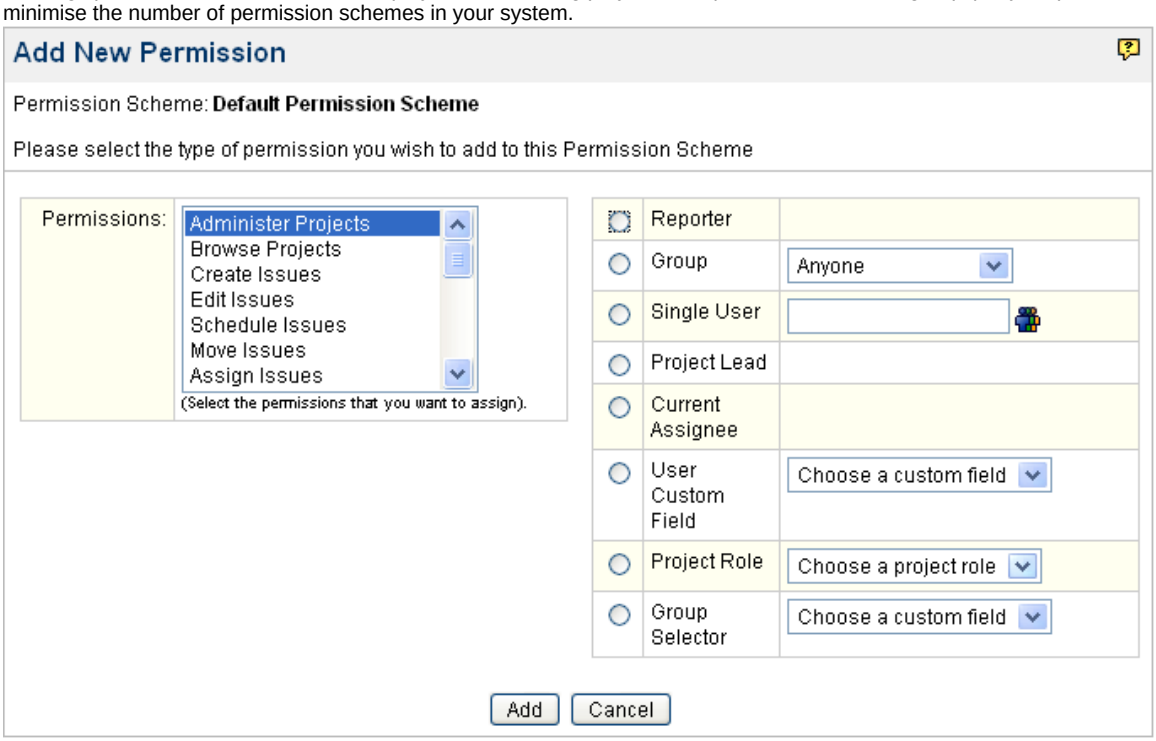
2. Click the '**Permissions**' link or on the name of the Permission Scheme to show a list of permissions



3. Click the '**Add**' link in the 'Operations' column.

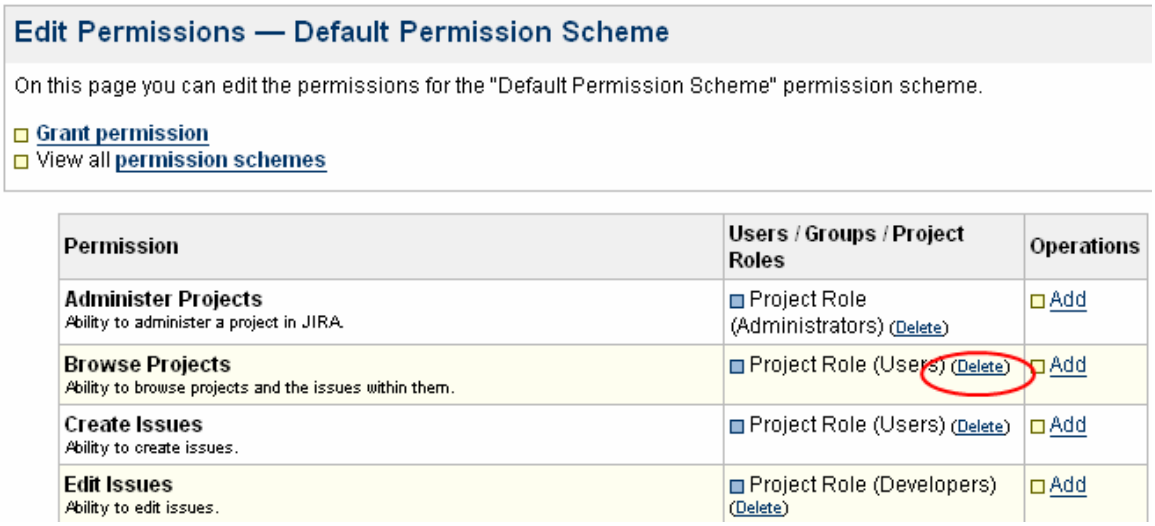


4. This will display the 'Add Permission' page. After selecting one or more permissions to add and who to add the selected permissions to, click the '**Add**' button. The users/groups/roles will now be added to the selected permissions. Note that project roles are useful for defining specific team members for each project. Referencing project roles (rather than users or groups) in your permissions can help you minimise the number of permission schemes in your system.



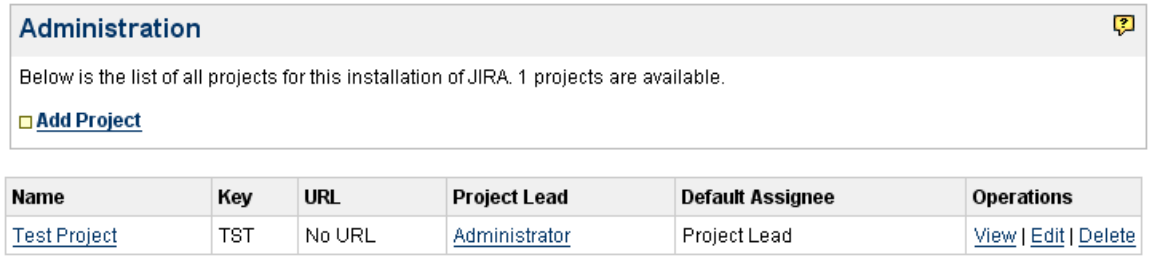
(Note: this screenshot is from JIRA Enterprise; in JIRA Professional/Standard only 'Group' and 'Project Role' are available)

5. Repeat the last 2 steps until all required users/groups/roles have been added to the permissions.
6. To delete a user/group/role from a permission click the '**Delete**' link in the "Users / Groups / Roles" column.

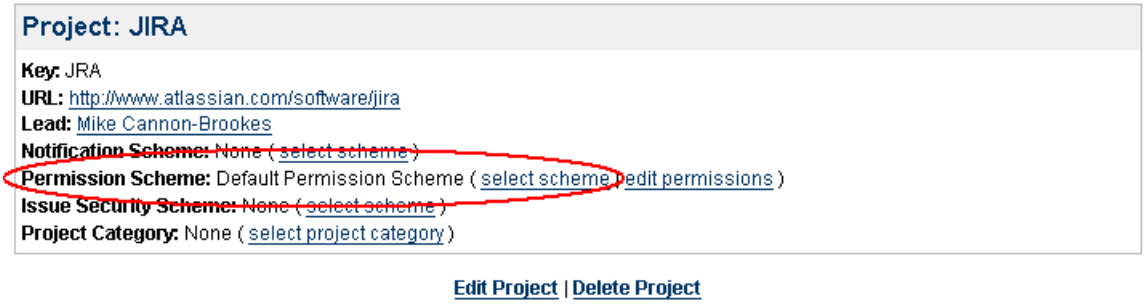


Associating a Permission Scheme with a Project

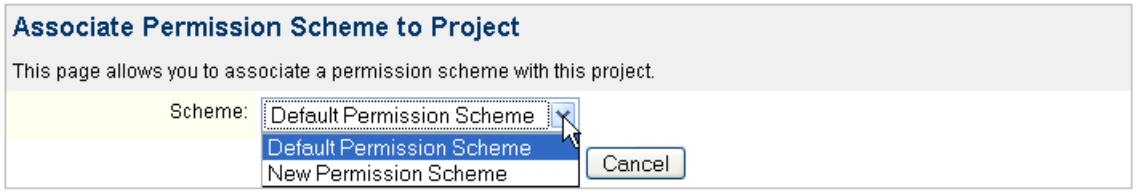
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. A list of projects is displayed



4. Select the project you want by clicking on the project name. This will display the project details
5. Click the '**select scheme**' link beside the Permission Scheme caption.



6. This will bring up a list of Permission Schemes. Select the Permission Scheme that you want to associate with this project.

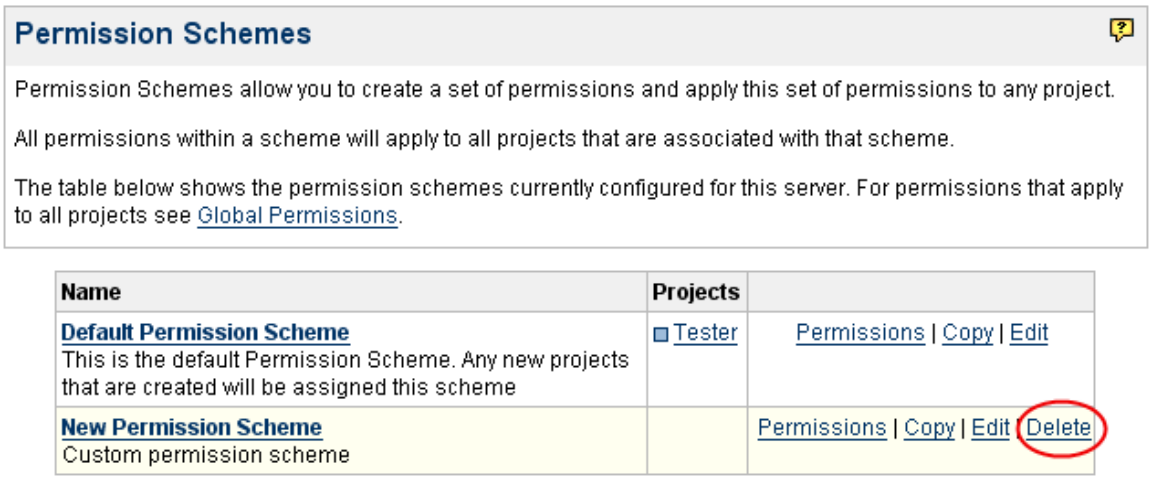


7. Click the '**Associate**' button to associate the project with the permission scheme.

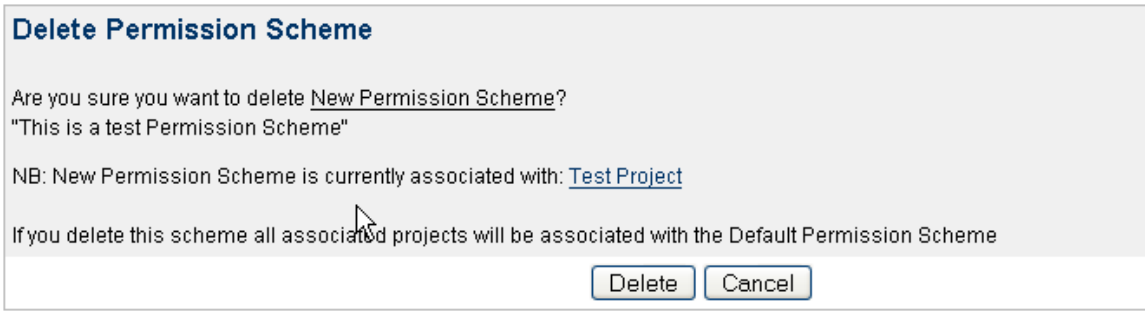
Deleting a Permission Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Features', click the link labelled '**Permission Schemes**'.

4. This will display the 'Permission Schemes' page. This page lists all the Permission Schemes that are currently defined in your JIRA system. Click the '**Delete**' link for the scheme that you want to delete.



5. A confirmation screen will appear. To delete click '**Delete**' otherwise click '**Cancel**'.

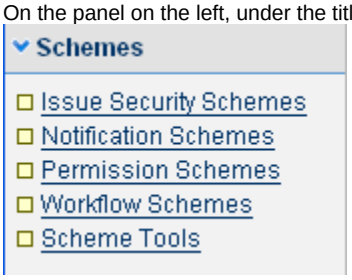


6. The scheme will be deleted and all associated projects will be automatically associated with the Default Permission Scheme. (Note that you cannot delete the Default Permission Scheme.)

See also Minimising the number of Permission Schemes and Notification Schemes.

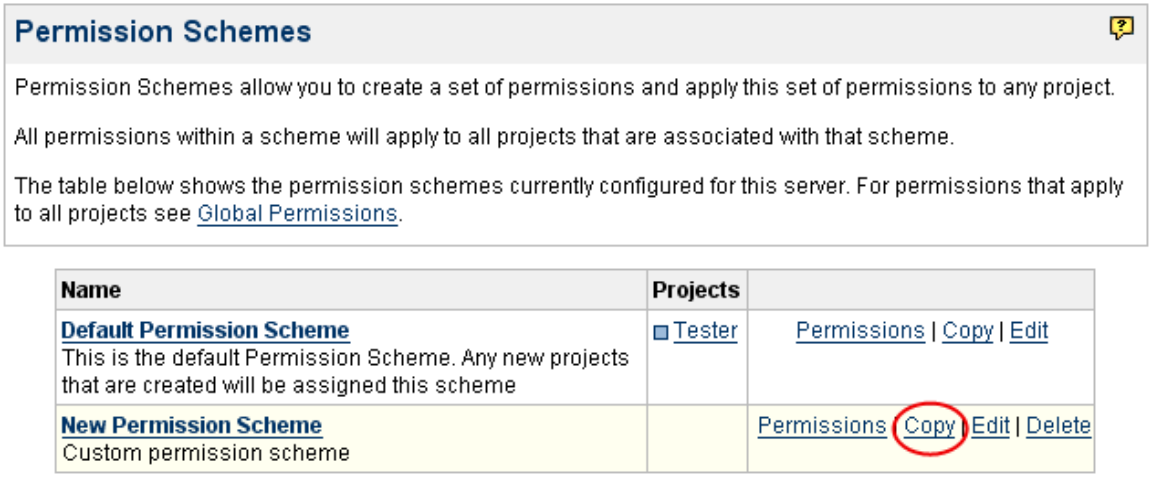
Copying a Permission Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Features', click the link labelled '**Permission Schemes**'.

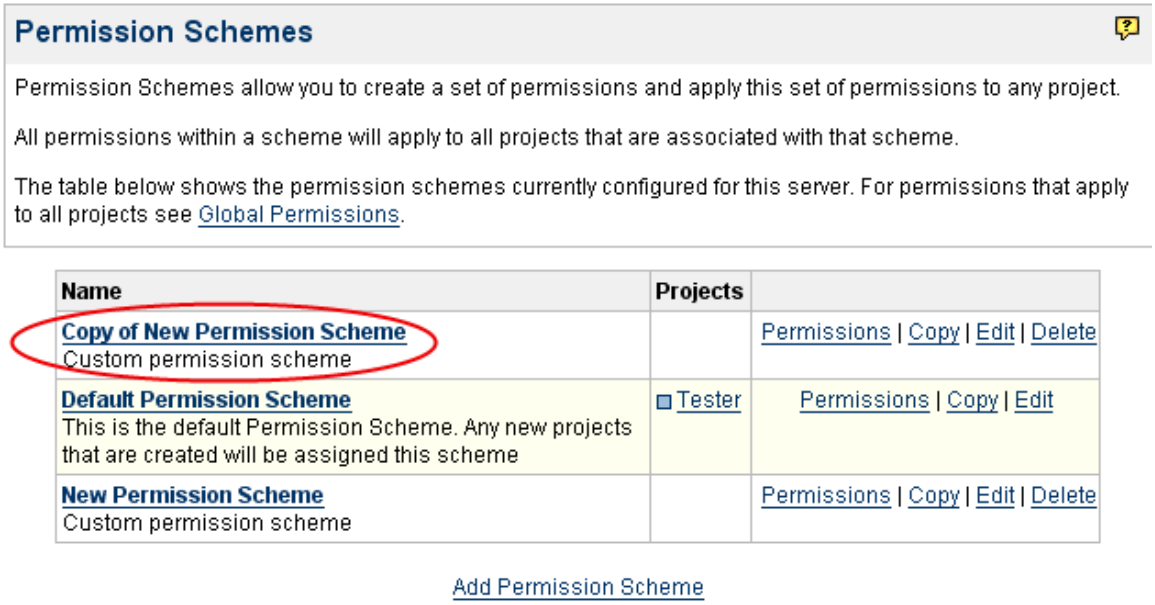


4. This will display the 'Permission Schemes' page. This page lists all of the Permission Schemes that JIRA currently has. Click the '**Copy**'

link for the scheme that you want to copy.



5. A new scheme will be created with the same permissions and the same users/groups/roles assigned to them.



Additional Resources

- Permission scheme overview tutorial video - Watch this short tutorial video to see how to set up a new permission scheme to control which users can perform which operations on an issue. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Configuring Issue Level Security

Issue security levels allow you to control who can see individual issues within a project (subject to the project's permissions).

An issue security level is a *named collection of users.* Issue security levels are created within issue security schemes, which are then associated with projects. Once an issue security scheme has been associated with a project, its security levels can be applied to issues in that project (note, sub-tasks will inherit the security level of their parent issue). Those issues will then only be accessible to members of that security level.

A security level's members may consist of:

- Individual users
- Groups
- Project roles
- Issue roles such as 'Reporter', 'Project Lead', and 'Current Assignee'
- 'Anyone' (eg. to allow anonymous access)
- A (multi-)user picker custom field.
- A (multi-)group picker custom field. This can either be an actual group picker custom field, or a (multi-)select-list whose values are group

names.

Only users with the project-specific 'Set Issue Security' permission can apply a security level to an issue, regardless of whether they are members of the security level.

On this page:

- Why use issue security?
- Creating an Issue Security Scheme
- Adding a Security Level to an Issue Security Scheme
- Setting the Default Security Level for an Issue Security Scheme
- Adding Users/Groups/Project Roles to a Security Level
- Assigning an Issue Security Scheme to a Project
- Deleting an Issue Security Scheme
- Copying an Issue Security Scheme
- Additional Resources

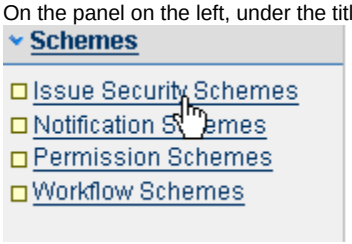
Why use issue security?

As an example, a company may have a public instance of JIRA running. Within this instance they may have several projects that external people (customers) can browse. However, it may not be appropriate to show all issues to the customers. To achieve this you could:

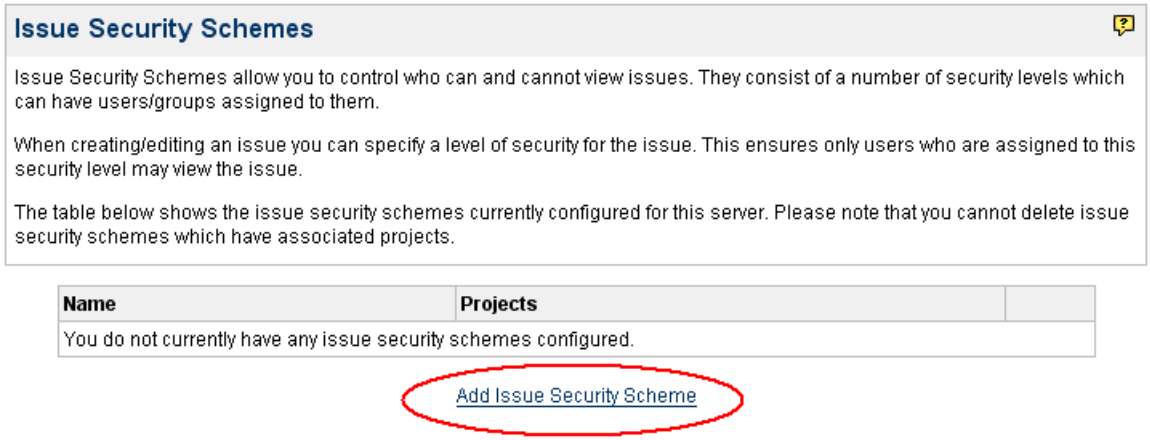
- Create an issue security scheme.
- Create an issue security level named 'Private' for this scheme.
- Add appropriate people to the 'Private' security level.
- Associate the issue security scheme with the relevant projects.
- Set the security level of specific issues to 'Private'.

Creating an Issue Security Scheme

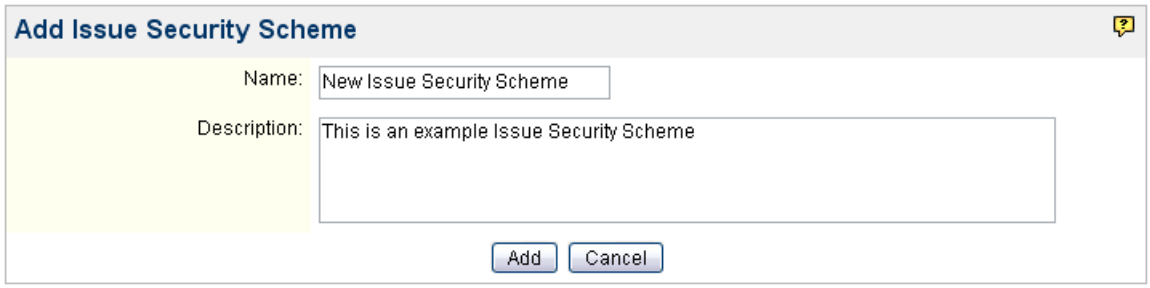
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click the link labelled '**Issue Security Schemes**'.



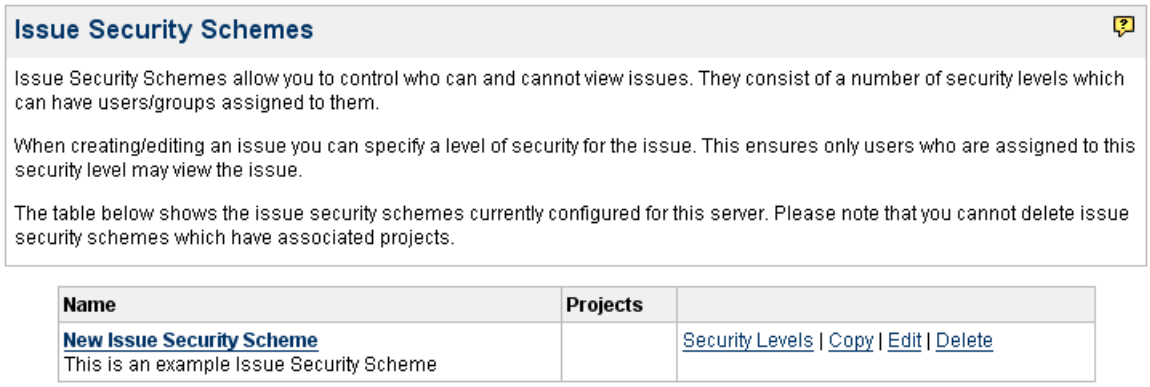
4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the '**Add Issue Security Scheme**' link.



5. In the 'Add Issue Security Scheme' form, enter a name for the issue security scheme, and a short description of the scheme. Then click the '**Add**' button.

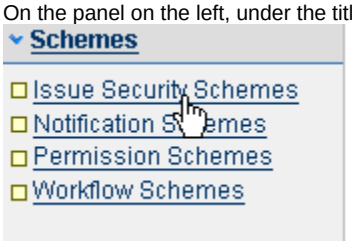


6. You will return to the 'Issue Security Schemes' page, which now contains the newly added scheme.

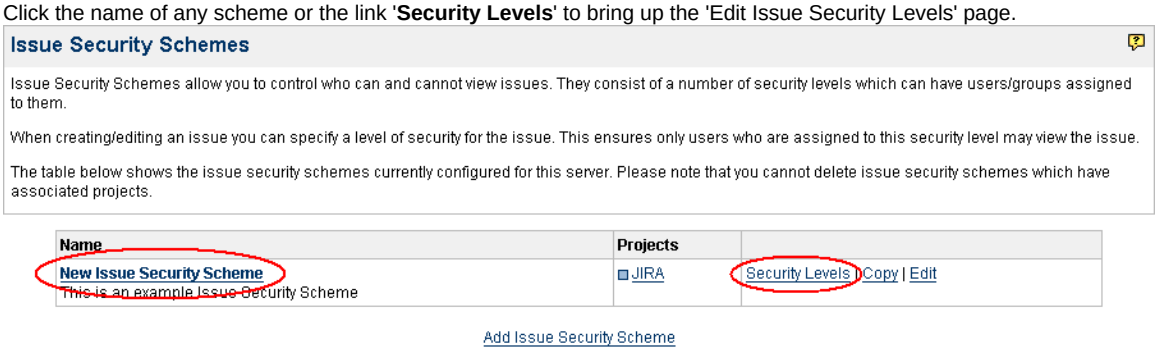


Adding a Security Level to an Issue Security Scheme

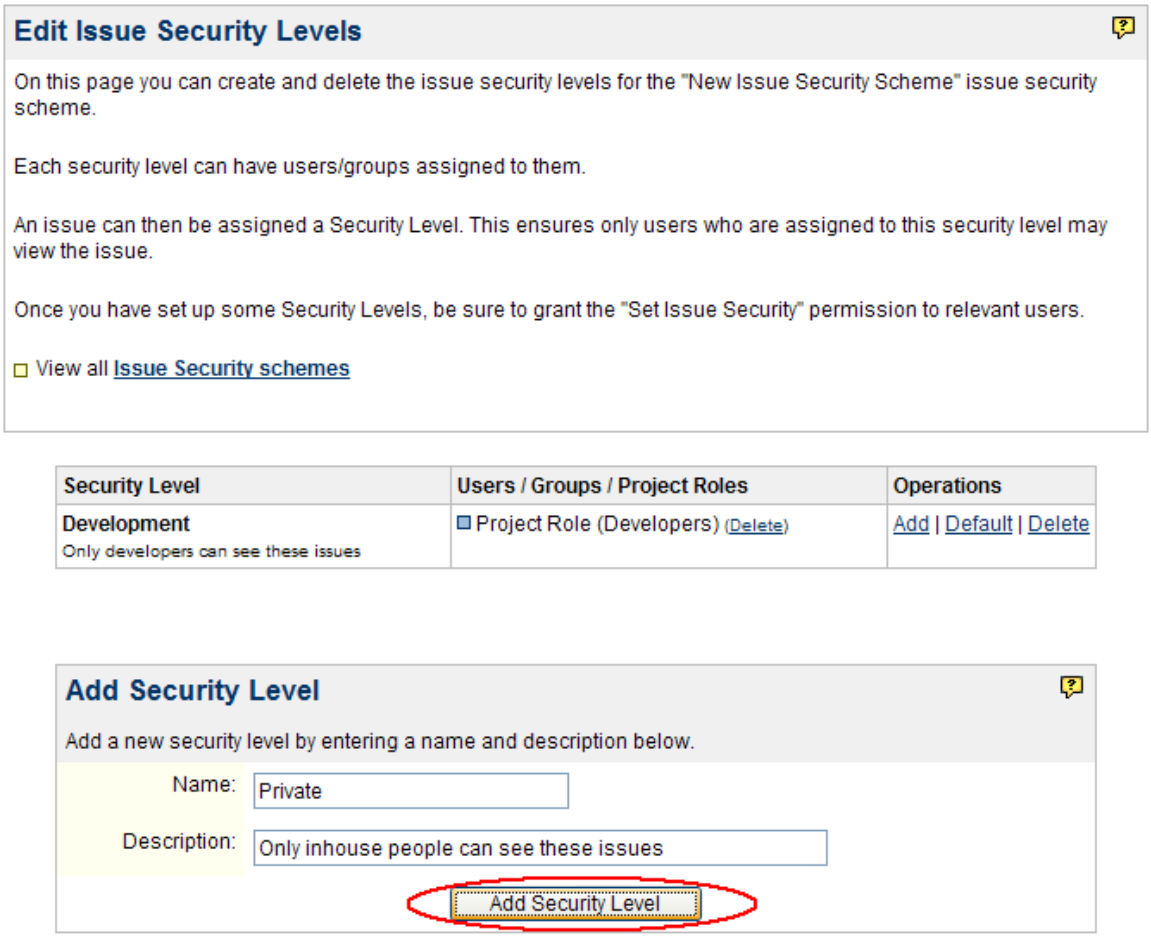
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click the link labelled '**Issue Security Schemes**'.



4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has.
5. Click the name of any scheme or the link '**Security Levels**' to bring up the 'Edit Issue Security Levels' page.



6. In the '**Add Security Level**' box, enter a name and description for your new security level. Then click the button '**Add Security Level**'.

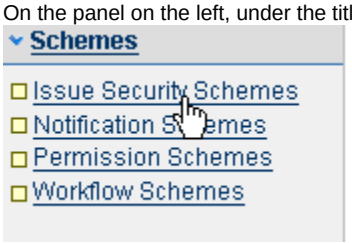


Setting the Default Security Level for an Issue Security Scheme

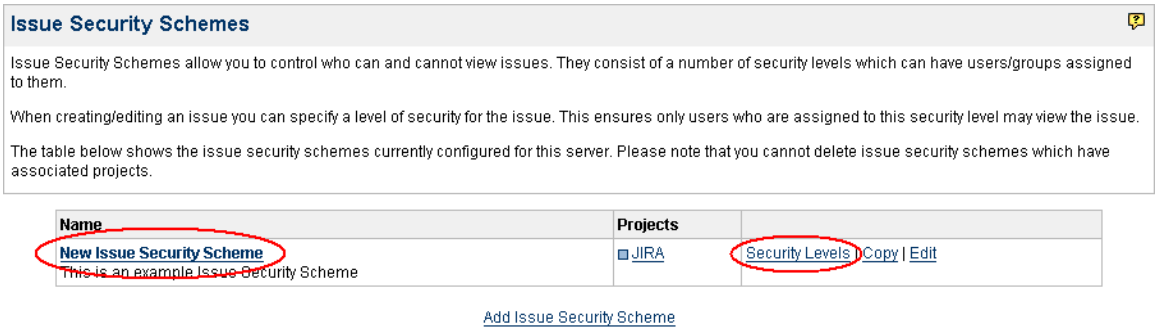
You can choose to specify a Default Security Level for your issue security scheme.

The Default Security Level is used when issues are created. If the reporter of an issue does not have the permission 'Set Issue Security', then the issue's security level will be set to the Default Security Level. If the project's issue security scheme does not have a Default Security Level, then the issue's security level will be set to 'None'. (A security level of 'None' means that anybody can see the issue.)

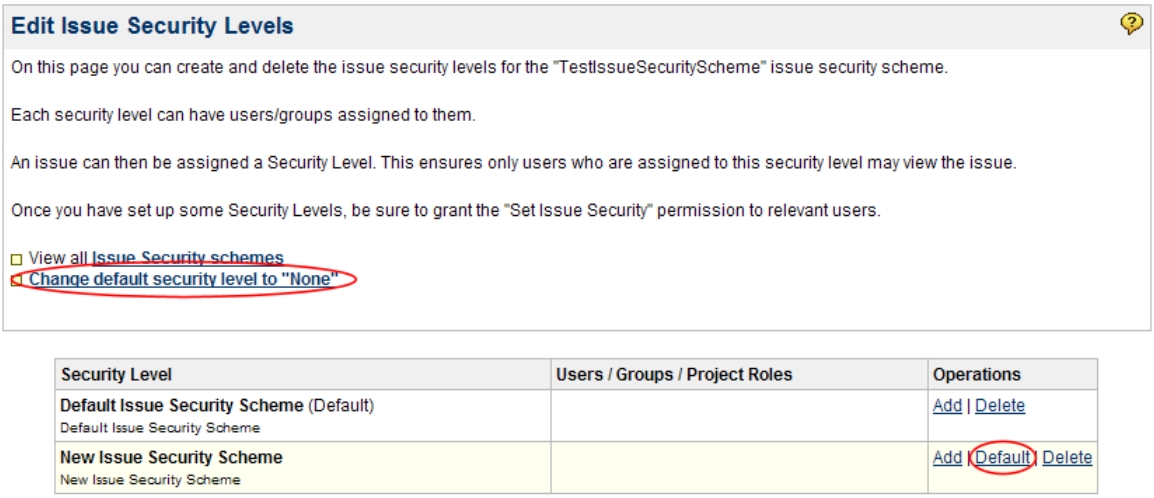
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click the link labelled '**Issue Security Schemes**'.



4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has.
5. Click the name of any scheme or the link '**Security Levels**' to bring up the 'Edit Issue Security Levels' page.

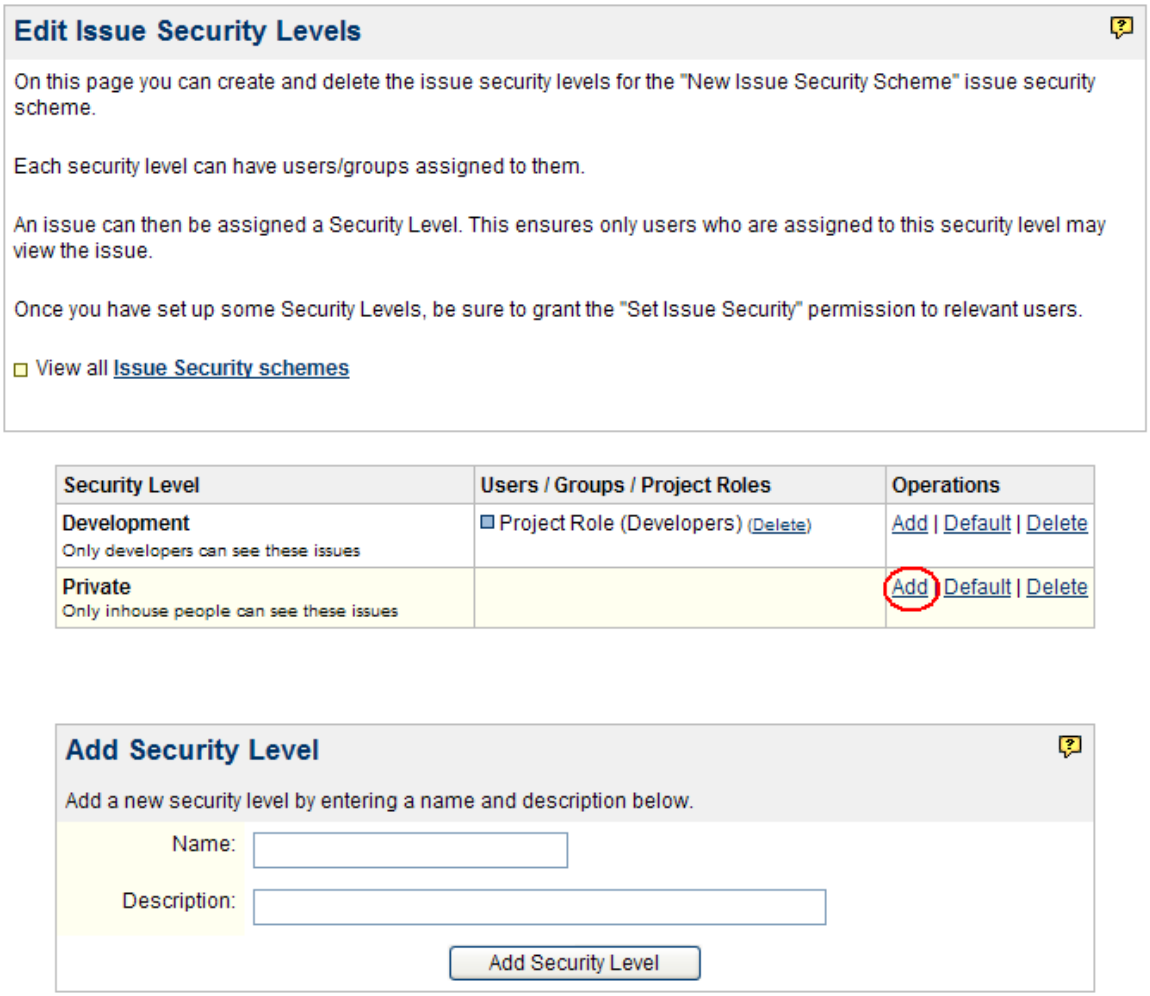


6. The default issue security level for a scheme can be set and removed, as follows:
    - To set the Default Security Level for an issue security scheme, locate the appropriate Security Level and click its '**Default**' link.
    - To remove the Default Security Level for an issue security scheme, click the '**Change default security level to "None"**' link.

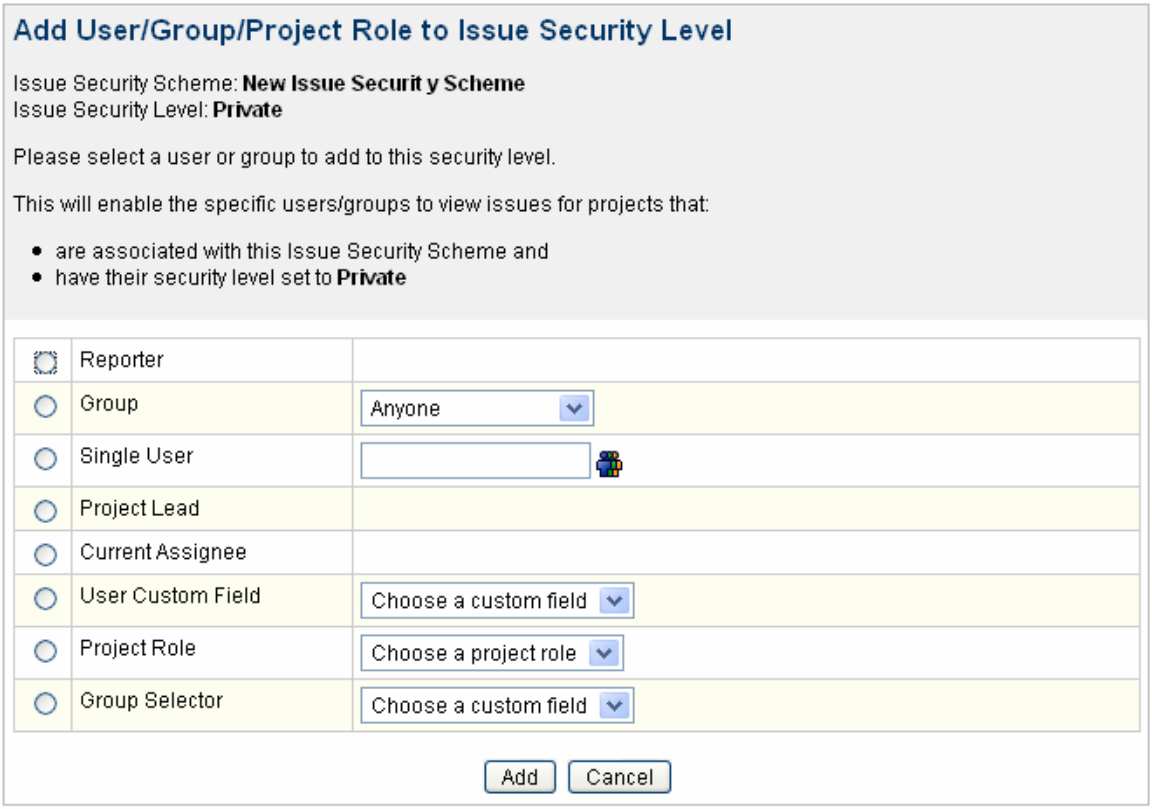


Adding Users/Groups/Project Roles to a Security Level

1. Go to the 'Edit Issue Security Levels' page (see above).
2. Locate the appropriate Security Level and click its '**Add**' link:



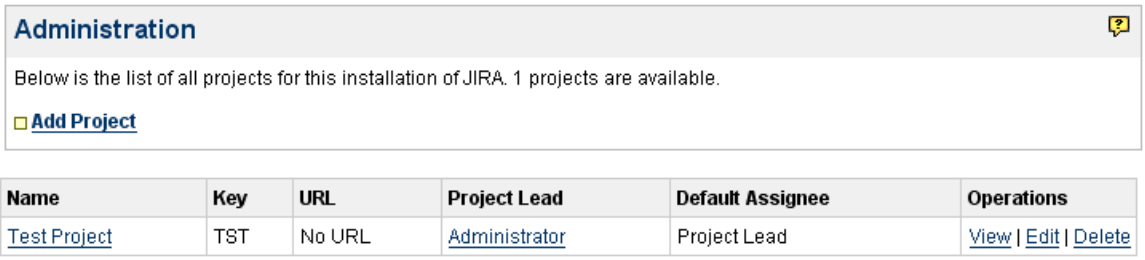
3. This will display the 'Add Issue Security' page. Select the appropriate user, group or project role, then click the '**Add**' button.



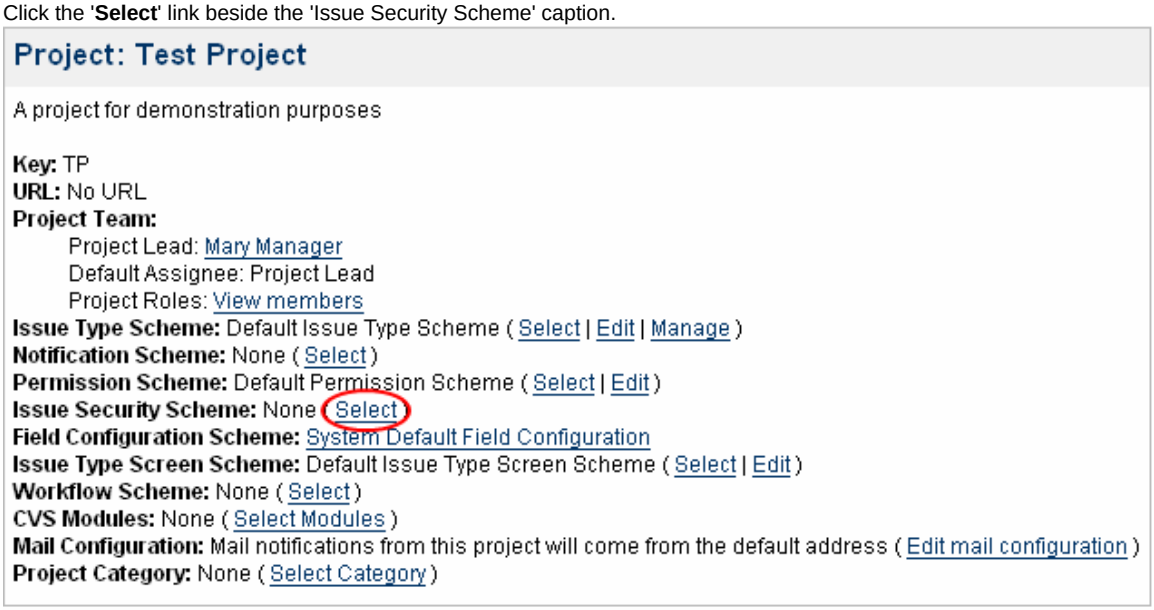
4. Repeat steps 2 and 3 until all appropriate users and/or groups and/or project roles have been added to the security level.

Assigning an Issue Security Scheme to a Project

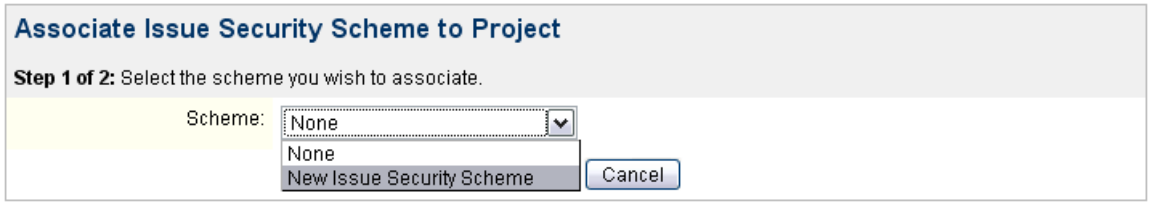
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. A list of projects is displayed



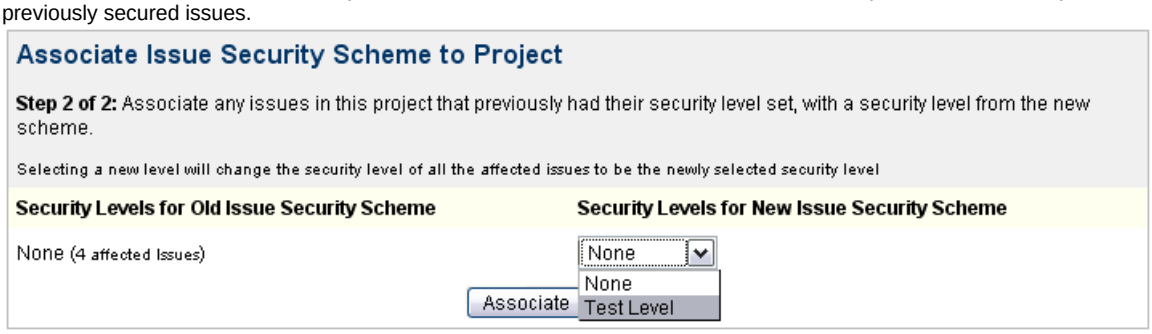
4. Select the project you want by clicking on the project name. This will display the project details
5. Click the '**Select**' link beside the 'Issue Security Scheme' caption.



6. This will bring up a list of Issue Security Schemes. Select the Issue Security Scheme that you want to associate with this project.



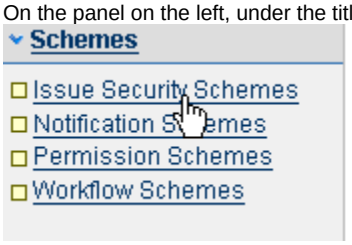
7. If there are no previously secured issues (or if the project didn't previously have an issue security scheme), skip the next step.
8. If there are any previously secured issues, select a new security level to replace each old level. All issues with the security level from the old scheme will now have the security level from the new scheme. You can choose 'None' if you want the security to be removed from all previously secured issues.



9. Click the '**Associate**' button to associate the project with the issue security scheme.

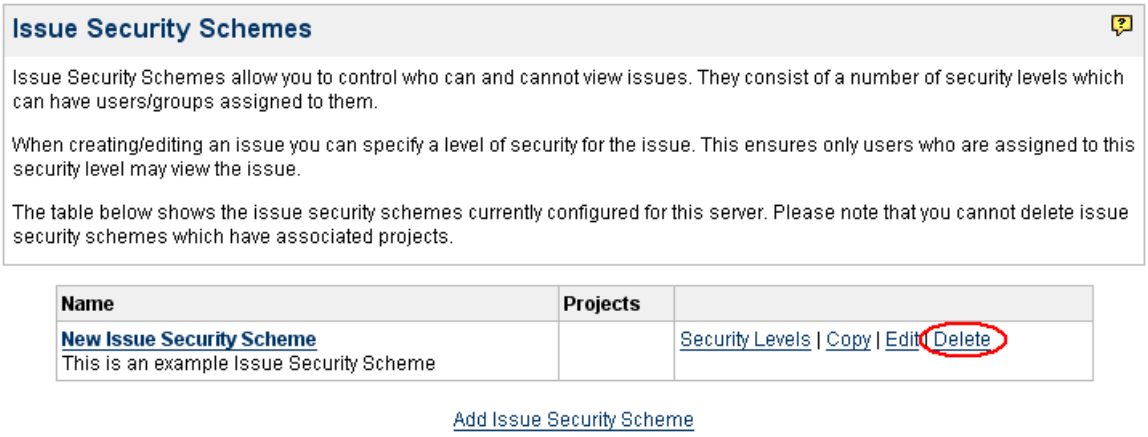
Deleting an Issue Security Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click on the link labelled '**Issue Security Schemes**'.

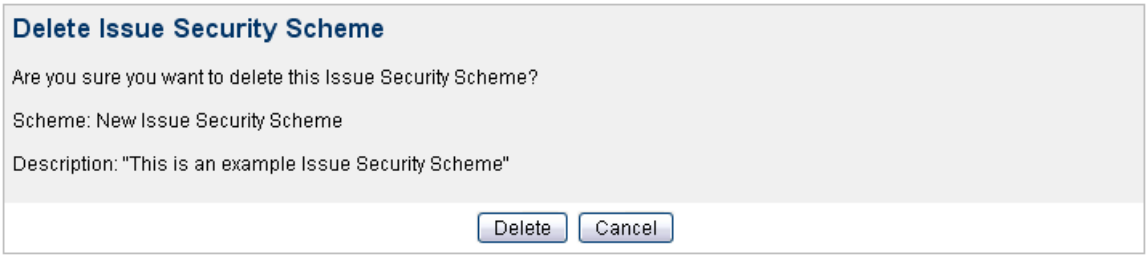


4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the '**Delete**' link for the scheme that you want to delete.

> You cannot delete a Issue Security Scheme if it is associated with a project; you must first unassign the scheme. To unassign a scheme, please refer to Assigning an Issue Security Scheme.

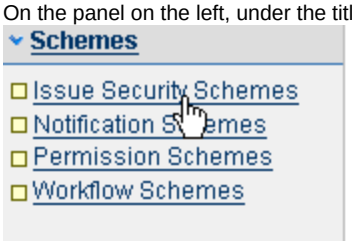


5. A confirmation screen will appear. To delete, click '**Delete**'; otherwise click '**Cancel**'.

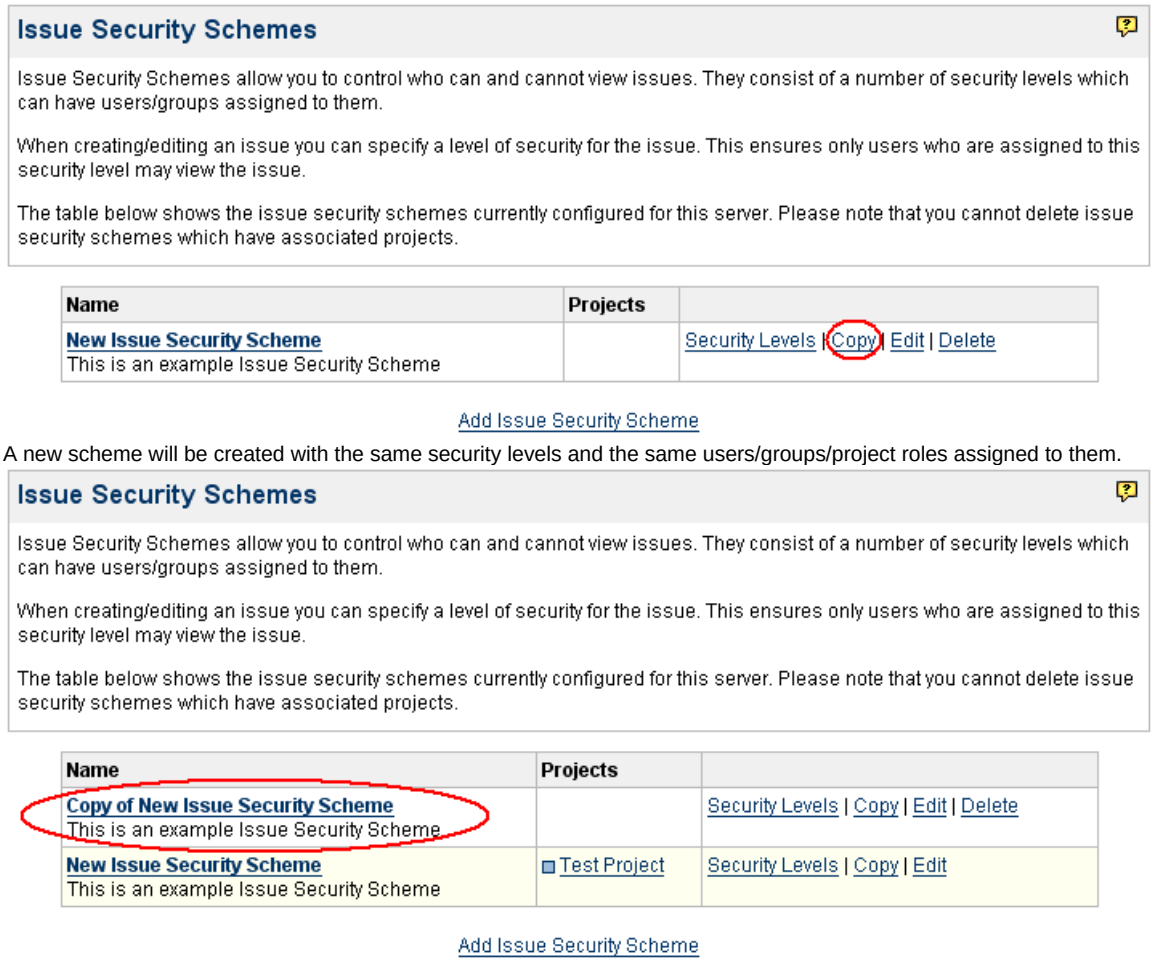


Copying an Issue Security Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes', click the link labelled '**Issue Security Schemes**'.



4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the '**Copy**' link for the scheme that you want to copy.



5. A new scheme will be created with the same security levels and the same users/groups/project roles assigned to them.

Additional Resources

- Issue security scheme overview tutorial video — Watch this short tutorial video to see how to use an issue security scheme to restrict the viewing of issues to specified users. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Configuring Fields and Screens

On this page:

- Overview
- Concepts
- Diagram: How Fields, Screens and Workflow interrelate

In this section:

- Configuring Built-in Fields
  - Defining 'Issue Type' Field Values
    - Associating Issue Types with Projects
  - Defining 'Priority' Field Values
  - Defining 'Resolution' Field Values
  - Defining 'Status' Field Values
  - Translating Resolutions, Priorities, Statuses and Issue Types
- Adding a Custom Field
  - Creating Help for a Custom Field
  - Configuring a Custom Field
- Specifying Field Behaviour
  - Associating Field Behaviour with Issue Types
  - Configuring Rich-Text Renderers
- Defining a Screen
  - Associating a Screen with an Issue Operation
  - Associating a Screen with an Issue Type

Overview

To help you tailor JIRA to your organisation's needs, JIRA enables you to manipulate the display and behaviour of issue fields ('Summary', 'Description', 'Issue Type', etc). You can:

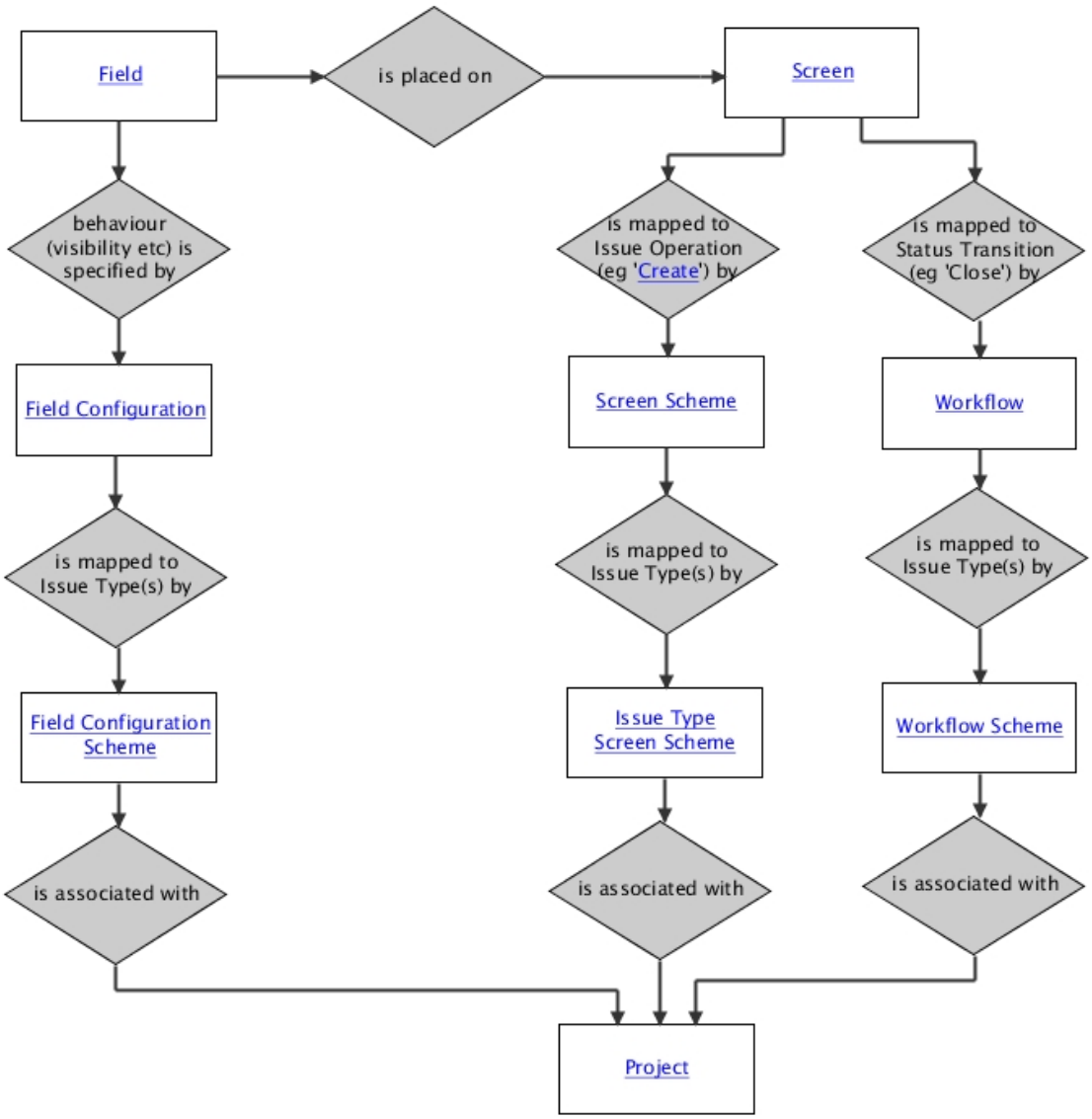
- Change a field's description
- Make a field hidden or visible
- Make a field required or optional
- Add your own values for 'Issue Type', 'Priority', 'Resolution' and 'Status'
- Create new 'custom' fields
- Enable a rich text renderer for (some) fields
- Position fields on a screen
- Choose which screen should be displayed for each issue operation (e.g. 'Create Issue', 'Edit Issue') or workflow transition (e.g. 'Resolve Issue', 'Close Issue')

Concepts

Some key JIRA concepts include:

- Field Configuration --- a set of definitions for all fields, comprising: each field's description; whether each field is hidden or visible; whether each field is required or optional; and what type of renderer to use for each text field.
- Screen — defines which fields are present on a screen, and their order. (Note that a hidden field can be present on a screen, but will still be invisible.)
- Screen Scheme — associates different screens with different issue operations (e.g. 'Create Issue', 'Edit Issue', 'View Issue').
- Issue Type Screen Scheme — associates Screen Schemes with issue types and projects. This allows you to specify different Screens for the same operation (e.g. 'Create Issue') in the same project for issues of different types. For example, you could use one screen when creating an issue of type 'Bug', and a different screen when creating an issue of type 'Task'.
- Field Configuration Scheme — associates Field Configurations with issue types and projects. This allows you to specify different field behaviour for the same field in the same project for issues of different types.
- Issue Type Scheme — associates issue types with projects.

Diagram: How Fields, Screens and Workflow interrelate



Configuring Built-in Fields

Each issue has a number of built-in fields, as shown in the sample issue in the *JIRA User's Guide*.

Some of the built-in fields can be customised as follows:

- Defining 'Issue Type' Field Values
  - Associating Issue Types with Projects
- Defining 'Priority' Field Values
- Defining 'Resolution' Field Values
- Defining 'Status' Field Values
- Translating Resolutions, Priorities, Statuses and Issue Types

Defining 'Issue Type' Field Values

JIRA ships with a set of default 'Issue Types' to help you get started. Everyone's needs are different and so JIRA also allows you to add, edit and delete your own custom Issue Types.

On this page:

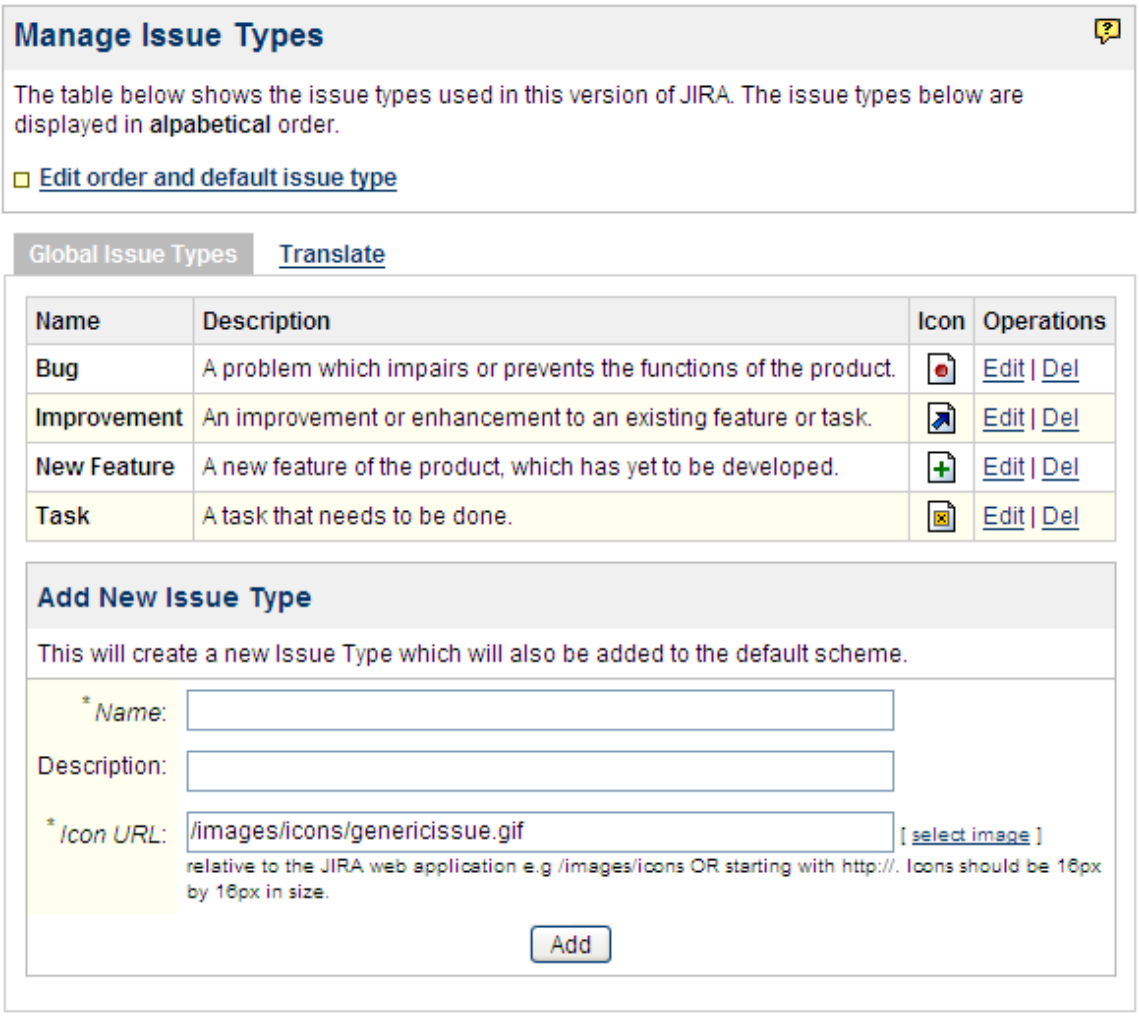
- Creating an Issue Type
- Deleting an Issue Type
- Editing an Issue Type

Note that you can also:

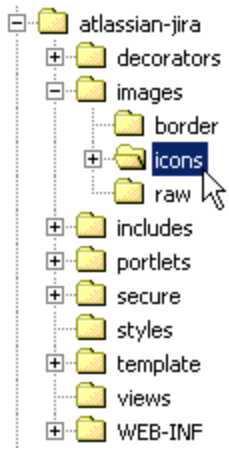
- control the set of available Issue Types for each project — see Associating Issue Types with Projects.
- control the display order of available Issue Types and the default Issue Type for each project — see Associating Issue Types with Projects. *Reordering* issue types changes the order in which they are displayed to the user who is creating an issue; and the *default* issue type is the one that is displayed in the selection-box (see Creating an Issue).
- associate particular Issue Types with particular Fields, Screens and Workflow — for details see 'Associating Field Behaviour with Issue Types', 'Associating Screens with Issue Types' and 'Activating Workflow', respectively. (Also see the diagram of how these interrelate.)

Creating an Issue Type

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:
3. On the panel on the left, under the title 'Issue Settings', click on the link labelled '**Issue Types**'.
4. This will bring up the 'Manage Issue Types' page. The page lists all issue types, along with a form underneath to add new issue types.



5. To add a new Issue Type, fill in the Add New Issue Type form. For the '**Name**', enter a short phrase that best describes your new Issue Type. For the '**Description**', enter a sentence or two to describe when this Issue Type should be used. For the '**Icon URL**' you need to supply the path of a 16-by-16-pixel image that has been placed somewhere inside JIRA's opened .war. We suggest you place it in /images/icons:



Once you create your new Issue Type, it will be automatically added to the **Default Issue Type Scheme** (in *Professional* and *Enterprise Editions* of JIRA). For more information, see *Managing Issue Type Schemes*.

Deleting an Issue Type

Before you begin:
So that JIRA can prompt you to choose a new Issue Type for any issues that are currently using the Issue Type which you are deleting, please ensure that the Issue Type which you are deleting is associated with:

- the same Workflow in all Workflow Schemes that are associated with one or more projects.
- the same Field Configuration in all Field Configuration Schemes that are associated with one or more projects.
- the same Screen Scheme in all Issue Type Screen Schemes that are associated with one or more projects.

Alternatively, you can simply search for all issues that currently use the Issue Type which you are deleting, and do a Bulk Edit to change them to a different Issue Type.

To delete an Issue Type,

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:
3. On the panel on the left, under the title 'Issue Settings', click on the link labelled '**Issue Types**'.
4. This will bring up the 'Manage Issue Types' page (as shown above). Click the '**Del**' link at the right of the issue type that you wish to delete.

Editing an Issue Type

To change the Name, Description or Icon for an Issue Type:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:
3. On the panel on the left, under the title 'Issue Settings', click on the link labelled '**Issue Types**'.
4. This will bring up the 'Manage Issue Types' page (as shown above). Click the '**Edit**' link at the right of the issue type that you wish to edit.

Note: To reorder an Issue Type, or set it as a default, see Associating Issue Types with Projects. (*Reordering* issue types changes the order in which they are displayed to the user who is creating an issue; and the *default* issue type is the one that is displayed in the selection-box --- see Creating an Issue.)

Associating Issue Types with Projects

On this page:

- What is an 'Issue Type Scheme'?
- Managing Issue Type Schemes
  - Creating a new scheme
  - Editing a scheme
  - Associating a scheme with projects
- Managing Schemes for a Project
- Issue Type Migration Wizard

What is an 'Issue Type Scheme'?

An **Issue Type Scheme** is a sub-set of issue types. An Issue Type Scheme allows you to:

- restrict the set of available issue types for each project.
- control the display order of available Issue Types and the default Issue Type for each project (ie. the Issue Type that is displayed in the selection-box when a user creates an issue).

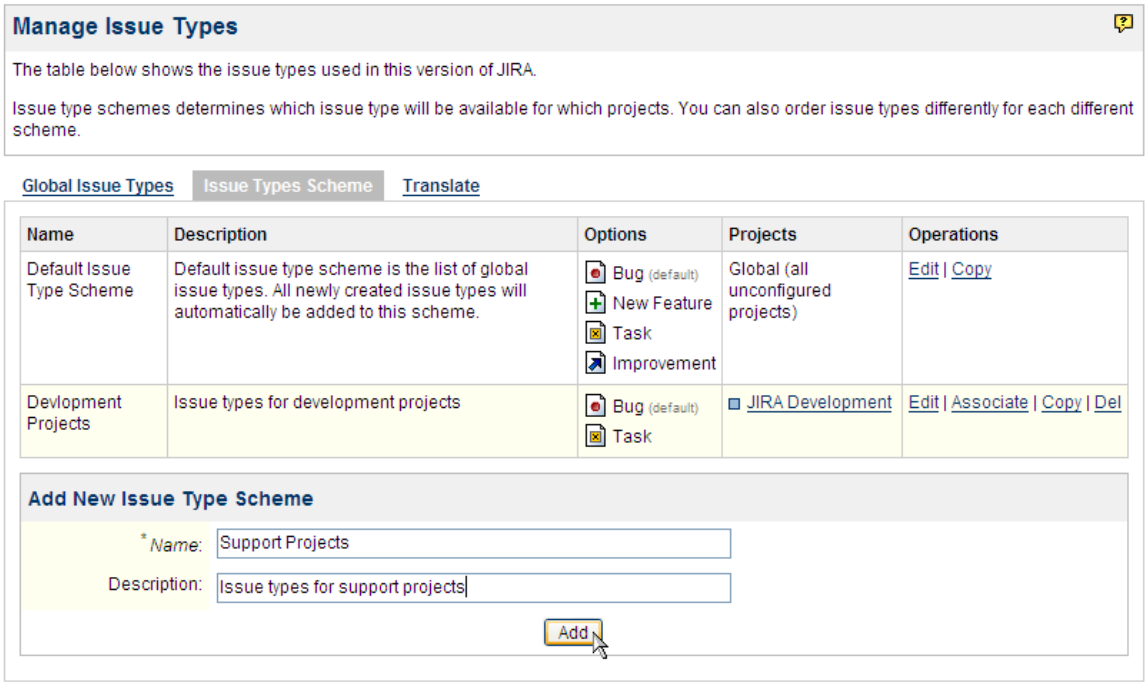
An Issue Type Scheme can be shared across multiple projects, so that a group of similar projects can share the same issue type settings.

For example, in your company all projects may be one of two types, a development project or a support project. You could then create one scheme called *Task* with issue types *Bug* and *Development* and another called *Support Issue Types,* with *Development Query* and *Support Request.* You can then associate each scheme with the appropriate project(s), giving your users a different set of issue types depending on which project they decide to create issues in. Your future maintenance workload is minimised, because any change you make to a scheme is made across all projects that are associated with the scheme. In this example, adding a new issue type to all support projects only requires the simple step of adding the issue type to the *Support Issue Types* scheme.

Managing Issue Type Schemes

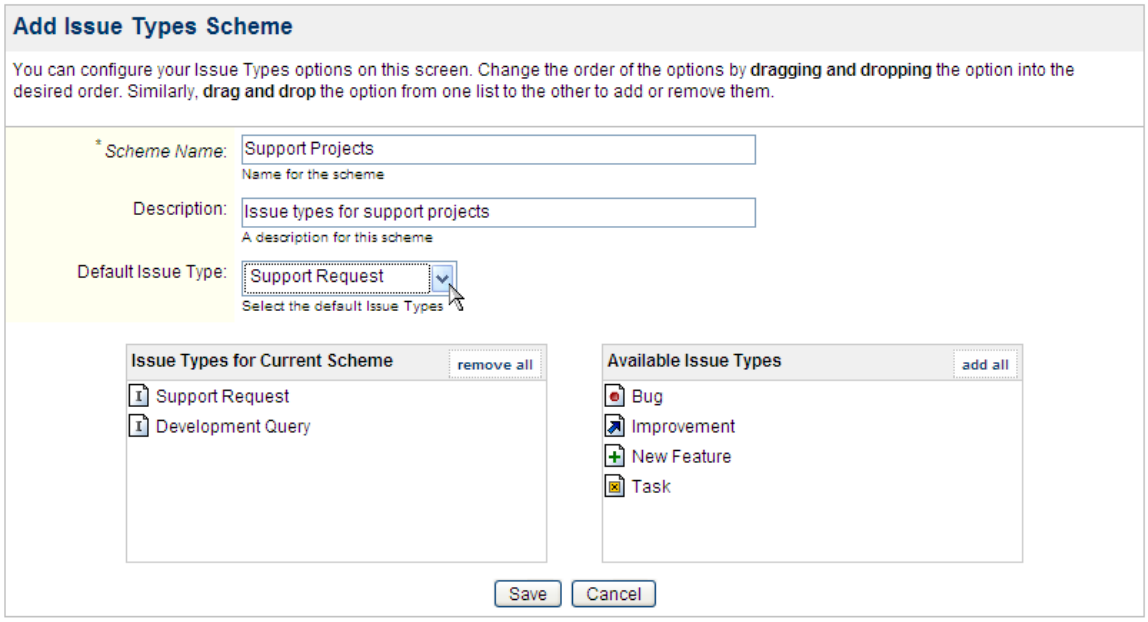
1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:
3. On the panel on the left, under the title 'Issue Settings', click on the link labelled '**Issue Types**'.
4. This will bring up the 'Manage Issue Types' page. Click the **'Issue Type Schemes'** tab. This will display all existing Issue Type Schemes, their related issue types and the associated projects.

The **'Default Issue Type Scheme'** contains all the issue types that exist in your JIRA system. This scheme is associated with all newly created projects by default. If some of your issue types are not relevant to all of your projects, create one or more new Issue Type Schemes (e.g. 'Development Projects' in this screenshot) as described below, and associate them with the appropriate projects instead of using the **Default Issue Type Scheme.**

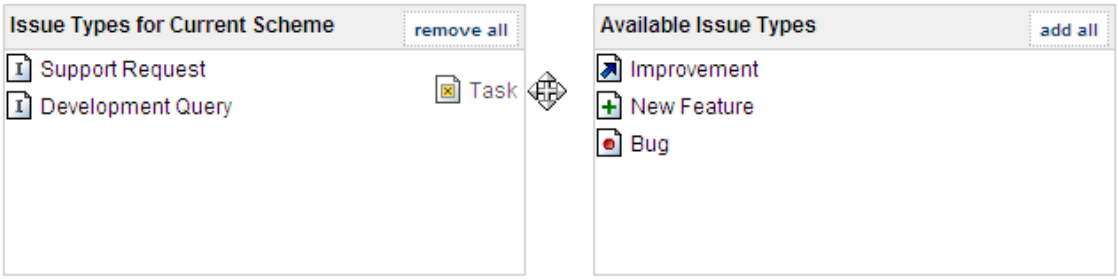


Creating a new scheme

1. To create a new scheme, enter the name and description for the new scheme. Ensure that the name is meaningful as this will be visible to other administrators and will allow them to better reuse the scheme.
2. Click the '**Add**' button and the screen below will be displayed.



3. Set the default for the new scheme from the select list. Add a new issue type to your scheme by dragging and dropping the issue type from the right hand list to the left. You can similarly reorder the issue types in the desired order by dragging and dropping them into the right positions.



4. If you need an issue type that does not currently exist, you can add this easily in using the add issue type form at the bottom of the page. This will add the issue type to the system and also add it to the scheme you're editing.
5. Once you've finished with your scheme. Click the '**Save' button to persist your changes. Note that unless you click on '*Save**', no scheme will be created.

Editing a scheme

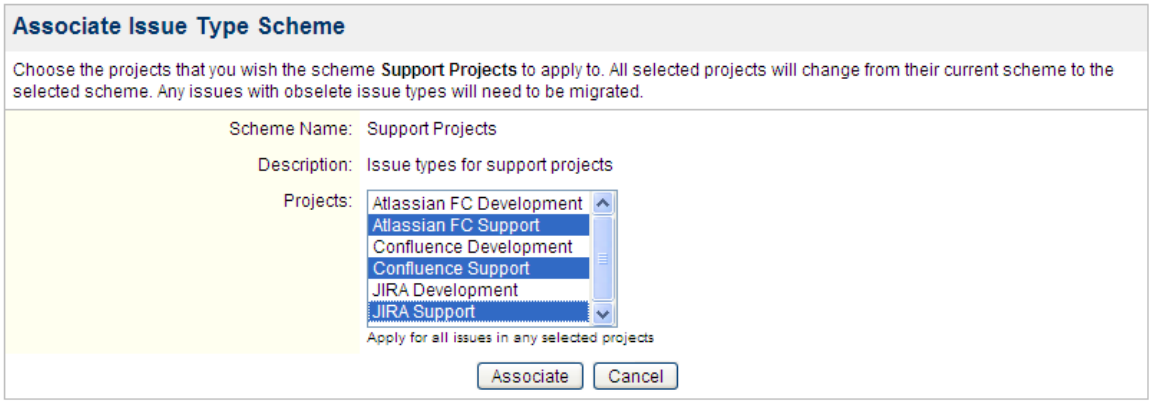
The process of editing a scheme is identical to the creation process. You can set defaults[1], reorder[2], add and remove issue types as before. However, if you're removing issue types from the scheme and there are issues associated with that issue type, you will be required to use the Issue Type Migration Wizard which will move your issues from the obsolete issue type to a valid one. Note that if you cancel out of this process at any time, your changes will not be committed. See below for more information about the migration wizard.

[1] [2] *Reordering* issue types changes the order in which they are displayed to the user who is creating an issue; and the *default* issue type is the one that is displayed in the selection-box (see Creating an Issue).

Associating a scheme with projects

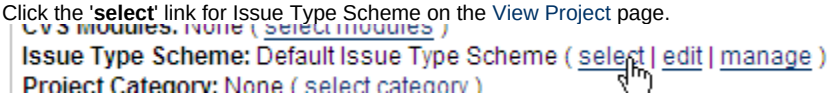
You can restrict the issue types available by associating your Issue Type Scheme with various projects. Click the '**Associate**' link and simply choose the projects that you wish your scheme to apply to. All selected projects will change from their current scheme to the selected scheme.

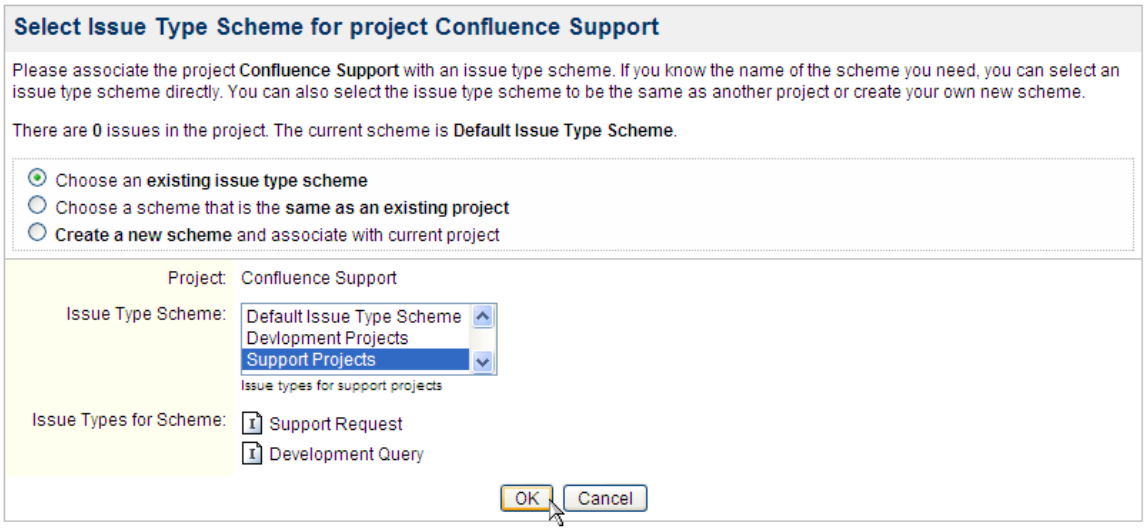
If the new scheme does not have an issue type that was present in the old scheme, you will be asked to use the Issue Type Migration Wizard to migrate the issues.

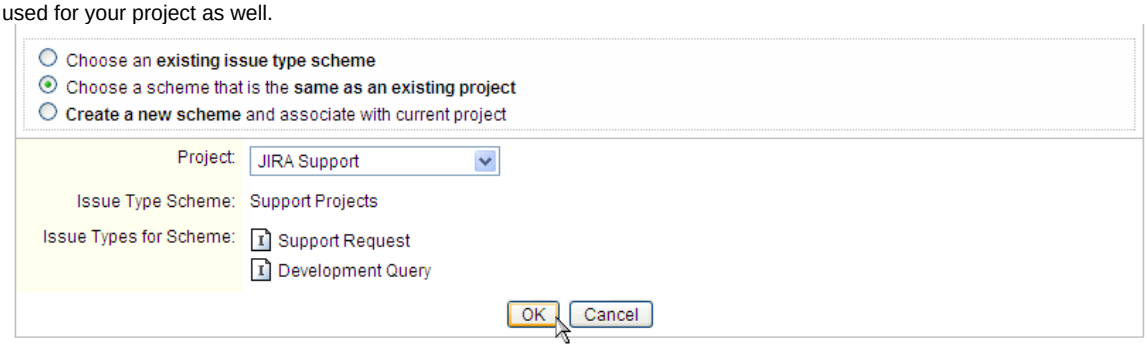


Managing Schemes for a Project

When updating a project you may often want to quickly restrict its issue types. However, the available Issue Type Schemes may not always be applicable, or you might not know which scheme to choose. The *Select Issue Type Scheme* screen makes this process simpler.

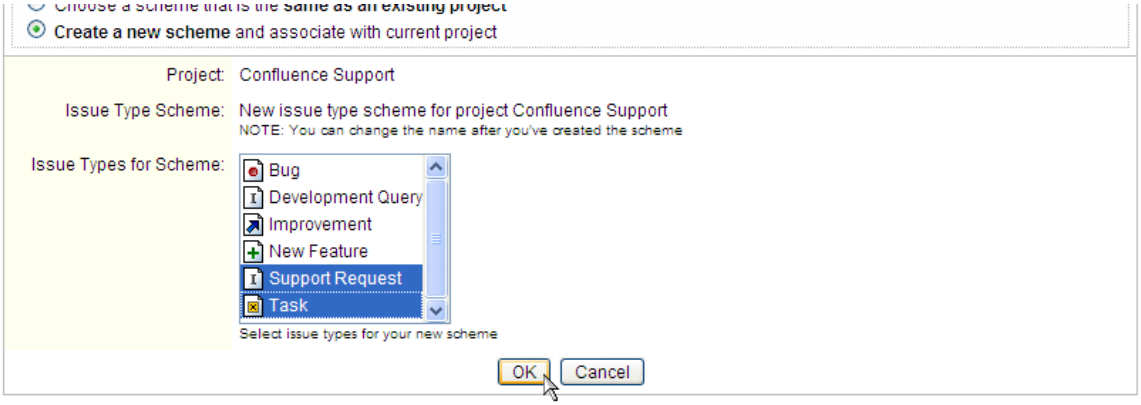
1. Click the '**select**' link for Issue Type Scheme on the View Project page.


2. This will bring you the screen below.


3. There are three ways you can select your issue type scheme. Select the radio button that is most relevant.
    a. *Choose an existing issue type scheme* — If you know the name of you scheme (e.g. Support Issue Type Scheme), you can immediately choose it from the list. You will see a preview of issue types that would be available for your project as well as the description of the scheme.
    b. *Choose a scheme that is the same as an existing project* — If you do not know the name of the scheme you would like to use, but you do know the name of the project whose set of issue types you wish to use for the project you are editing, please select this option. You will be prompted to select a project and the scheme that is currently associated with the selected project will be used for your project as well.


    c. *Create a new scheme and associate with current project* — Select this option if you can't find any existing scheme that fits your needs and would like to quickly create a new scheme. Simply select the relevant issue types for your project and a new scheme

will be created with the default name and order. You can edit the name, default value and order of the newly created scheme later.



4. If after you make your changes there are any issues in the selected project that will have obsolete issue types, they will have to be migrated with the JIRADOC:Issue Type Migration Wizard.

Issue Type Migration Wizard

This Issue Type Migration Wizard allows you to migrate issues from an obsolete issue type to a valid issue type. The wizard will be triggered whenever an action (e.g. editing a project's issue type scheme) results in an issue type becoming obsolete (not available in the scheme).

The wizard bears some resemblance to the Bulk Move function except for that you can't change the project of the issues. The major steps are:

1. Overview — provides a summary of the issues that will require migration
2. Choose Issue Type
3. Set new status
4. Set field values
5. Confirmation

Steps 2 to 4 will be repeated for each issue type that requires migration. After you have migrated all the issues you'll see a summary of changes that will occur. If you click the '**Confirm**' button, the wizard will migrate your issues to the new issue types and then complete your action.

Please refer to the Bulk Move documentation for more information on status changes and setting of fields values.

Defining 'Priority' Field Values

An issue's *priority* is its importance in relation to other issues.

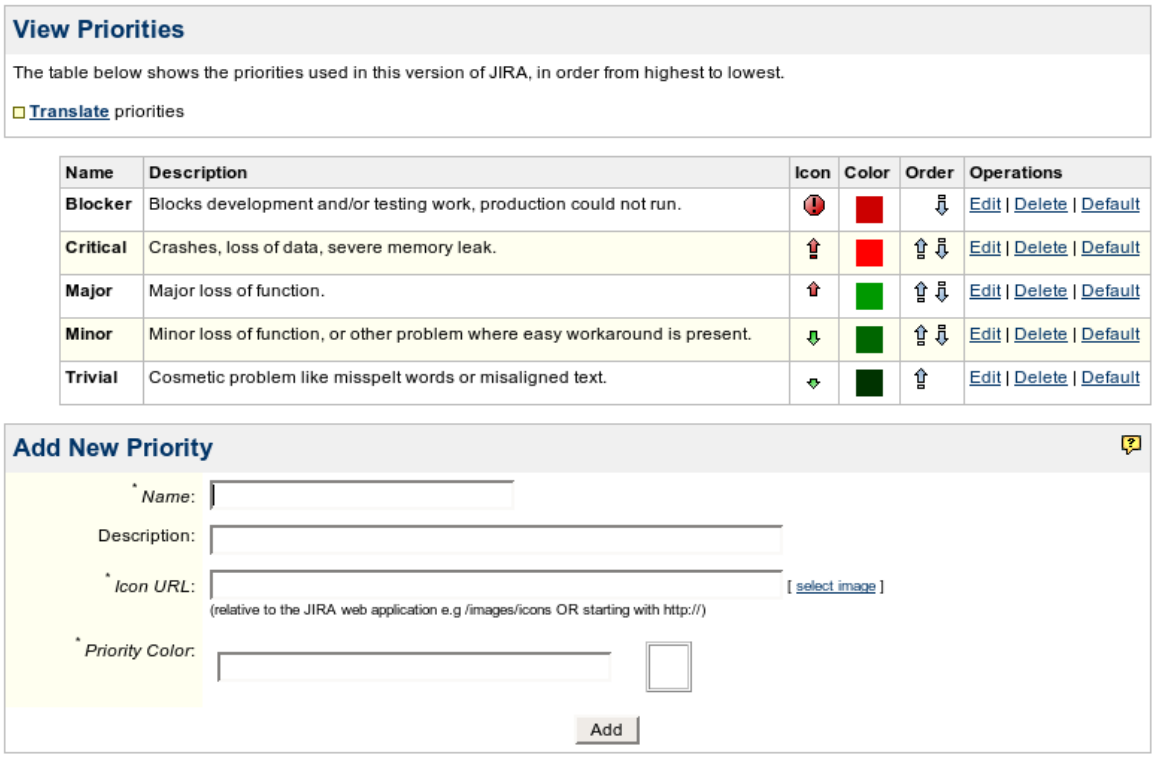
JIRA ships with a set of default priorities. You can modify these or add your own as follows.

On this page:

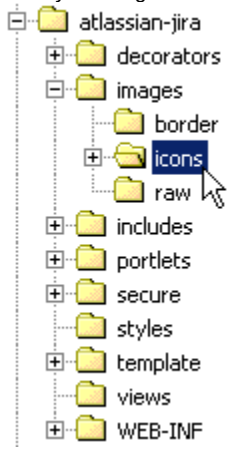
- Defining a new priority
- Editing a priority
- Re-ordering priorities
- Translating priorities
- Deleting a priority

Defining a new priority

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand panel, under the title **'Issue Settings',** click the **'Priorities'** link.
4. The **'View Priorities'** page will appear. This page lists the currently-defined Priorities, below which is the **'Add New Priority'** form.



5. In the **'Name'** field, type a word or two to describe your new priority. (The Name will appear in the drop-down field when a user creates or edits an issue).
6. In the **'Description'** field (optional), type a sentence or two to describe when this priority should be used.
7. In the **'Icon URL'** field, specify an image file to represent this priority. The dimensions of the image must be 16-pixels by 16-pixels. You can either type a URL, or click the **'Select image'** link to browse to a file location somewhere inside your JIRA installation directory, usually in /images/icons:



8. In the **'Priority Color'** field, specify a colour to represent this priority. You can either type the HTML colour code, or click the box at the right of the field to select from a colour chart.
9. Click the **'Add'** button.

Editing a priority

1. Go to the **'View Priorities'** page as described in steps 1-4 of **'Adding a priority'** (above).
2. Click the **'Edit'** link corresponding to the priority you wish to edit.
3. Update the fields as described under **'Defining a new priority'** (above), then click the **'Update'** button.

Re-ordering priorities

Re-ordering priorities changes the order in which they appear in the drop-down list when a user creates or edits an issue.

1. Go to the **'View Priorities'** page as described in steps 1-4 of **'Adding a priority'** (above).
2. To re-order the priorities, click the arrows in the **'Order'** column:
    - Click the up-arrow to move a priority higher up in the list.
    - Click the down-arrow to move a priority lower down in the list.

Translating priorities

To translate your priorities into another language, please see Translating Resolution/Priority/Status/Type.

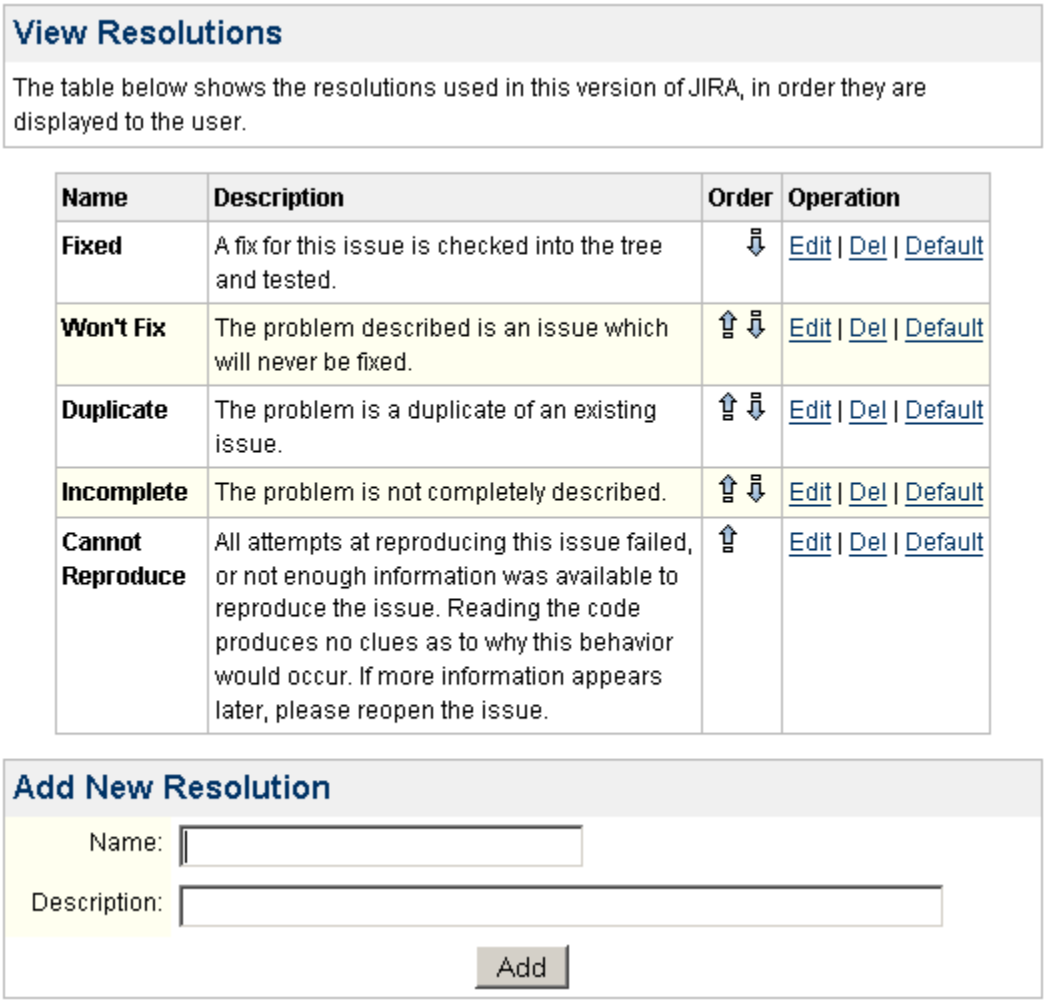
Deleting a priority

1. Go to the **'View Priorities'** page as described in steps 1-4 of **'Adding a priority'** (above).
2. Click the **'Del'** link corresponding to the priority you wish to delete.

Defining 'Resolution' Field Values

Resolutions are the ways in which an issue can be closed. JIRA ships with a set of default resolutions, but you can add your own. To do so, follow the following steps.

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title '**Issue Settings**', click the link labelled '**Resolutions**'.
4. This will bring up the 'View Resolutions' page. The page lists the standard resolutions, along with a form underneath to add new resolutions.



5. To add a new resolution, fill in the 'Add New Resolution' form. For the name put a short phrase that best describes your new resolution. For the description, put a sentence or two to describe when this resolution should be used. Click the '**Add**' button to add your new resolution.
6. The View Resolutions table can be used to edit, delete, set as default, and re-order the resolutions as they are displayed to the user who is resolving an issue.

Defining 'Status' Field Values

Statuses are used to represent the position of the issue in its workflow. A workflow represents a business process, represented as a set of stages that an issue goes through to reach a final stage (or one of the final stages). Each stage in the workflow (called a *workflow step)* is linked to an

issue status, and an issue status can be linked to only one workflow step in a given workflow.

JIRA ships with a set of default statuses that are used by the default workflow. In JIRA *Enterprise* and *Professional editions* you can add your own statuses and customise the workflow. In *Standard edition,* workflow is not customisable and you cannot add new Statuses — but in all editions you can change the names, descriptions and icons of existing Statuses.

Defining a New Status

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Open the left-hand sub-menu titled "Issue Settings" if it is not open already and click on the link labelled "Statuses".
4. This will bring up the *View Statuses* page. The page lists the existing Statuses, along with a form underneath to add a new Status.

Status Details	Mode	Workflows	Operations
Open The issue is open and ready for the assignee to start work on it.	Active	jira Copy of jira	Edit
In Progress This issue is being actively worked on at the moment by the assignee.	Active	jira Copy of jira	Edit
Reopened This issue was once resolved, but the resolution was deemed incorrect. From here issues are either marked assigned or resolved.	Active	jira Copy of jira	Edit
Resolved A resolution has been taken, and it is awaiting verification by reporter. From here issues are either reopened, or are closed.	Active	jira Copy of jira	Edit
Closed The issue is considered finished, the resolution is correct. Issues which are not closed can be reopened.	Active	jira Copy of jira	Edit

5. To add a new Status, fill in the Add New Status form. For the name put a short phrase that best describes your new Status. For the description, put a sentence or two to describe what stage this Status represents. For the Icon URL you need to supply the path of a 16x16-pixel image that has been placed somewhere inside JIRA's opened `.WAR`. We suggest you place it in `/images/icons`: !statuses-icondir.png|title="Directory view of icons dir in a JIRA webapp!JIRA ships with a number of images that can be used as status icons. These images are located in the `/images/icons directory` inside the JIRA .war and include:
    - status_assigned.gif
    - status_closed.gif
    - status_document.gif
    - status_down.gif
    - status_email.gif
    - status_generic.gif
    - status_information.gif
    - status_inprogress.gif
    - status_invisible.gif
    - status_needinfo.gif
    - status_open.gif
    - status_reopened.gif
    - status_resolved.gif
    - status_trash.gif
    - status_unassigned.gif
    - status_up.gif
    - status_visible.gif
6. The View Statuses table can be used to edit and delete Statuses. Please note that only statuses that are not used in any workflow can be deleted.

Translating Resolutions, Priorities, Statuses and Issue Types

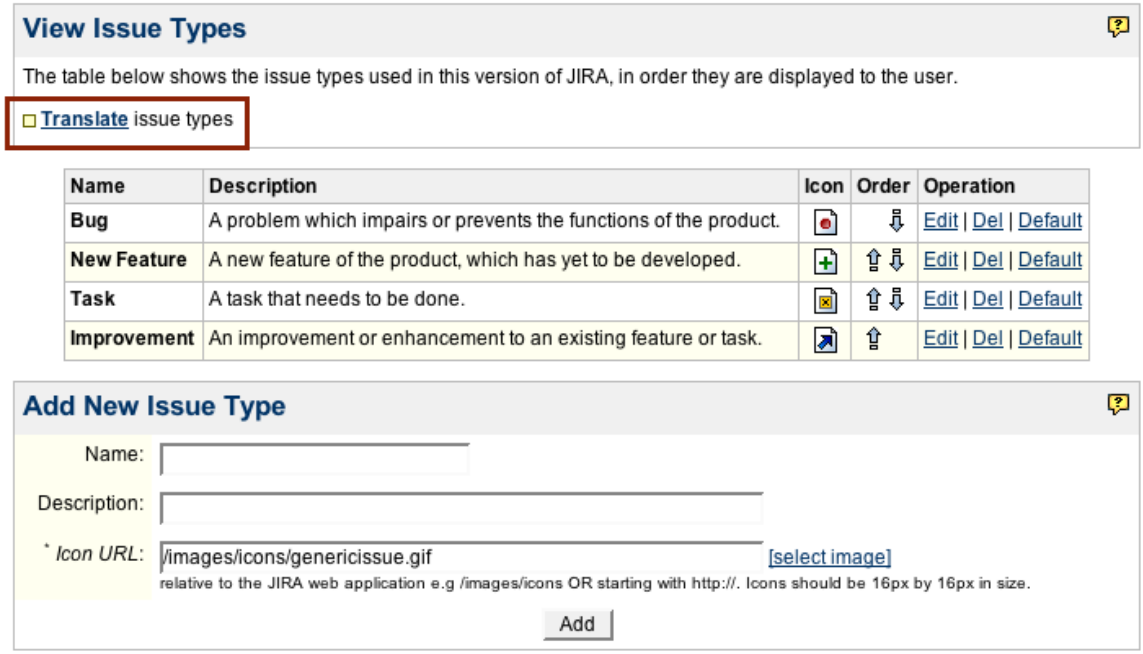
Further extending JIRA as an internationalisable issue manager, it is possible to specify a translated name and description for each issue constant — i.e all Issue Type, Status, Resolution and Priority fields.

This functionality allows the administrator to specify an issue constant translation set for each available language - providing each user with a more complete translation in their own chosen language. The translated issue constant names and descriptions appear throughout JIRA; in reports, portlets and all issue views.

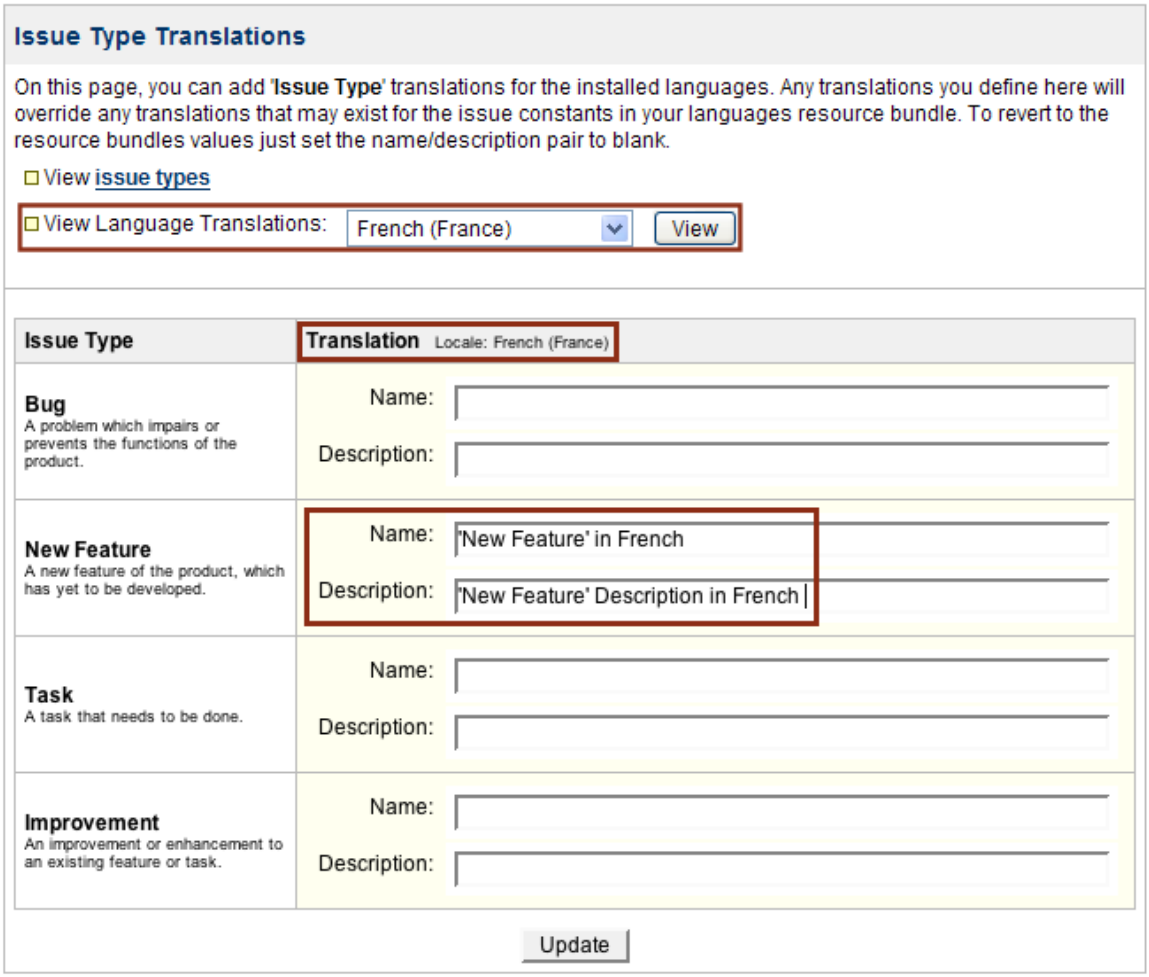
Issue Constant Translation

Each issue constant can be configured to have a translation set for each available language. The default issue constant name and description are displayed if no translation has been configured.

1. The issue constant translation operation is accessed through the **Translate** link located on each issue constant management screen.



2. The issue constant translation screen displays the translation set for the currently selected language. In order to view/update a translation set for a specific language, select the required language from the list at the top of the screen and click the **View** button.



3. The currently selected language is displayed above the translation set table.
4. A translated name and description set can be specified for each type of issue constant. Once all translations have been entered, the translation set can be saved by clicking the **Update** button at the bottom of the translation table.
5. The process can be repeated for the all types of issue constants - i.e. Issue Type, Status, Resolution and Priority fields.
6. The translated issue constant name and description will be displayed throughout JIRA; in reports, portlets and all issue views.

The default issue constant name and description are displayed if a translation has not been specified.

Adding a Custom Field

On this page:

- Custom Fields Overview
  - Custom Field Types
  - Search templates
  - Custom field context
  - Custom field configuration schemes
- Adding a Custom Field
  - Steps to define a custom field
  - Additional Resources

Custom Fields Overview

Custom field types were introduced in JIRA 2.0 to allow greater customisability of the types of data collected with your issue. In 3.0, the number types have been expanded and you can even add your own custom field types. JIRA 3.2 adds a new level of flexibility to your custom fields. You can now configure your custom fields to only appear for certain issue types in certain projects or multiple issue types over multiple projects. On top of that, you can even configure each custom field differently for each context

This page outlines some of the key concepts relating to custom fields.

To build your own custom field types, check out the tutorial at the JIRA Development Hub.

Custom fields are always optional fields. This means you can add custom fields without requiring existing issues to be changed. The current issues contain no value for the custom field, even if a default is defined.

Custom Field Types

JIRA now ships with over 20 custom field types and you can find more custom field types and other examples in the JIRA Extensions space (e.g. JIRA Toolkit). A sample of the types are listed below.

Custom Field Type	Description
Cascading Select	Multiple select lists where the options for the second select list dynamically updates based on the value of the first
Date Picker	Input field allowing input with a date picker and enforcing valid dates
Date Time	A custom field that stores dates with a time component.
Free Text Field (unlimited text)	Multiple line text-area enabling entry of longer text strings
Multi Checkboxes	Checkboxes allowing multiple values to be selected
Multi Select	Select list permitting multiple values to be selected
Number Field	Input field storing and validating numeric (floating point) values
Project Picker	Select list displaying the projects viewable by the user in the system
Radio Buttons	Radio buttons ensuring only one value can be selected
Select List	Single select list with a configurable list of options

Text Field	Basic single line input field to allow simple text input of less than 255 characters
URL Field	Input field that validates a valid URL
User Picker	Choose a user from the user base via a popup picker window.
Multi User Picker	Choose one or more users from the user base via a popup picker window.
Group Picker	Choose a user group using a popup picker window.
Multi Group Picker	Choose one or more user groups using a popup picker window.
Single Version Picker	Choose a single version from available versions in the project.
Version Picker	Choose one or more versions from available versions in the project.

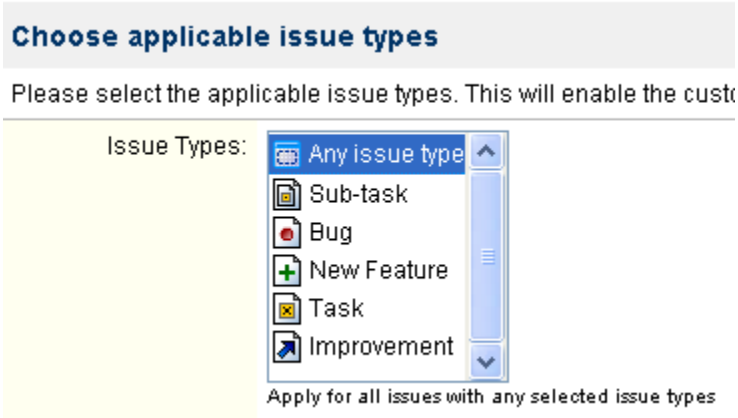
Search templates

Search templates are responsible for indexing a custom field as well as making it searchable through the Issue Navigator (note that custom fields are not searchable via QuickSearch). Each of the default custom field types has a related pre-configured search template.

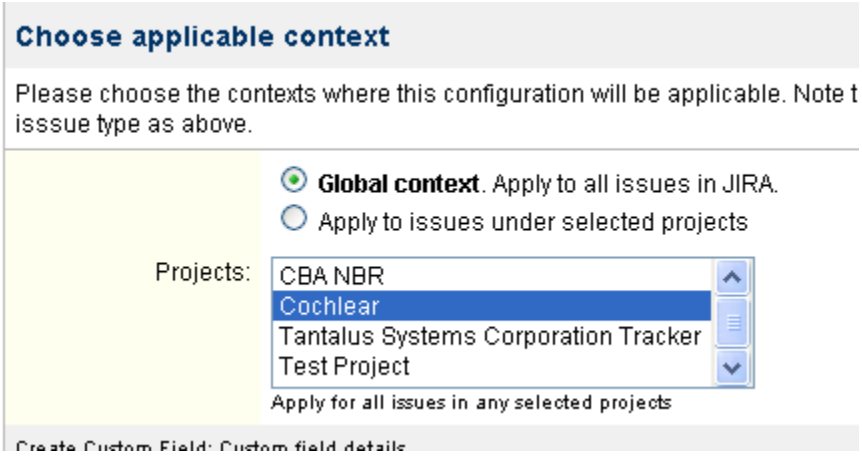
Custom field context

The custom field context (introduced in JIRA 3.2) allows your custom field to be configured (that is, enabled) for any numerous different combinations of issue types and projects. You can have different default values in different projects, different options for different projects and the like.

The context is made up of an issue type component and a project component. You can select multiple issue types and multiple projects or declare the custom field to be global.

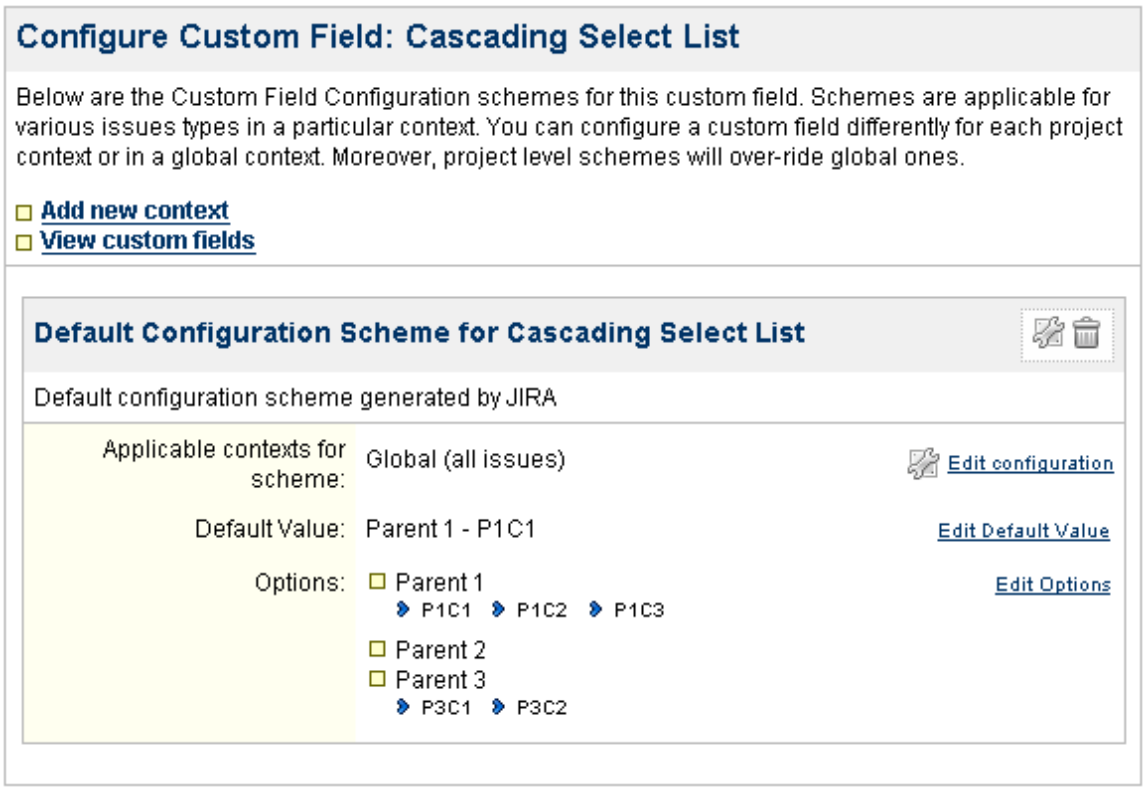


The context itself can now be modified at any time. You can change the project or issue type applicable for the custom field at any time.



Custom field configuration schemes

If you start digging deeper into custom fields (or indeed, any part of JIRA) you'll notice many references to schemes. Custom field configuration schemes are how JIRA allow you to manage custom field contexts and configuration. A configuration scheme is configuration set for a group of issue types for a set of projects. If you have two different default values, you'd need two configuration schemes and so on.



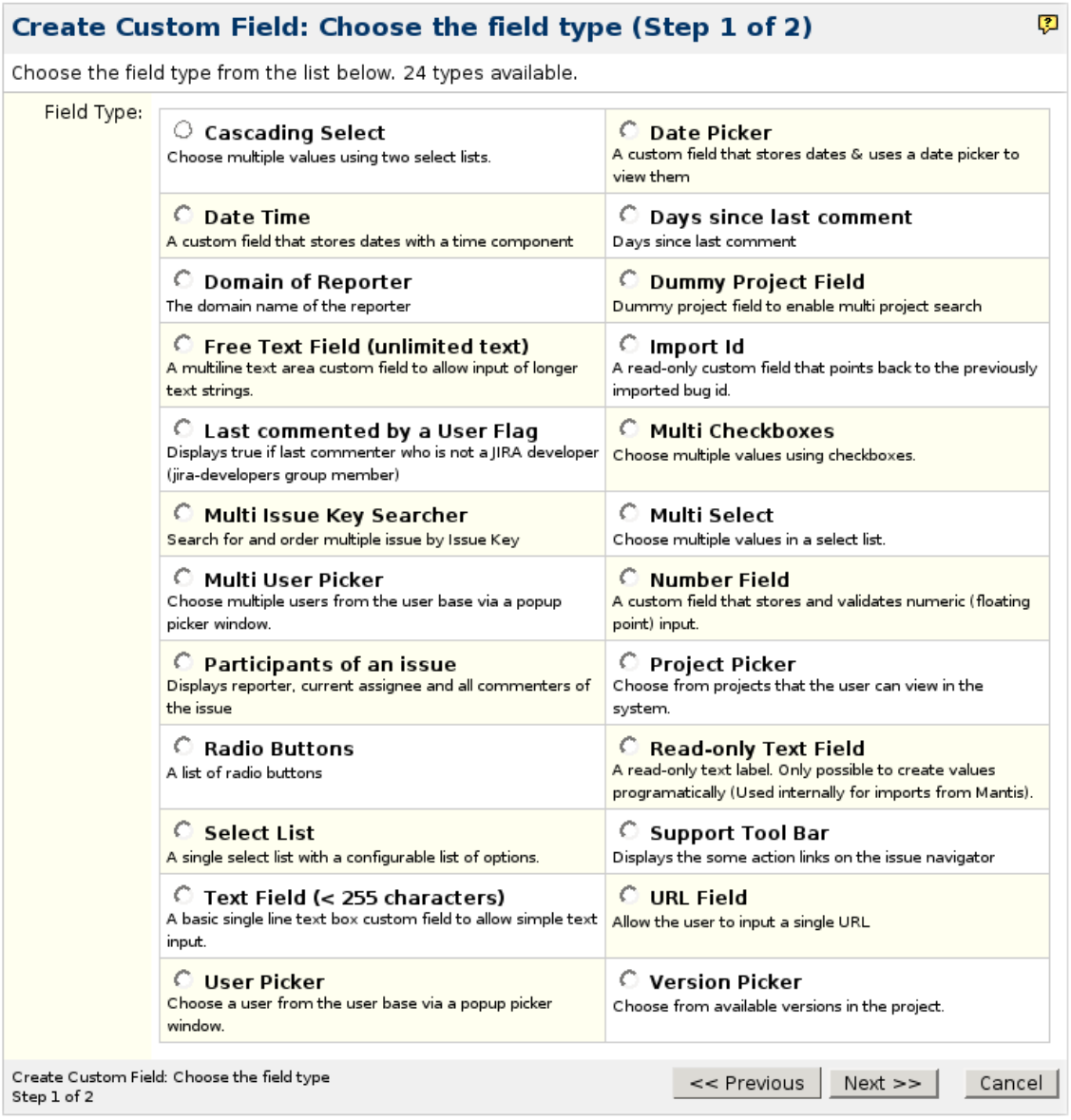
Specific project based configuration schemes will override configurations from a Global project context. So you could configure a **default** global scheme for all projects and the configure for each projects that are different. You may for example, have a global select lists that have values that applies for 80 of your 81 projects but is different in the other. You'd configure a one configuration scheme for global context and other for the specific field that is different.

Also note that to avoid conflicts, a project can only be part of a single configuration scheme. Once you've selected a specific project to be part of a scheme, it will be removed from the list of selectable options

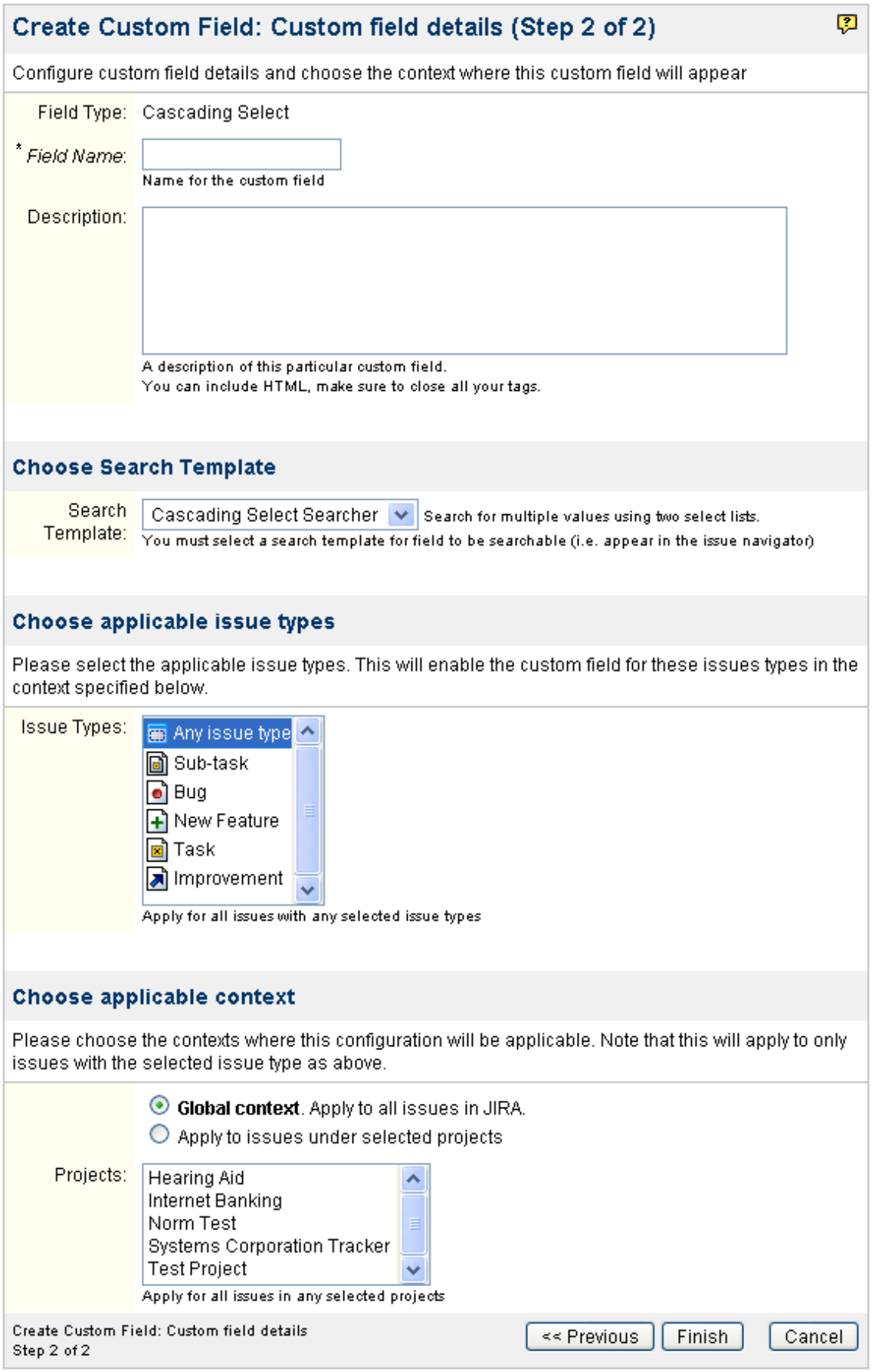
Adding a Custom Field

Steps to define a custom field

1. Log in as a user with the ' **JIRA Administrators** ' global permission .
2. Bring up the administration page by clicking either the ' **Administration** ' link on the top bar or the title of the Administration box on the dashboard:
3. On the menu bar on the left, under "Issue Fields" sub-menu click the Custom Fields link, and then click on the Add Custom Field link on the presented page.

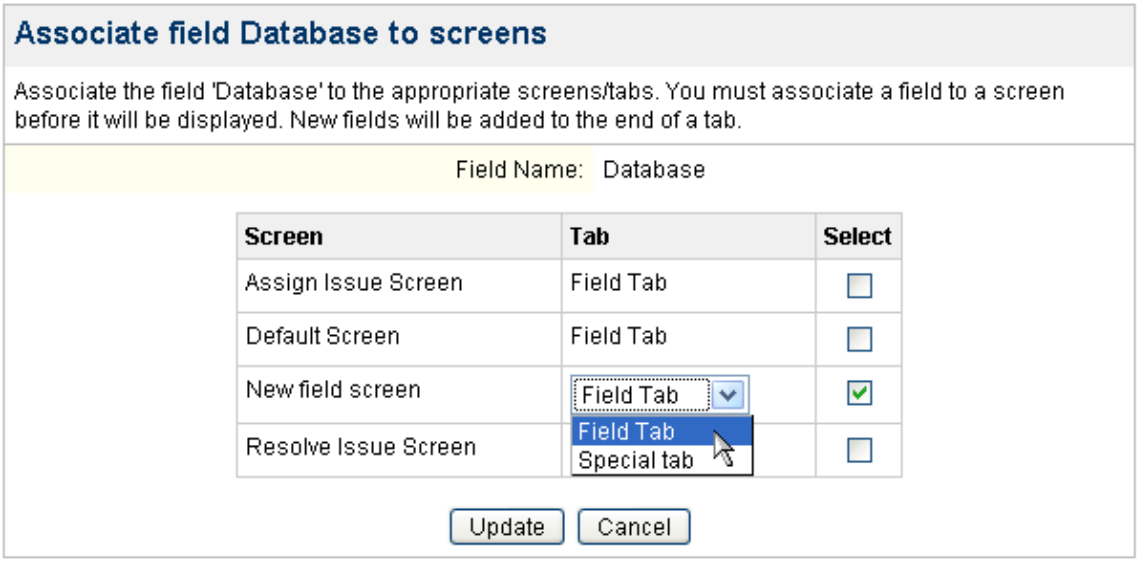


4. Select from the list the appropriate custom field type.
5. Click on the next button.



6. Fill in the Field Name and Field Description. The Field Name will appear as the custom field's title in both entering and retrieving information on issues. The Field Description is displayed beneath the data entry field when entering new issues and editing existing

issues, but not when browsing issues.
7. Select an appropriate Search Template. Pre-configured search templates are available for each shipped custom field type. A description of each search template will appear next to the select list when you select one.
8. Select one or any number of issue types that this custom field will be applicable for. You can change this value in the future if you need to.
9. Select the applicable project context. The custom field will be available to the selected projects. If issue types were chosen, it will only appear for those issue types for that project
10. Click **Finish.**
11. This will bring you to the screen association page where you can put your newly created custom field onto a screen in JIRA. You can associate the field any screens or tabs in JIRA. You must associate a field to a screen before it will be displayed. New fields will be added to the end of a tab.



12. Clicking **Update** will return you to the View Custom Fields page that displays a summary of the custom fields in the system. You can edit, delete or configure custom fields here. This page is also directly accessible from the menu bar to the left of all Administration pages.

Additional Resources

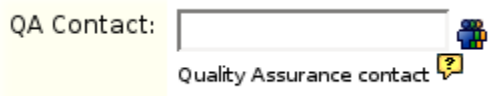
- Adding a custom field tutorial video - Watch this short tutorial video to see how to add and configure a new custom field. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Creating Help for a Custom Field

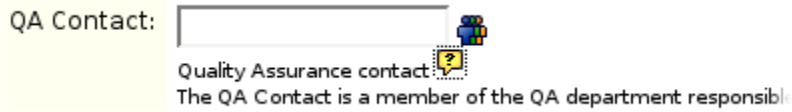
To provide online help for a custom field, use HTML or Javascript in the field's description. E.g. you can have a simple link to an external help page:

```
<a href="http://www.mycompany.com/jirahelp/fieldhelp.html">get help</a>
```

Or using Javascript, you can have help text right in the field:



where clicking the help icon makes hidden help text appear:



This can be done by entering the following as the field's description:

```
Quality Assurance contact
<script type="text/javascript">
  function showHelp() {
        var listenersDiv = document.getElementById("qaFieldHelp");
        if (listenersDiv.style.display == 'none') {
          listenersDiv.style.display = '';
        } else {
          listenersDiv.style.display='none';
        }
  }
</script>
<a href="#"  onclick="showHelp(); return false;"><img src="/images/icons/help_blue.gif"/></a>
<div id="qaFieldHelp" style="display:none">
  The QA Contact is a member of the QA department responsible for taking this issue through testing.
  They will be notified by email of this and subsequent issue state transitions.
</div>
```

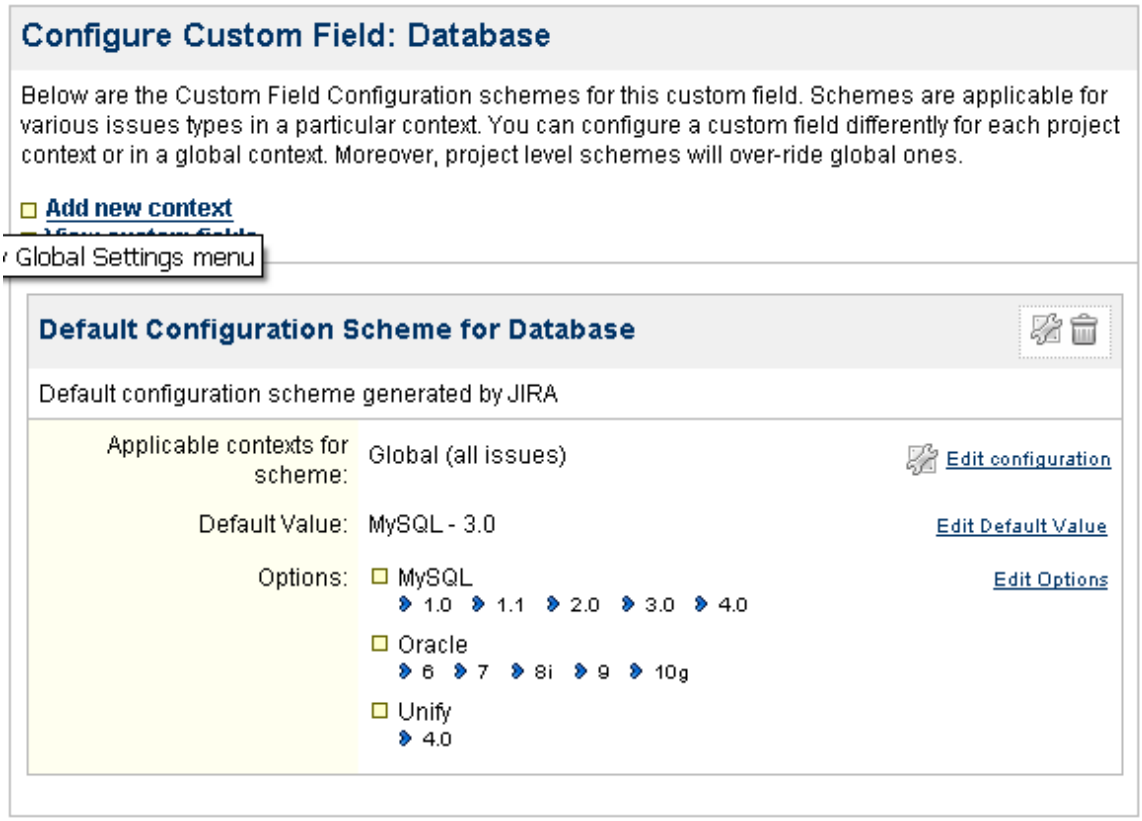
(Incidentally, Javascript in descriptions can also be used to set field values.)

Configuring a Custom Field

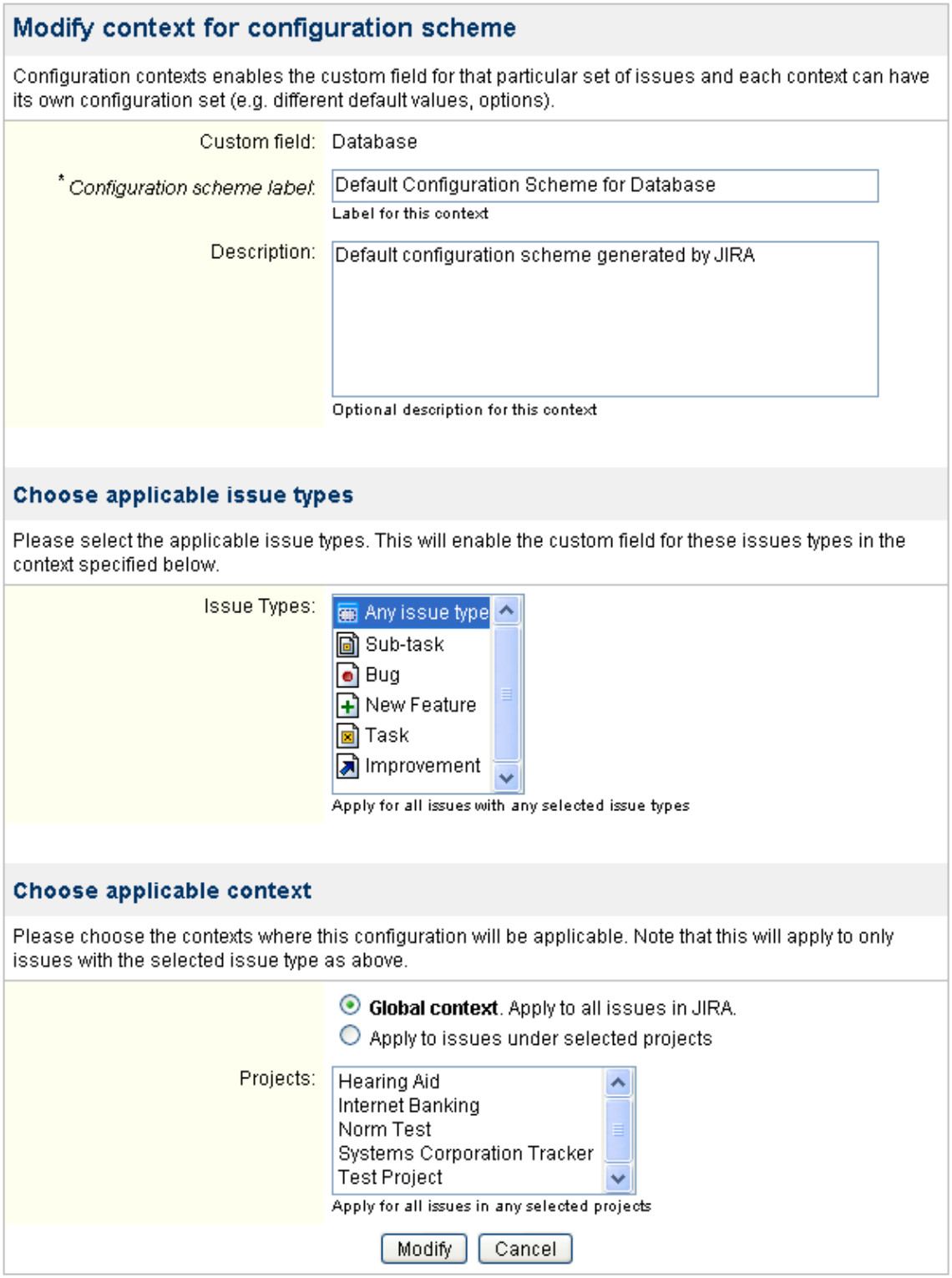
On this page, we show you how you can configure your custom field after you've created them.

Configuring Custom Fields

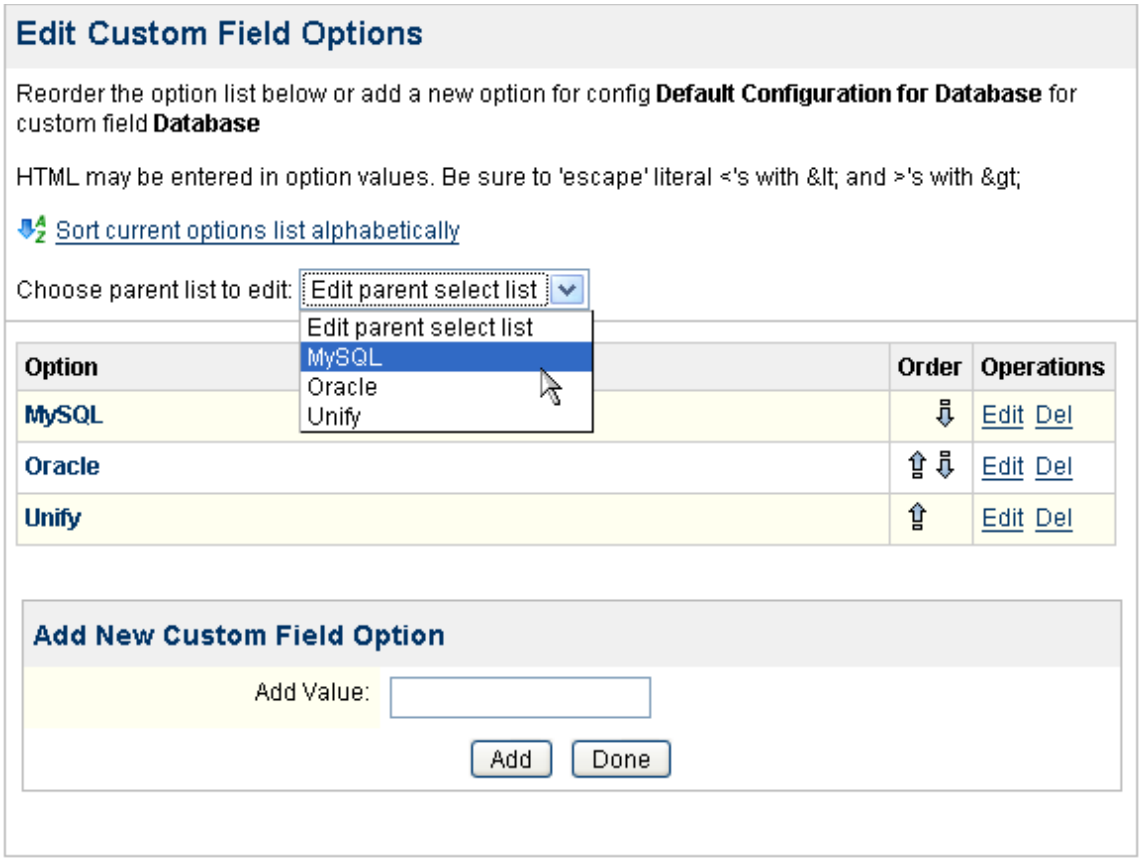
For most custom fields, you can configure its default values and set the options for lists. When you click on the *configure* link, you'll be faced with a page similar to the one below.



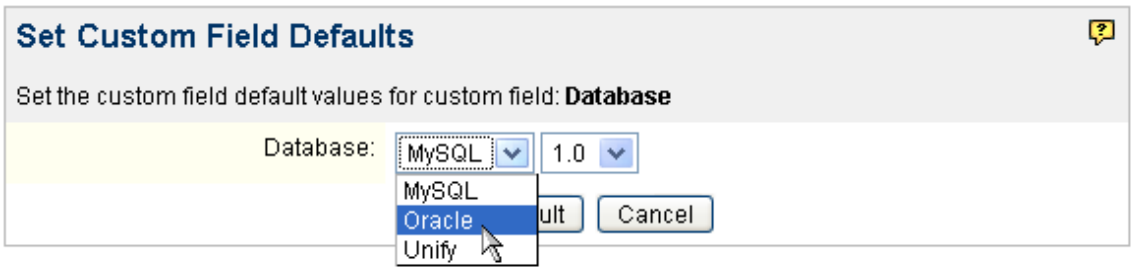
You'll notice that there is a configuration scheme named *Default Configuration Scheme for....* This is the configuration scheme created automatically by JIRA when you initially added your custom field. The *Applicable contexts for scheme* refers to context that this scheme will be applied to; which projects & issue types the defaults and options will be displayed. For most people this will be only thing you need to know about custom field configuration schemes. You can edit the context by clicking *Edit configuration* or the edit icon on the top left hand corner.



Here you can edit the label and description of the configuration scheme. These are used for administrative purposes only and isn't shown to the end users. You can also change the context that this scheme is to be applied to here. It's now easier than ever to change the projects or issue types for the configuration context.



Select lists, multi select lists and cascading selects lists can have their options manipulated. You can add, remove and sort the options alphabetically. You can also have HTML in an option value. Be sure to use complete all tag pairs and ensure that it will display correctly.

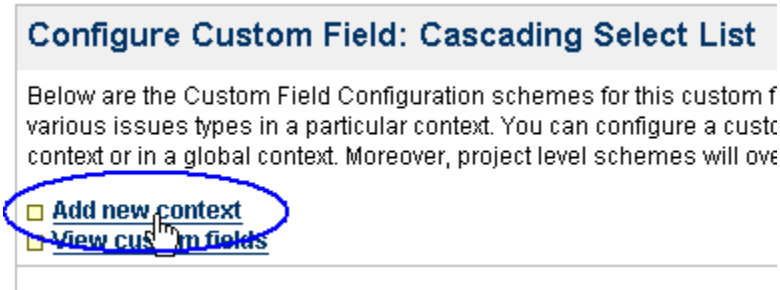


Click the *Edit defaults* to modify the default value of a custom field for this configuration scheme. Setting the defaults of will take you to a screen that is particular to the issue type. Certain custom fields such as calculated custom fields may not allow for defaults to be selected and will not have the "edit defaults" link.

Managing multiple configuration schemes

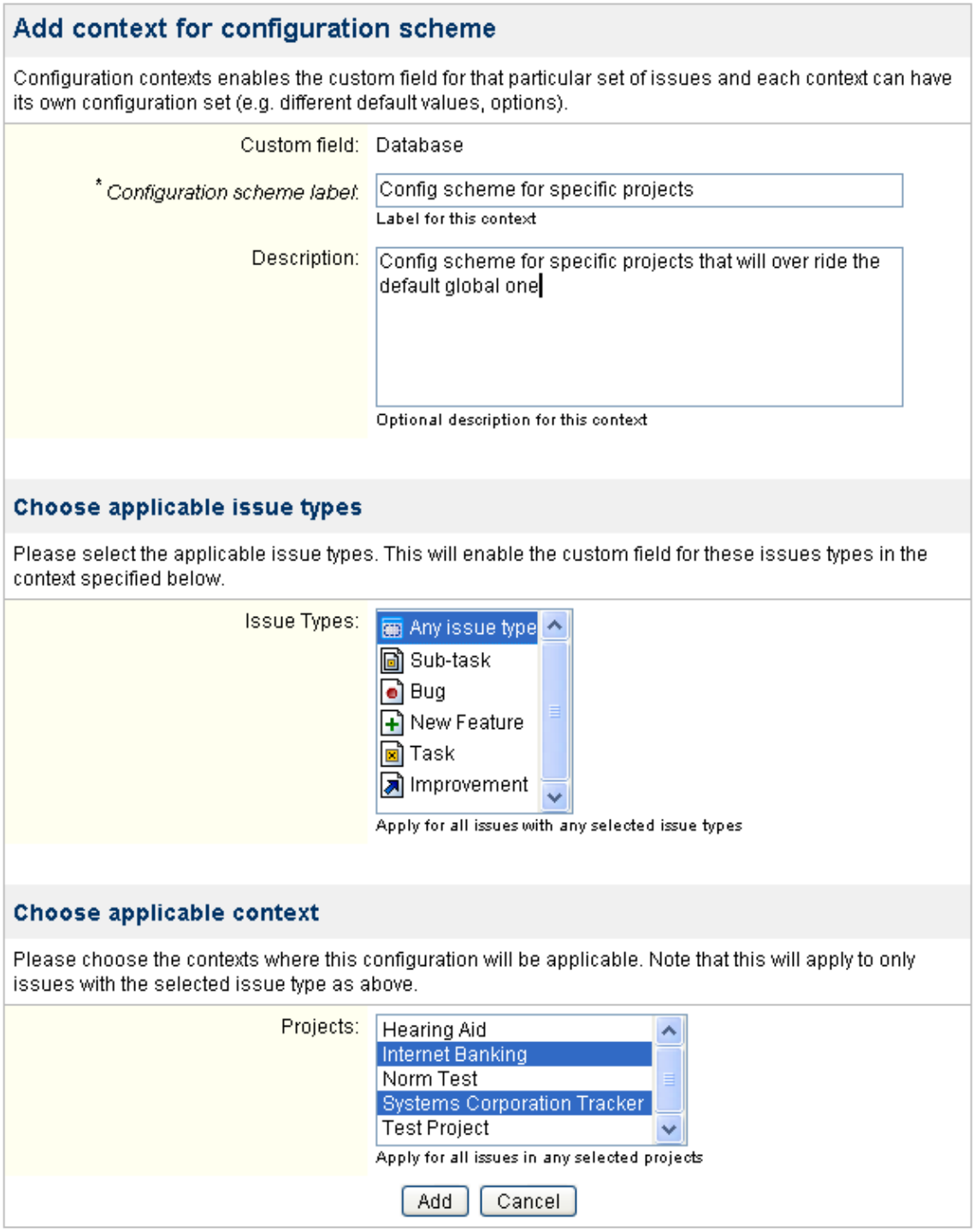
Since JIRA 3.2, it is now possible to configure a custom field differently for different issue types and project combinations. This can be achieved through configuring different custom field schemes.

1. Click on the *Add context* link to create a new custom field configuration scheme.

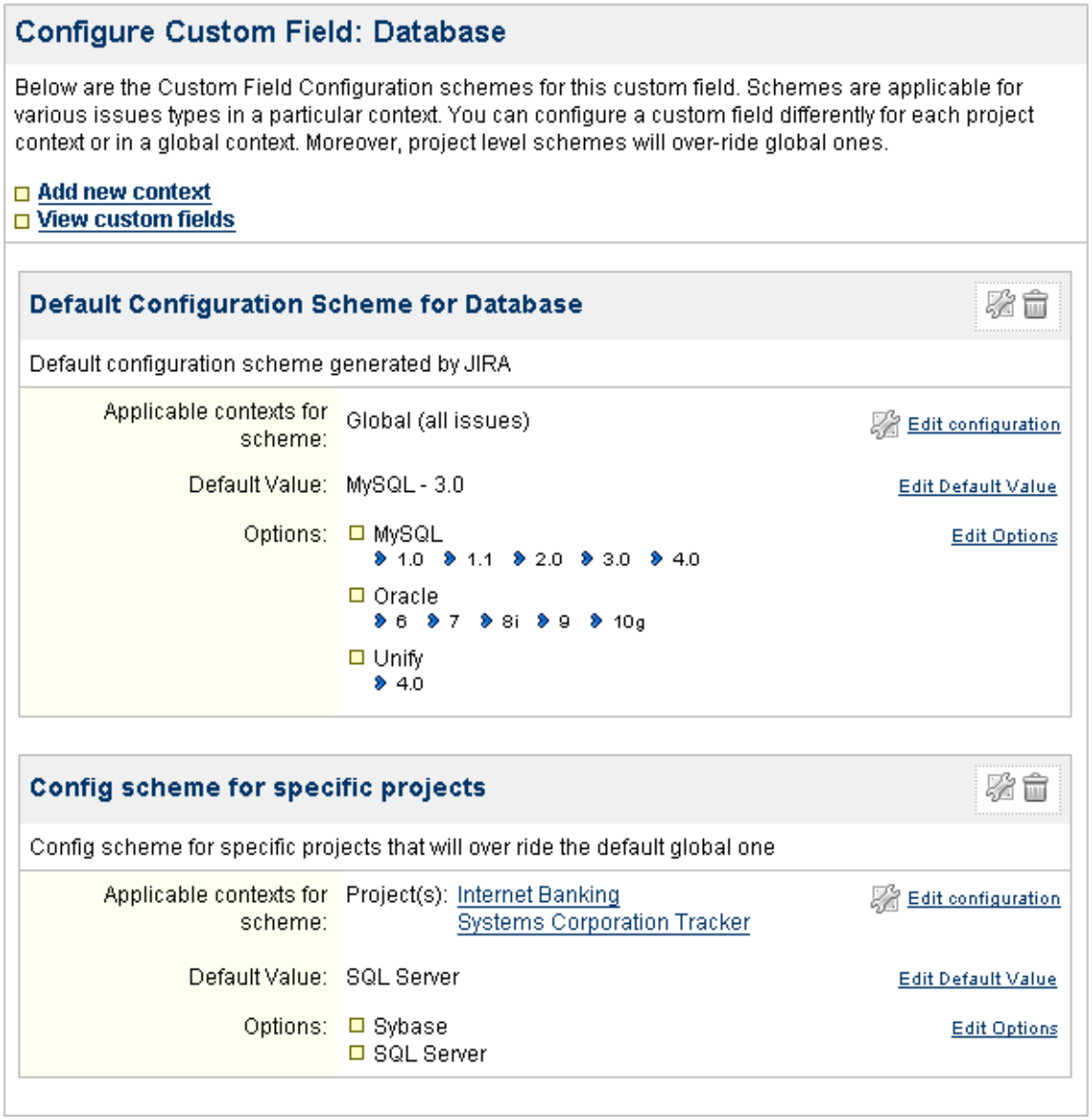


2. You'll see a screen that very similar to the edit configuration scheme. Here you can select the applicable context for your new configuration scheme. One difference is that the project context will only show projects and options that have not be previously selected

in another configuration scheme. So if you already have an existing "Global" configuration scheme, that option will not be available. Moreover, be aware that project specific configuration scheme will override Global configurations. For example, if you have one global configuration scheme with one default value and a project configuration scheme with a different value, the project value will be used instead.



3. Click *Add*
4. You will now have a new configuration scheme that can be reconfigured separately to the default scheme. You can add different default values and options for each one. In the screen show below, the *Database* custom field have specialised options and defaults for two to the projects, which will over-ride the global options list.



Additional Resources

- Adding a custom field tutorial video - Watch this short tutorial video to see how to add and configure a new custom field. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Specifying Field Behaviour

On this page:

- What is a 'Field Configuration'?
- Editing a Field Configuration
  - Editing a Field's Description
  - Hiding/Showing a field
  - Required/Optional fields
  - Renderers
- Managing Multiple Field Configurations
  - Default Field Configuration
  - Adding a Field Configuration
  - Editing a Field Configuration
  - Deleting a Field Configuration
  - Copying a Field Configuration
- Activating a Field Configuration

What is a 'Field Configuration'?

A **Field Configuration** provides the ability to change field behaviour. For each field, a Field Configuration specifies:

- the **description** that appears under the field when an issue is edited
- whether the field is **hidden** or **visible**
- whether the field is **required** (i.e. the field will be validated to ensure it has been given a value) or **optional**
- (for text fields only) which **renderer** to use

A Field Configuration specifies individual behaviour for every issue field, including both standard fields and custom fields.

You can create multiple Field Configurations (see below). You can then associate different Field Configurations with different issue types by creating a Field Configuration Scheme. Once created, a Field Configuration Scheme can be associated with one or more projects, allowing you to control field behaviour on a per project, per issue type basis.

For information about placing fields on Screens, and using Screen Schemes to associate screens with issue operations, please see the Overview.

Editing a Field Configuration

To change the behaviour of fields, you first need to navigate to the Field Configuration. The way this is done depends on the edition of JIRA:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the **'Issue Fields'** sub-menu on the left-hand column, and choose '**Field Configurations**' from the list.
4. Locate the Field Configuration of interest, and click the '**Configure**' link. (Note: the 'Edit' link only allows you to change the Name and Description.)
5. This will bring you to the '**View Field Configuration**' page.

The "View Field Configuration" page lists all system and custom fields in a table as shown below. The "Operations" column lists all the operations that are available for each field. These operations could be:

- Edit — change the field's description
- Hide/Show — hide the field from view or show it
- Require/Optional — set a field to be required (so that whenever a field is edited it must be given a value) or optional.
- Renderers — change a field's renderer (see Configuring Renderers for more information).

View Field Configuration

The table below shows all fields configured in JIRA and their properties for **Default Field Configuration**.

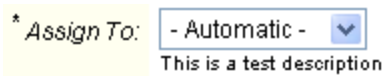
You can use this page to make fields required, hide/show fields and specify their description. You can also change the screens the field appears on by using the "Screens" link next to each field.

- View all field configurations
- Restore Defaults

Name	Screens	Operations
Affects Version/s	Default Screen	Edit | Hide | Required | Screens
Assign To	Assign Issue Screen Default Screen Resolve Issue Screen	Edit | Hide | Screens
Attachment	Default Screen	Edit | Show
Comment [Default Text Renderer]	This field can not be placed on screens by users.	Edit | Renderers
Component/s	Default Screen	Edit | Hide | Required | Screens
Custom text area [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Custom text field [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Description [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Due Date	Default Screen	Edit | Hide | Required | Screens
Environment [Default Text Renderer] For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).	Default Screen	Edit | Hide | Required | Screens | Renderers
Fix Version/s	Default Screen Resolve Issue Screen	Edit | Hide | Required | Screens
Issue Type Required	Default Screen	Edit | Screens
Priority	Default Screen	Edit | Hide | Required | Screens
Reporter	Default Screen	Edit | Hide | Required | Screens

Editing a Field's Description

Fields can be given descriptions to better identify the meaning of the field. These descriptions are usually displayed under the field on the creation of an issue and whenever it is edited. An example of this, is shown below.



To edit the description of a field:

1. On the 'View Field Configuration' page, click on the '**Edit**' link next to the field you want to change. This will bring you to the 'Edit Field Description' page.
2. On this page you can edit the field's description
3. Click the '**Update**' button.

Hiding/Showing a field

If your organisation or project has no use for a particular field, you have the option to hide it. Hiding a field will ensure that the the field does not appear on any Screens (i.e. issue operation Screens, workflow transition Screens) where the Field Configuration applies.

Hiding a field in the Field Configuration is distinct from not adding a field to a Screen. Fields hidden through the Field Configuration will be hidden in *all* applicable Screens, regardless of whether or not they have been added to the Screen.

> **For fields that have a default value:**
> If the field is hidden in the Field Configuration, then it will not receive a value when an issue is created, regardless of whether the field is present on the 'Create Issue' screen(s). (The following fields can have a default value: 'Resolution'; 'Status'; 'Priority'; 'Issue Type'; [custom fields.])

To hide a field:

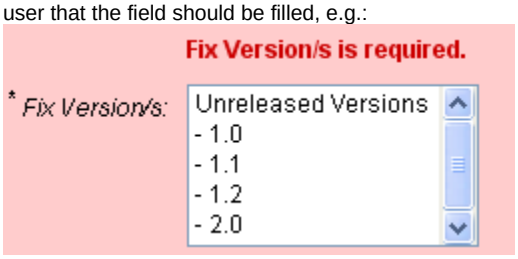
1. On the 'View Field Configuration' page, click the '**Hide**' link next to a field you no longer want. The field will then fade to grey to signify that it has been hidden.
2. As per the image above, the 'Environment' field has been hidden. You can make this field visible again anytime by pressing on the '**Show** ' link.

> The fields 'Summary' and 'Issue Type' cannot be hidden and as such there is no 'Hide' option available for these fields.

> **Please note:**
> If you hide the '**Fix For Version**' field, the Change Log and Road Map reports will not work.

Required/Optional fields

Certain fields within your organisation may be compulsory for issues. In this case you can set a field to be required, so that JIRA validates that the field has been given a value whenever an issue is edited. If a required field has not been given a value, JIRA will return an error informing the user that the field should be filled, e.g.:



> If you set a field to "required", ensure that the field is present on your 'Create Issue' screen(s). Note that in you can have different field configurations for different projects and issue types (see 'Associating field behaviour with Issue Types'); so you need to ensure that all "required" fields are present on the 'Create Issue' screens for all associated projects and issue types (see 'Associating screens with Projects and Issue Types').

To make a field required:

1. On the 'View Field Configuration' page, click the '**Required**' link next to the appropriate field. The text "Required" will appear next to the field.
2. As per the image above, the 'Fix Versions' field has been made required. You can make this field optional again by clicking the '**Optional**' link.

> Fields that are hidden cannot be set to required. Making a hidden field required will make it "shown" as well.

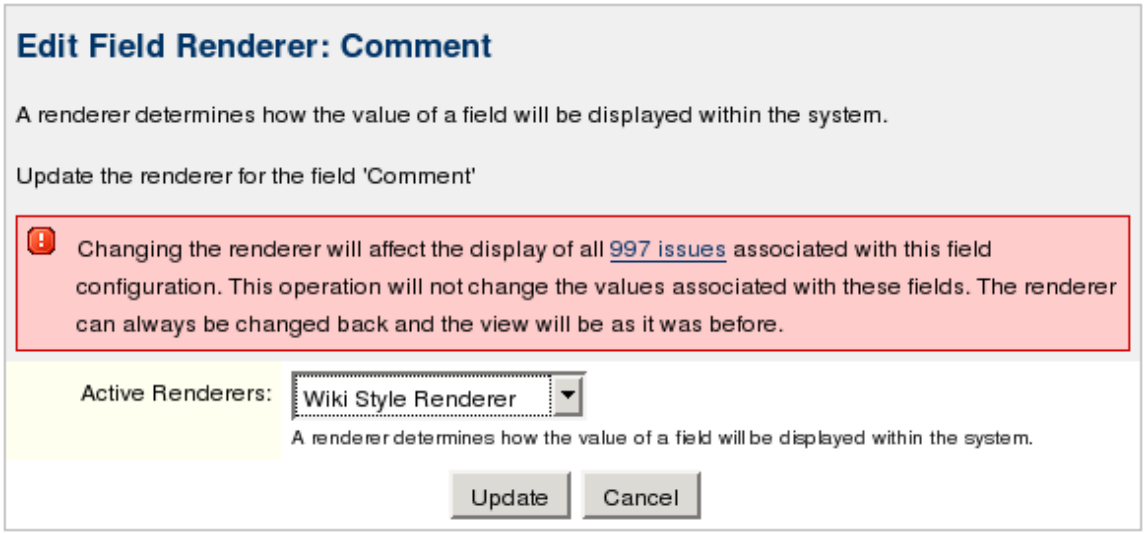
Renderers

> Before you begin, please read 'Configuring rich-text renderers', paying particular attention to the section 'Implications for JIRA operations'.

The "View Field Configuration" page indicates which renderers are currently enabled for all renderable fields.

In the above screenshot you will notice the grey text '[Wiki Style Renderer]' under the Description field's name. This indicates that the field is currently configured to use the Atlassian Wiki Renderer. The grey text '[Default Text Renderer]' under the Comment field's name indicates that the field is currently configured to use the Default Text Renderer.

To change the renderer type for a specific field, click on the '**Renderers**' link in the 'Operations' column of the screen for the field you want to change. This will take you to a page where you will have the option to select a renderer from all configured and available renderers.



As shown above, this page will warn you if there are issues that will be affected by the change. If no issues will be affected then the warning does not show. From this page, choose the renderer you wish to use and click '**Update**'. You are then presented with a confirmation page, shown below.

Click the '**Update**' button to finish setting the new renderer on the field.

> Changing the renderer only affects the *display* of the issue data that exists in the system. You can therefore toggle back and forth between renderer types safely.

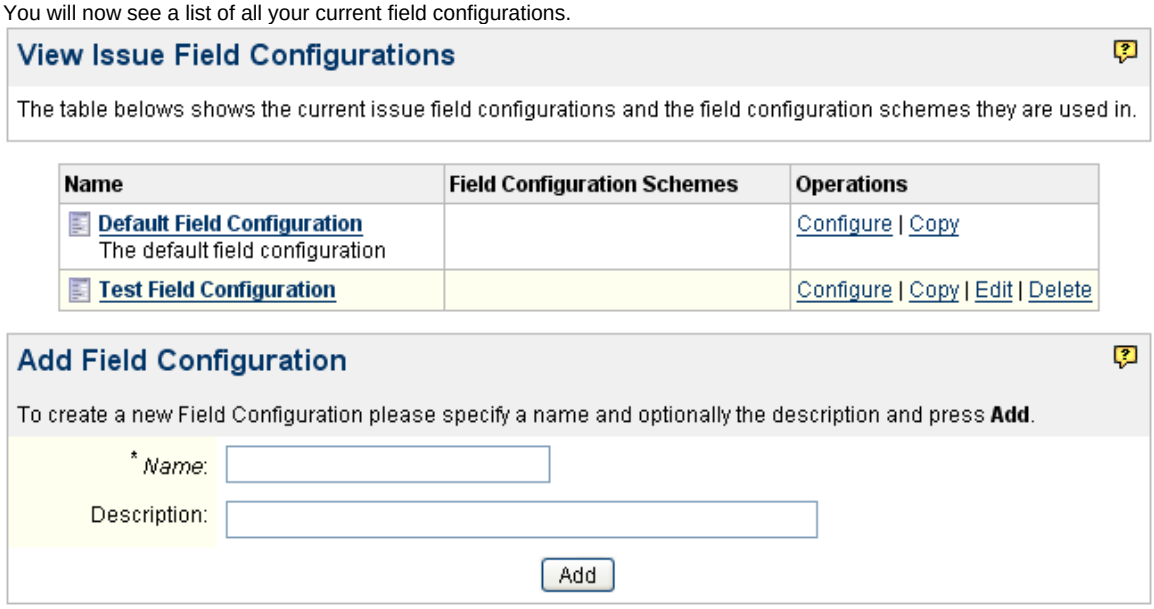
Managing Multiple Field Configurations

You can create multiple field configurations for use on separate projects and issue types.

> Multiple field configurations are organised into Field Configuration Schemes based on issue type. A scheme can be associated with one or more projects, allowing you to control fields on a per project per issue type basis. See Associating Field Behaviour with Issue Types for details.

To create, edit, delete and copy individual field configurations, go to the "View Field Configuration" page as follows:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**'' sub-menu on the left-hand column, and choose '**Field Configurations**' from the list.
4. You will now see a list of all your current field configurations.



Default Field Configuration

When JIRA is installed, the 'Default Field Configuration' is created automatically. All new projects are associated with this configuration. This configuration is also used for projects that are not associated with a Field Configuration Scheme..

> It is not possible to delete the Default Field Configuration.

Adding a Field Configuration

1. The '**Add**' new field configuration form is located at the bottom of the 'View Field Configuration' page
2. Enter the name of the new configuration on this form.
3. You can optionally add a description to this configuration for better identification.
4. Click on the '**Add**' button to submit the form. The page will be automatically updated with your new field configuration added.

> A newly created Field Configuration will not take effect until it has been activated (see below).

Editing a Field Configuration

1. On the 'View Field Configuration' page, click the '**Edit**' link next to the desired field configuration.
2. You will now see the 'Edit Field Configuration' page. Here it is possible to edit the configuration name and description.
3. Click on the '**Update**' button.
4. On completion you will be returned to the page where you can view your changes.

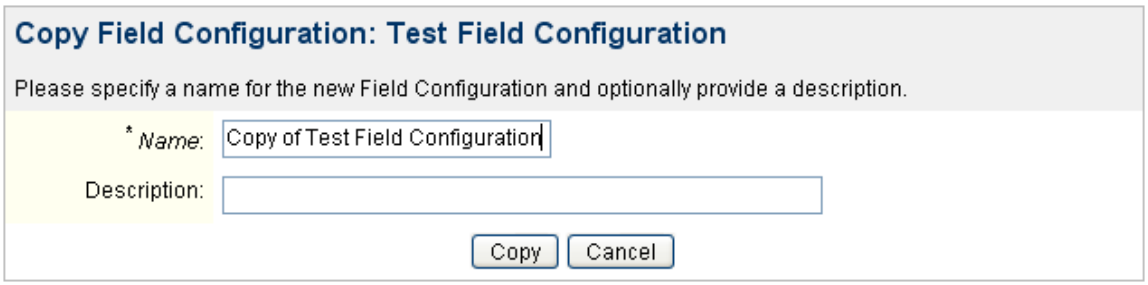
Deleting a Field Configuration

1. On the 'View Field Configuration' page, click the '**Delete**' link next to the desired field configuration.
2. Click the '**Delete**' button to confirm this operation.
3. The 'View Field Configuration' page will now show the updated list of the field configurations.

- The Default Field Configuration cannot be deleted.
- You can only delete a field configuration that is not associated with a Field Configuration Scheme.

Copying a Field Configuration

1. On the 'View Field Configuration' page, click the '**Copy**' link next to the field configuration you wish to copy.
2. This will bring you to the 'Copy Field Configuration' page.
3. Enter the name and description of the new field configuration.
4. Click the '**Copy**' button.
5. You will now be directed back the 'View Field Configuration' page, with your new field configuration added to the list. The field settings on the original and the new field configurations will be identical.



A newly created Field Configuration will not take effect until it has been activated (see below).

Activating a Field Configuration

To activate a Field Configuration:

1. configure a Field Configuration Scheme to associate the Field Configuration with appropriate issue types;
2. then associate the Field Configuration Scheme with a project.

For details of both procedures, see 'Associating field behaviour with Issue Types'.

Associating Field Behaviour with Issue Types

On this page:

- What is a 'Field Configuration Scheme'?
- Field Configuration Schemes
- Adding a Field Configuration Scheme
- Configure a Field Configuration Scheme
  - Associating a Field Configuration with an Issue Type
  - Removing an association between a Field Configuration and an Issue Type
  - Editing an association between a Field Configuration and an Issue Type
- Editing a Field Configuration Scheme
- Deleting a Field Configuration Scheme
- Copying a Field Configuration Scheme
- Associating a Field Configuration Scheme with a Project

What is a 'Field Configuration Scheme'?

A **Field Configuration Scheme** maps Field Configurations to issue types. A Field Configuration Scheme can be associated with one or more projects.

This means that you can define Field Configurations for a particular issue type of a given project. For example, it is possible to have a separate field configuration for the 'Bug' issue type and the 'Improvement' issue type for the 'Test' Project.

Because field configuration schemes can be associated with more than one project, your administrative workload is minimised as you can reuse

the same field configuration for issue type mappings across multiple projects.

Field Configuration Schemes

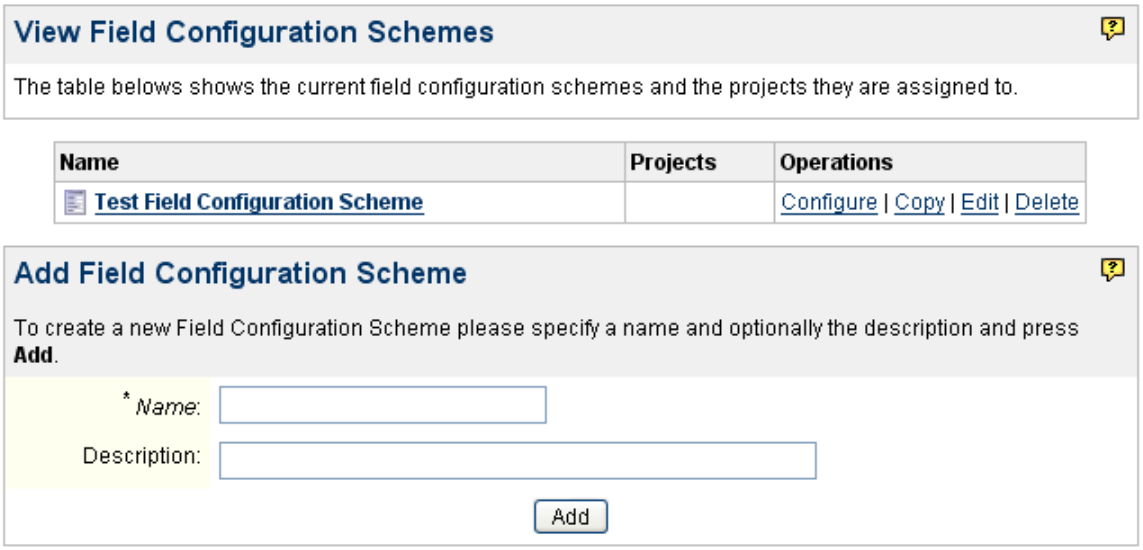
Scheme operations are available from the View Field Configuration Scheme page. To reach this page follow these steps:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**' sub-menu on the left hand side if it is not open already, and choose '**Field Configuration Schemes**' from the list.
4. You will be directed to the page 'View Field Configuration Schemes' with a list of all the Field Configuration Schemes currently configured. From this page you can:
    - Add — create a new Field Configuration Scheme
    - Configure — add or remove associations between issue types[1] and Field Configurations[2].
    - Edit — edit the name and description of the Field Configuration Scheme.
    - Delete — remove a Field Configuration Scheme
    - Copy — create a new Field Configuration Scheme with the same details as an existing one

[1] To create and edit Field Configurations, please see 'Configuring field behaviour'.
[2] To create and edit Issue Types, please see 'Defining 'Issue Type' field values'.

Adding a Field Configuration Scheme

1. The 'Add New Field Configuration Scheme' form is located at the bottom of the 'View Field Configuration Scheme' page (described above).
2. On this form enter the name of the new scheme.
3. You can optionally add a description to the scheme for better identification.
4. Click the '**Add**' button. The Field Configuration Scheme list will be updated automatically with the new scheme.

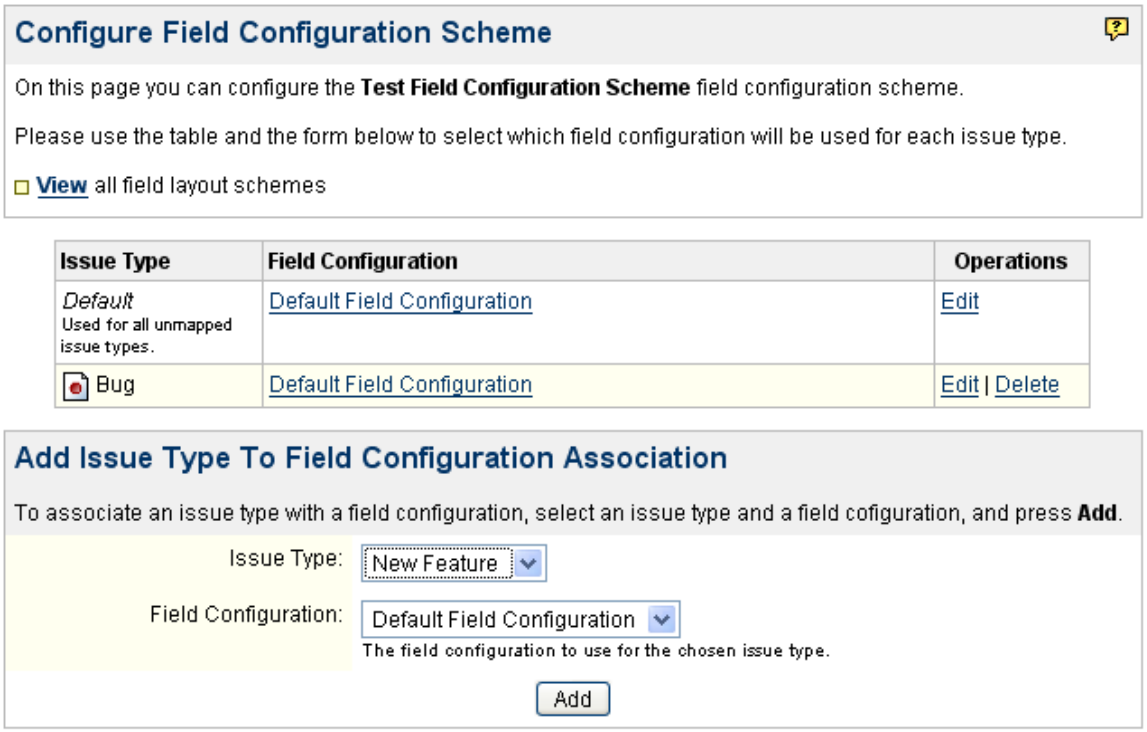


Configure a Field Configuration Scheme

With this operation you can associate, un-associate or change an association between a Field Configuration and a particular issue type. To do this please follow these steps:

1. Click the '**Configure**' link in the same row as the Field Configuration Scheme you wish to configure.
2. The 'Configure Field Configuration Scheme' page will appear, showing the scheme's current mappings of Field Configurations to issue types.
3. The operations available when configuring a field configuration scheme are:
    - Associate an issue type to a field configuration
    - Remove an association between an issue type and a field configuration
    - Edit an association between an issue type and a field configuration

If you have not added any field configurations you will only have the Default Field Configuration to work with.



Associating a Field Configuration with an Issue Type

To associate an issue type with a field configuration:

1. Select the issue type you wish to associate.
2. Select the field configuration you wish to associate with this issue type
3. Click the '**Add**' button. The table will above will be automatically updated with the the issue type in the left most column and the associated field configuration in the central column.

An issue type can only have one association within a given configuration scheme.

If an issue type does not have an association in the scheme, the field configuration associated with the *Default* entry in the scheme will be used for issues of that type.

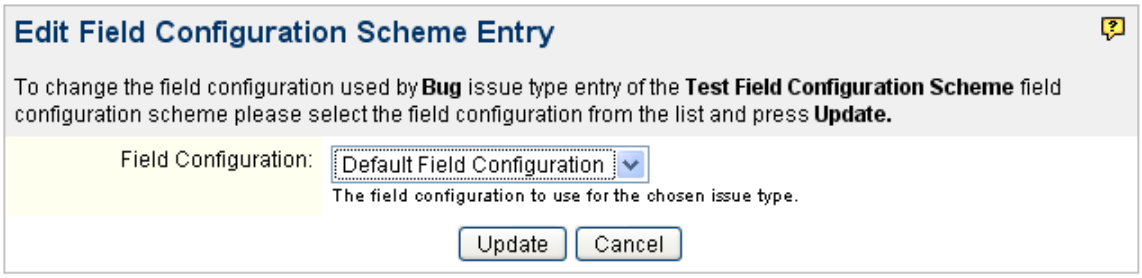
Removing an association between a Field Configuration and an Issue Type

1. Click the '**Remove**' link next to the issue.
2. The issue type association will automatically be removed from the field configuration scheme.

The *Default* entry cannot be removed the scheme.

Editing an association between a Field Configuration and an Issue Type

1. Click the '**Edit**' link next to the issue.
2. Select the new field configuration you would like to associate with this issue type.
3. Click the '**Update**' button.
4. The Issue Type will now be associated with the new field configuration.



Editing a Field Configuration Scheme

To change the name or description of a Field Configuration Scheme:

1. Click the '**Edit**' link next to the desired field configuration scheme. You will now see the 'Edit Field Configuration Scheme' page.
2. Change the name and/or description as necessary.
3. Click the '**Update**' button.

Deleting a Field Configuration Scheme

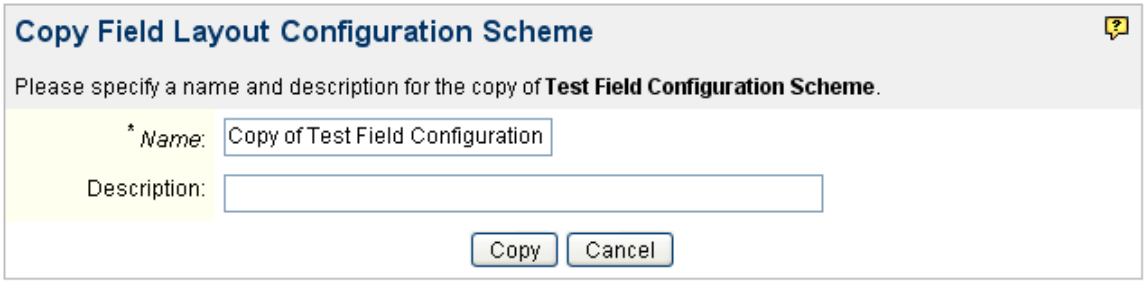
To delete a Field Configuration Scheme:

1. Click the '**Delete**' link next to the desired field configuration scheme. You will now see the 'Delete Field Configuration Scheme' page.
2. Confirm that you would like to delete the scheme by clicking the '**Delete**' button.

Copying a Field Configuration Scheme

To copy a Field Configuration Scheme:

1. Click the '**Copy**' link next to the field configuration scheme you wish to copy. This will bring you to the 'Copy Field Configuration Scheme' page.
2. Enter the name and description of the new field configuration scheme.
3. Click the '**Copy**' button.
4. You will now be directed back the View Field Configuration Scheme page with your new scheme added. The new scheme will have the same configuration as the copied scheme.

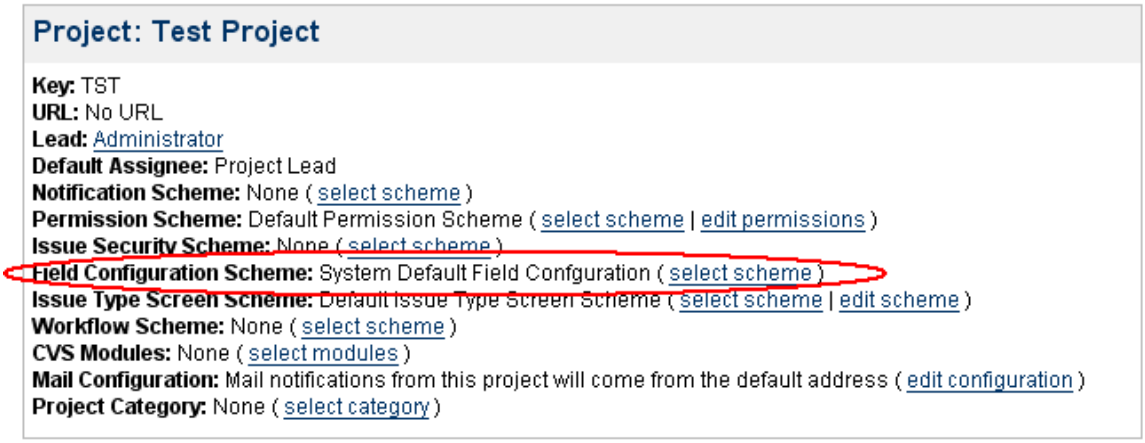


Associating a Field Configuration Scheme with a Project

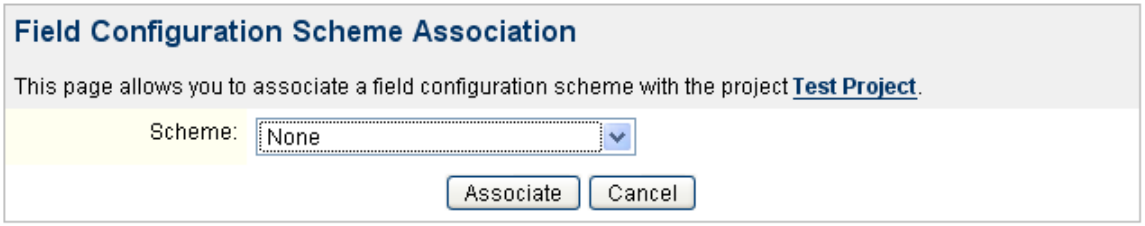
To activate a Field Configuration Scheme, you need to associate it with a project. An association means that the Field Configuration Scheme will now be applied to the chosen project. The issues in that project will use the Field Configuration that is mapped to their issue type by the scheme. Note also that you can use Issue Type Schemes to associate issue types with a project.

To associate a Field Configuration Scheme with a project:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the administration page is a list of projects which this user is allowed to manage. Select the project of interest.
4. Click on the "**select scheme**" link beside the Field Configuration Scheme caption.



5. This will bring up a list of all existing field configuration schemes. Select the scheme you want to associate with this project.



> Selecting *None* will make all the issues in the project use the Default Field Configuration.

6. Click the '**Associate**' button. You will be returned to the project administration page , with the project now associated with the selected Field Configuration Scheme.

> Newly created projects are not associated with any Field Configuration Schemes, and hence use the Default Field Configuration for all issues.

Configuring Rich-Text Renderers

On this page:

- Overview
- Renderable Fields
- Renderer Types
  - Default Text Renderer
  - Atlassian Wiki Renderer
  - Atlassian Wiki Renderer Macro Support
- Implications for JIRA operations
  - Bulk Move
  - Bulk Edit
  - Email Notifications
    - HTML Emails
    - Text Emails
  - Excel View
  - RSS/XML View
  - Other
- Configuring Renderers
  - Applying a Renderer to a Field
  - Enabling a Renderer Plugin
    - Renderer Plugins Configuration
    - Macro Plugins Configuration - Atlassian Wiki Renderer

Overview

JIRA *renderers* affect the display of a field's content. Renderers were introduced in JIRA 3.4 to allow a greater range of expression within text-based fields such as the **Description** and **Comment** fields - see 'Renderable Fields' (below) for a full list.

JIRA currently ships with two renderers: the Default Text Renderer, which displays plain text; and the Atlassian Wiki Renderer (utilising the

Confluence wiki engine), which displays rich text (HTML). See 'Renderer Types' (below) for a full list.

Renderers are configured on a per field basis, allowing a flexible combination of plain text and rich text fields. To configure a renderer for a particular field, see ' Configuring field behaviour '. Note that you can configure the same field differently for different projects and issue types — see 'Associating Field Behaviour with Issue Types '

Renderers are implemented as JIRA plugins, meaning that any renderer can be easily added to or removed from use within JIRA. This includes any custom renderers that may be developed. For details see 'Configuring Renderers' (below).

Please read Implications for JIRA operations below before configuring renderers

> Renderers affect the rendering (view) of a field's value. This means that you can migrate to a different renderer without affecting your issue data; only the view will be changed. It also means that if you do not like the way your issues look using the new renderer you can simply switch back with no impact on your issue data.

Renderable Fields

Potentially any field within JIRA can be a renderable field, but this only really makes sense in the case of text-based fields (since a date field would look nonsensical in wiki-markup). The following table shows the JIRA fields that are renderable out-of-the-box:

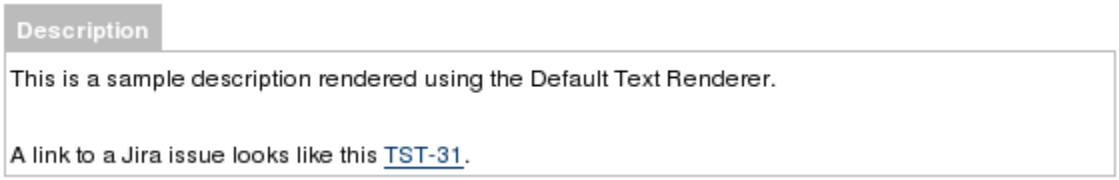
Field Type	Description
Description	The description field of an Issue can have a renderer applied.
Comment	The comments field of an Issue can have a renderer applied.
Environment	The environment field of an Issue can have a renderer applied.
Custom Field - Free Text Field (unlimited text)	Any instance of this type of custom field can have a renderer applied.
Custom Field - Text Field	Any instance of this type of custom field can have a renderer applied.

Renderer Types

JIRA version 3.4 and later ships with two renderers, the Atlassian Wiki Renderer and the Default Text Renderer.

Default Text Renderer

The Default Text Renderer, as the name implies, is the default renderer for JIRA. Out of the box, JIRA is configured to use the text renderer for all renderable fields. The text renderer renders a field's content as plain text, with the following additional auto-linking feature: if the text contains text that resolves to a JIRA issue key then an HTML link will be generated that points to that issue. Below is a sample of how some description text looks when rendered through the Default Text Renderer.



> It is not possible to disable the Default Text Renderer plugin as it is required for the system to function properly. If a field is setup to use a renderer that is later disabled, the field will revert to using the Default Text Renderer.

Atlassian Wiki Renderer

The Atlassian Wiki Renderer allows a user to enter wiki markup to produce html content, as described in 'Editing Rich-Text Fields' in the *JIRA User's Guide.*

This renderer uses the Confluence wiki renderer engine and therefore uses the Confluence wiki notation. The Confluence notation is easy to learn and allows for:

- Italic, bold and underlined text.
- Multiple levels of headings to organise your document.
- Bullets, numbering, tables and quotations.

- Images, screenshots, and emoticons.
- Powerful mini-applications using macros.
  A full notation guide can be found here.

> The Atlassian Wiki Renderer can only be used with JDK 1.4 and up. The renderer will not run on JDK 1.3.

Atlassian Wiki Renderer Macro Support

The Atlassian Wiki Renderer supports pluggable macros in the same way that Confluence does. Macros provide an easy and powerful extension point to the wiki markup language. JIRA ships with a number of macros as described in the *JIRA User's Guide.*

> JIRA and Confluence can share macros, but keep in mind that many Confluence macros are very specific to the Confluence application and will therefore not run within JIRA. For example, the Children macro in Confluence shows links to all of a Page's child pages. JIRA has no concept of 'page' and therefore this macro will not function in JIRA.

Implications for JIRA operations

The fact that JIRA allows you to configure different renderers across different projects/issue types for the same field has implications for bulk operations. Also, since the Atlassian Wiki Renderer inherently creates HTML as its end product, there are implications as to how this will behave when issue data is viewed outside JIRA's web front-end.

Bulk Move

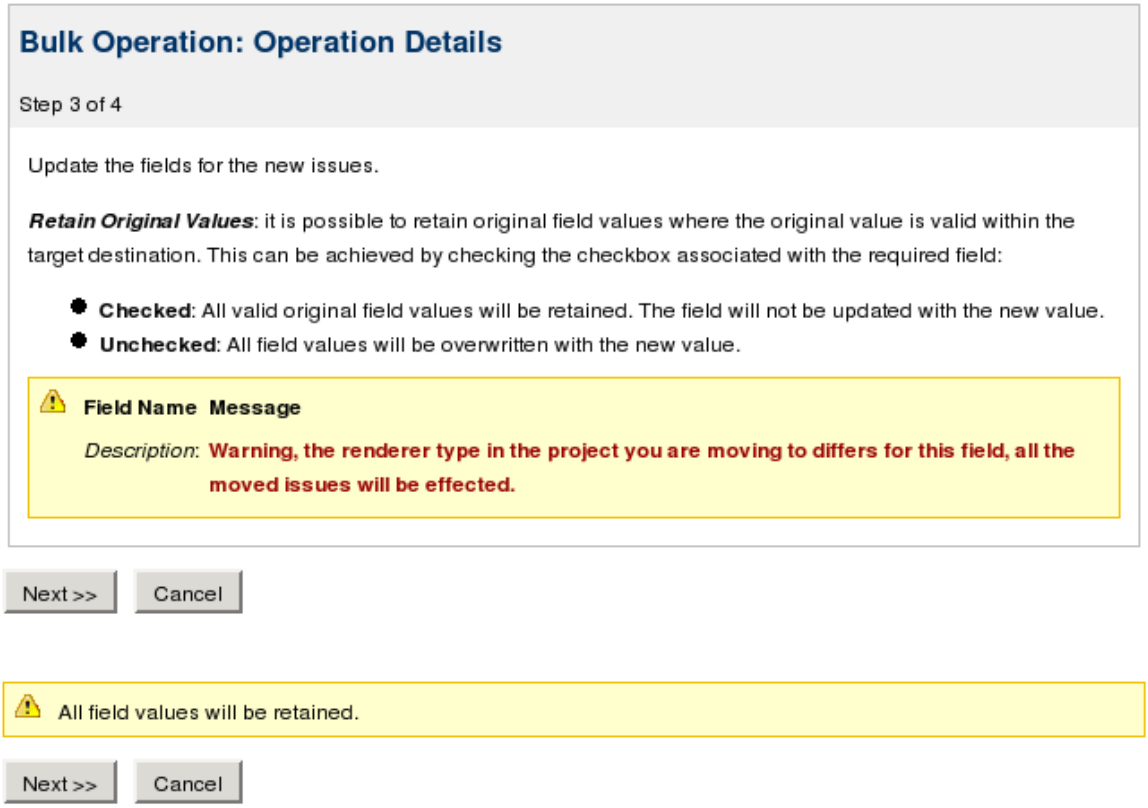
When performing a bulk move operation you can either move issues to an environment (project/issue type) where the renderer types for the fields are the same or where they will be different. If the renderer types for where you are moving to are the same then you will not notice any changes to the way the issues data is displayed once the move has occurred and the move operation will not prompt the user with any warnings.

When bulk moving issues to an environment (project/issue type) that has a different renderer type defined for one of the fields being affected by the move, if any of the issues have a non empty value associated with the field, the move operation will present the user with a warning so that you can be aware of the change. The warning does not affect the move operation in any way but it is there to alert you to the fact that the moved issues' affected fields may look different in their new project/issue type.

This is best illustrated with an example. Let's say you have project 'A' which is configured to use the Atlassian Wiki Renderer for the Description field. Let's say you also have a project 'B' which is configured to use the Default Text Renderer for the Description field. You have three issues that exist in project 'A' and you want to perform a bulk move of the three issues to project 'B'. If none of the issues in project 'A' have a value set for the Description field they will be moved and you will not notice any changes since there is no value to render. If one of the issues has the following value in its Description:

```
{color:green}green text{color}

                    *this is a test issue*
```

You would be presented with this screen in the bulk move to alert you that you are changing renderers as a result of the move:



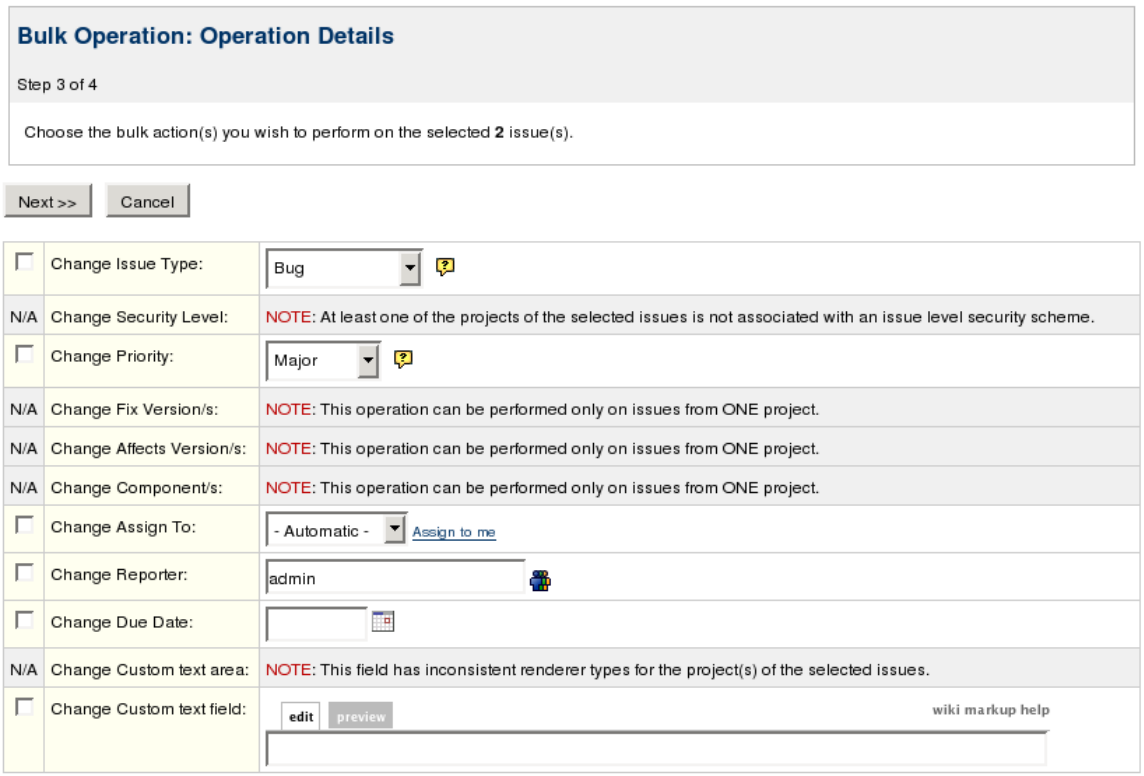
The move operation does nothing to affect the data itself so after the move the wiki markup will display through the Default Text Renderer. In our example the before and after look like this:

Bulk Edit

When performing a bulk edit operation the only renderable fields you may be able to bulk edit are instances of the Text Field, and Free Text Field (unlimited text) custom fields. The bulk edit operation does not allow you to bulk edit the description, environment, or comment fields.

You will only be allowed to bulk edit a renderable field if all the issues selected for edit use the same renderer type. If the renderer type differs for any of the selected issues you will be presented with an error message.

This is best illustrated with an example. Let's say you have two global custom fields, 'Custom text area' and 'Custom text field', whose types are as their names imply. Let's say you have project 'A' which is configured to use the Atlassian Wiki Renderer for both of the fields. Lets say you also have a project 'B' which is configured to use the Default Text Renderer for the 'Custom text area' field and the Atlassian Wiki Renderer for the 'Custom text field'. Let's also say that you have one issue in each project. If you were to perform a bulk edit operation on the two issues in these projects you will be presented with the screenshot below:



You will notice that for the 'Custom text area' field you are presented with an warning that the field has inconsistent renderer types and that it is not available to be selected for bulk edit. This is because the fields do not share the same renderer in the two issues. You will also notice that for the 'Custom text field' field you are presented with an editable input that allows for wiki preview. This is because the field shares the same renderer in the two issues.

Email Notifications

JIRA allows for extensive configuration in relation to email notifications. JIRA can be send out two types of emails, HTML and text (see 'Email Formatting').

HTML Emails

When using the Atlassian Wiki Renderer, the rendered content (i.e. exactly what you see on the 'View Issue' page) will be sent out in the emails. This will create emails which are as rich as the content makes it. If using the Atlassian Wiki Renderer this is the preferred type of email since it is a real representation of the wiki markup.

Text Emails

When using the Atlassian Wiki Renderer, the actual wiki markup (unrendered) will be displayed in text emails for fields that use the Atlassian Wiki Renderer. This is obviously less readable than the rendered version of the markup, but because the markup's syntax is quite simple the text does remain easy to read.

Excel View

JIRA allows the Issue Navigator view to be exported to an Excel spreadsheet. If any of the fields being exported to Excel are using the Atlassian Wiki Renderer, the value exported to the cell in Excel will be the original wiki markup. Attempting to display complex HTML within a cell in Excel adds rows and columns that make using the data for formulas very difficult.

> The unrendered wiki markup will be shown in Excel cells for fields that use the wiki renderer.

RSS/XML View

JIRA allows the Issue Navigator view to be exported to RSS/XML. If a field is using the Default Text Renderer its values will be exported in a CDATA section within the generated XML. If a field is using the Atlassian Wiki Renderer, its rendered value will be XML escaped and included in the generated XML. If the XML view is being used as an RSS feed, most RSS readers will render the generated HTML so you will see the rich content within your RSS reader.

If you would like to have this view feed out the raw values (unrendered) then you can send an additional request parameter 'rssMode=raw'. If the

original link looks like this:

```
http://localhost:8080/browse/AAA-1?decorator=none&view=rss
```

Then the URL to have the raw values placed inside a CDATA should look like this:

```
http://localhost:8080/browse/AAA-1?decorator=none&view=rss&rssMode=raw
```

Other

This section describes other issues to be aware of in relation to the renderers.

- When editing a renderable custom field's default value, even if it is only ever configured to use the Atlassian Wiki Renderer you will not be presented with the 'Edit' and 'Preview' tabs. Unfortunately it is not possible, in that context, to tell which renderer should be used for editing. This said, if you enter a default value using wiki markup then this will render correctly in environments (project/issue type) where the field has been configured to use the Atlassian Wiki Renderer.

Configuring Renderers

Applying a Renderer to a Field

To enable a renderer for a particular field, edit the Field Configuration and choose the appropriate renderer for the field. For details, see Configuring field behaviour.

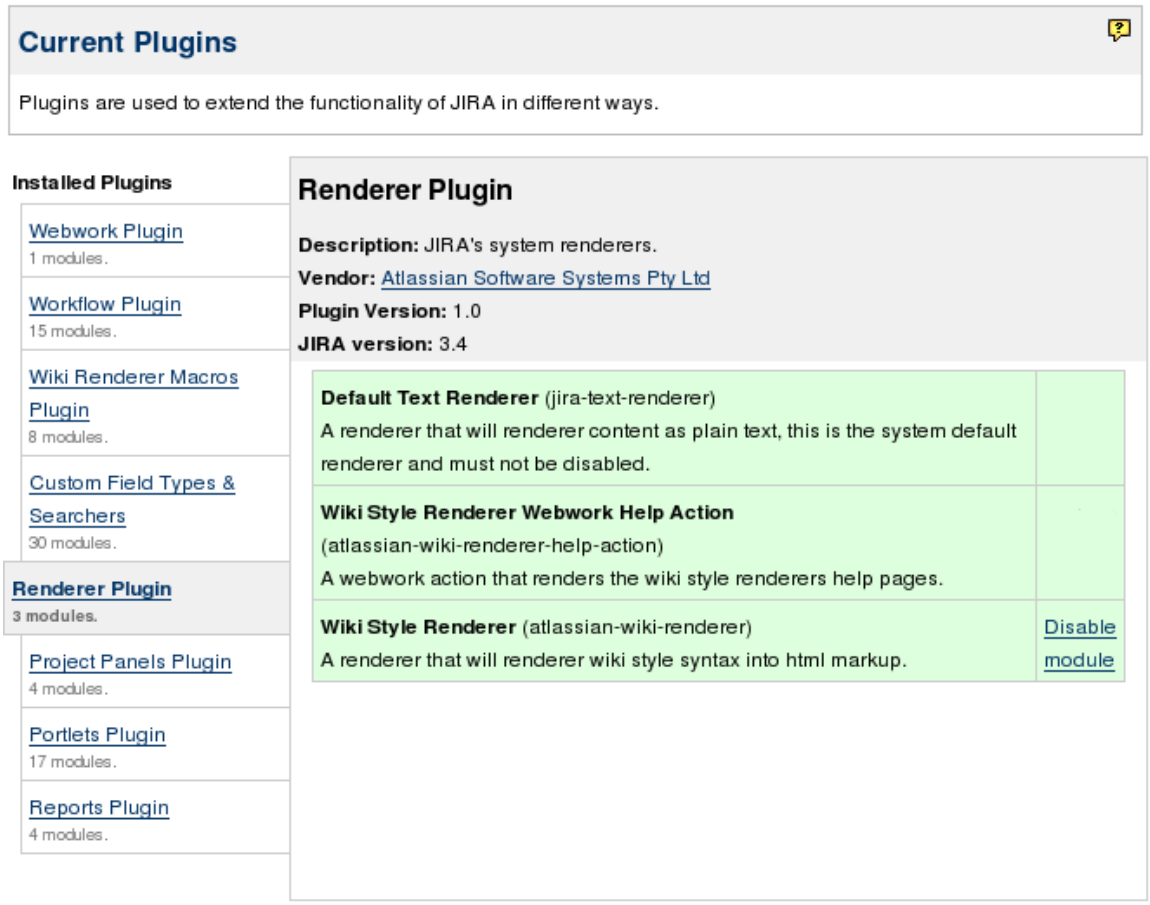
Enabling a Renderer Plugin

Renderers within JIRA are implemented as JIRA plugins. The macros that the Atlassian wiki renderer uses are also implemented as JIRA plugins. For general information on plugins please see this guide.

Plugins are configured at a site-wide level - it is not possible to configure plugins at a project/issue type level.

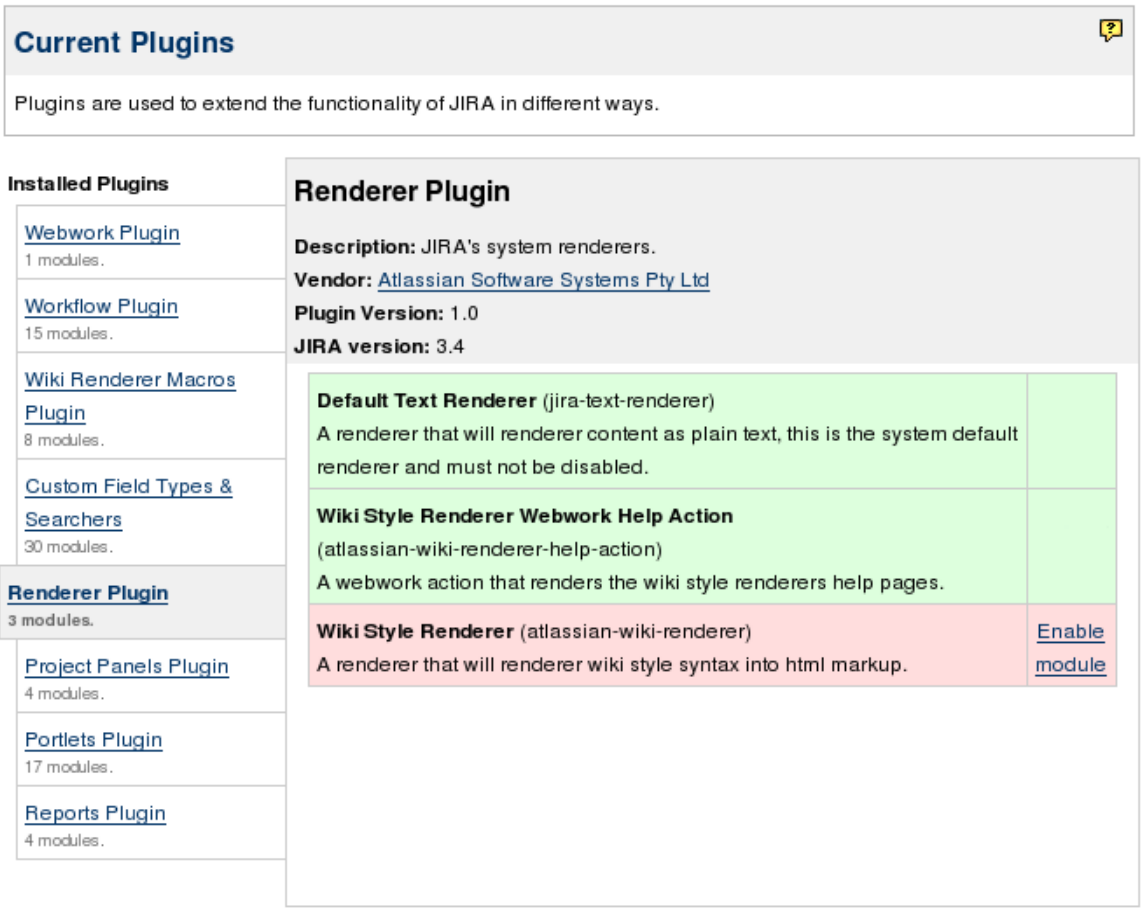
Renderer Plugins Configuration

Renderers and their dependant components, except for the default text renderer, can be enabled/disabled via the plugin administration menus. If you navigate, as an administrator, to 'Administration' > 'Plugins' and then click on the option 'Renderer Plugin' you will see the following screen.



The plugin titled 'Wiki Style Renderer Webwork Help Action' is a front-end helper for showing the Atlassian wiki renderer notation guide and it can not be disabled.

From this screen you will see all the configured Renderers within JIRA. At the moment only two renderers exist but if more are created you will see there configuration here. If you click on the 'Disable Module' link for the 'Wiki Style Renderer' this will deactivate the renderer for the entire instance of JIRA.

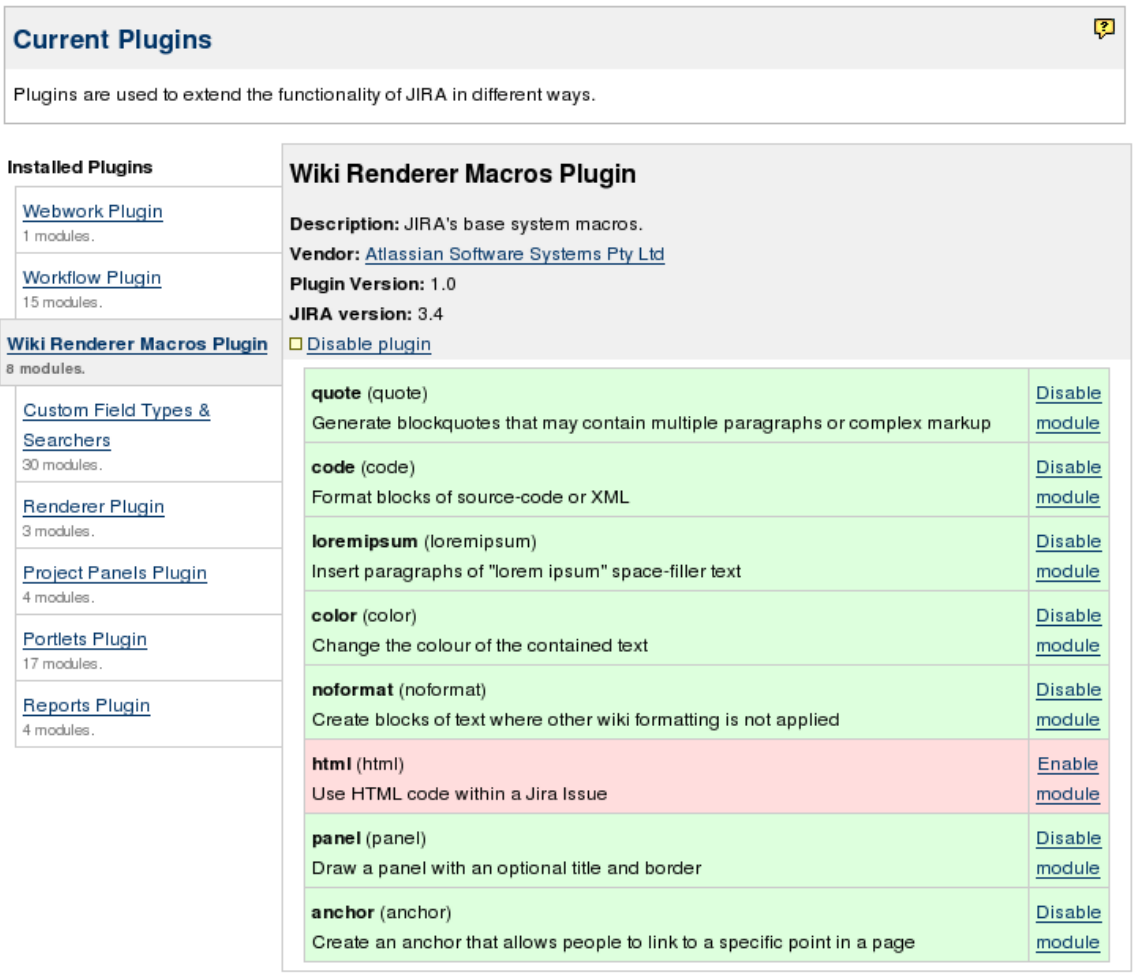


Any fields that are still setup to use the disabled renderer will fall back to the default text renderer and when you attempt to edit the field a warning message will alert you to the fact that you are configured to use a renderer that is not available.

When a renderer is disabled it will not be available for selection when changing a fields renderer. To enable the renderer just click the 'Enable Module' link. Enabling/Disabling a renderer has no effect on the renderer settings in the field configurations so it is possible to disable and then re-enable a renderer without effecting any data.

Macro Plugins Configuration - Atlassian Wiki Renderer

The macros used by the Atlassian wiki renderer can be enabled/disabled via the plugin administration menus. If you navigate, as an administrator, to 'Administration' > 'Plugins' and then click on the option 'Wiki Renderer Macros Plugin' you will see the following screen.



From this screen you will see all the configured macros within JIRA. If a macro is disabled then it will not be available to the wiki renderer, likewise a macro must be enabled for it to be available to the wiki renderer. If you deploy any additional macros that you wish to use, they must be enabled here to be available to the wiki renderer. For more information on writing plugins please see this guide.

Defining a Screen

On this page:

- What is a 'Screen'?
- Configuring a Screen's Fields
  - Adding a Field to Screen
  - Removing a Field from a Screen
  - Reordering Fields on a Screen
- Adding a Screen
- Editing a Screen's Details
- Deleting a Screen
- Copying a Screen
- Configuring Tabs
  - Adding a Tab
  - Moving fields between Tabs
  - Navigating between Tabs
  - Deleting a Tab
  - Renaming a Tab
  - Reordering Tabs
- Activating a Screen

What is a 'Screen'?

Screens group multiple issue fields. Using Screens, you can control which fields are displayed, and the fields' vertical display order, during issue operations (e.g. 'Create Issue' and 'Edit Issue') or workflow transitions (e.g. 'Resolve Issue'). You can also split fields on a Screen into multiple tabs.

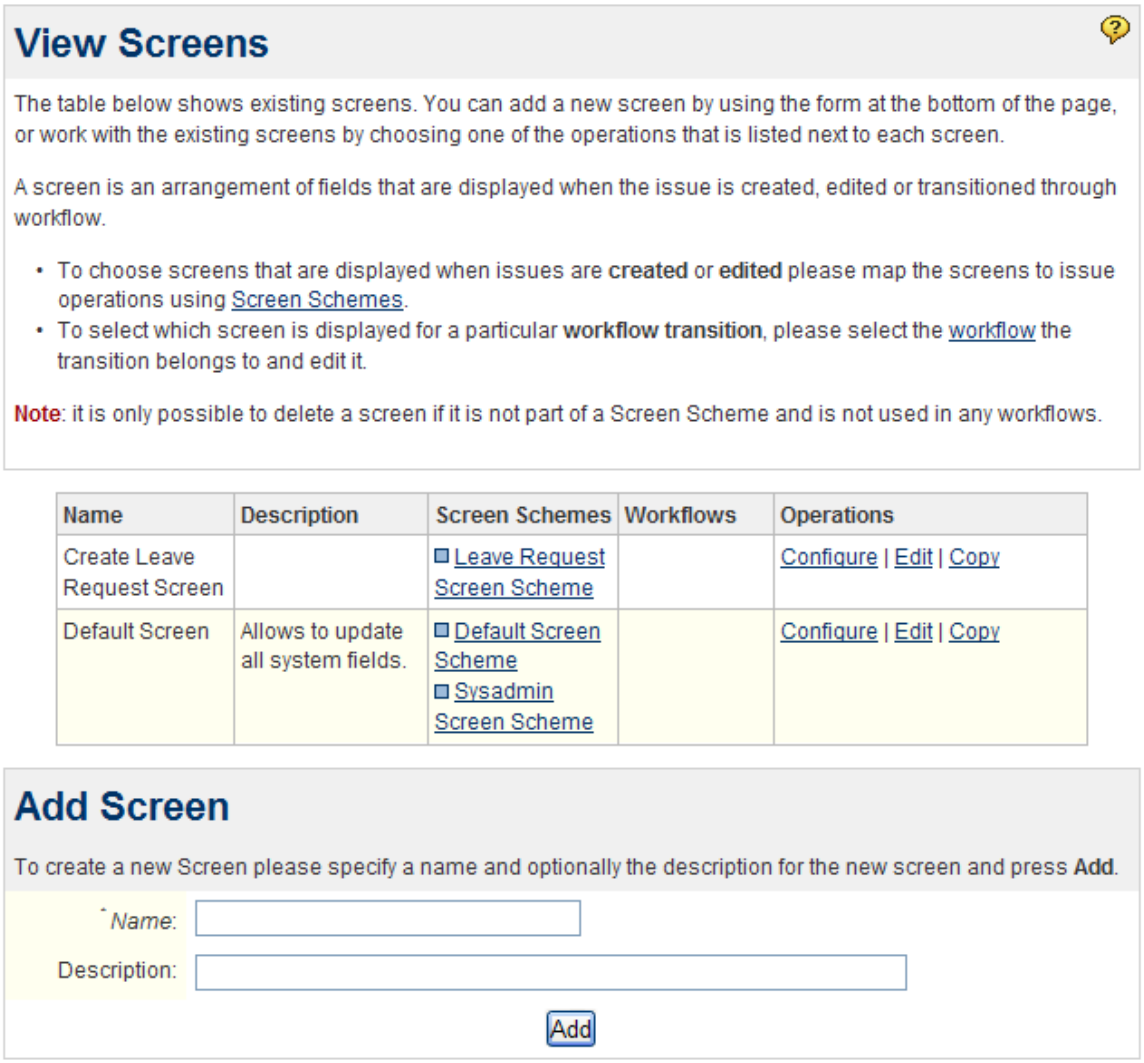
Screens overlap slightly with Field Configurations in regards to field visibility. Note that when a Screen is displayed to a user, for example, during issue creation, the user will see only the issue fields that:

1. the user has permissions to edit (e.g. the 'Due Date' field can only be edited by users with the 'Schedule Issues' permission).
2. are present on the Screen that is associated with the 'Create Issue' operation for this issue.
3. are not hidden in the Field Configuration applicable to the issue.

A field may be present on a Screen, but if it is hidden in an appropriate Field Configuration, it will not be visible to the user when the Screen is displayed. Note also that, if a particular field needs to be hidden at all times, it is simpler to hide the field in an applicable Field Configuration rather than remove it from all Screens. For more information please see the Overview.

Configuring a Screen's Fields

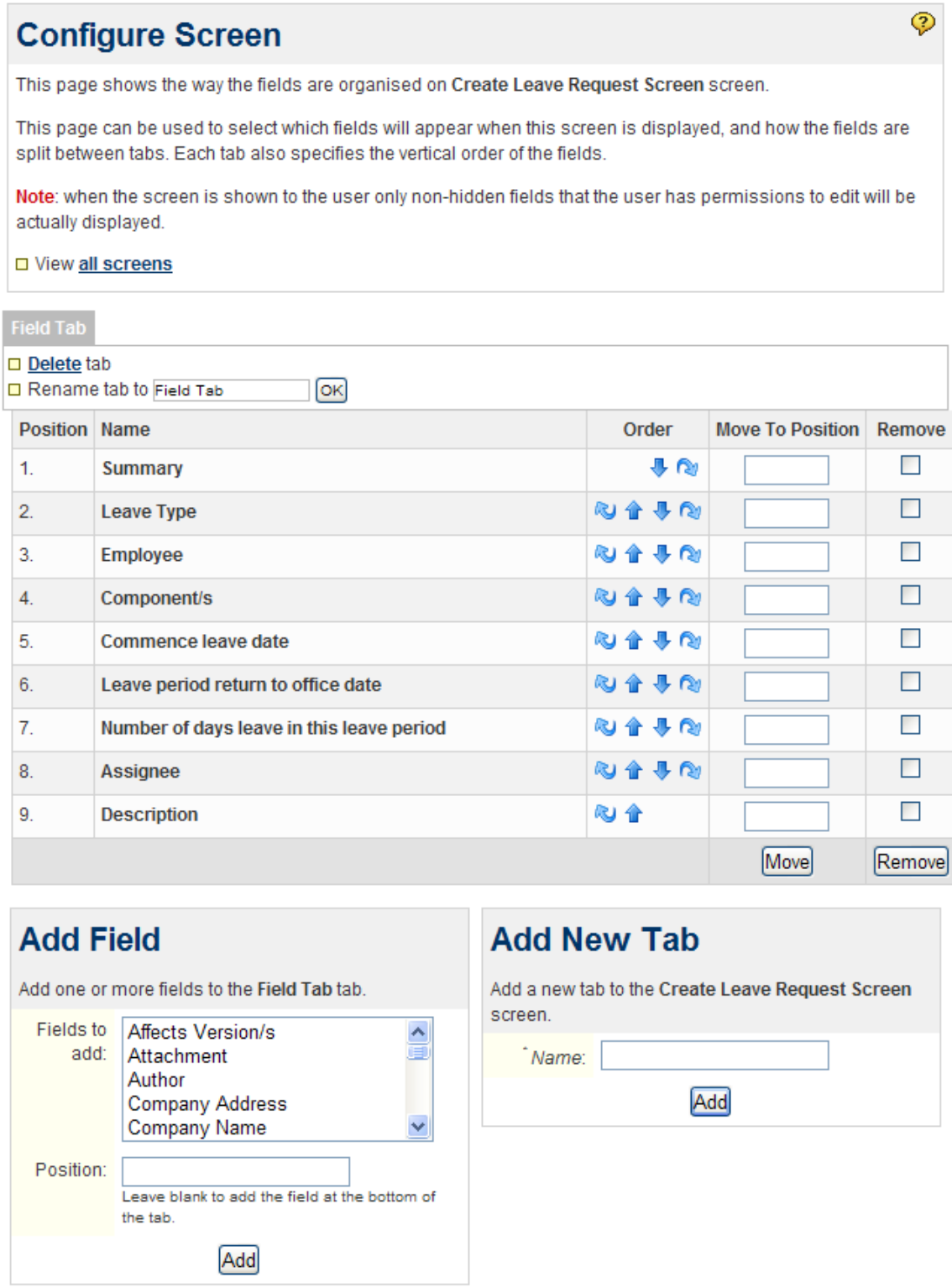
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**' sub-menu in the left-hand side column, and choose '**Screens**' from the list.
4. You will then be directed to the "View Screens" page:



From this page you can:

- Add a new screen.
- Configure a screen's fields
- Edit a screen's name and/or description
- Delete a screen.
- Copy a screen.

5. Click the '**Configure**' link next to the Screen of interest.
6. You will now see the "Configure Screen" page. (Note: the screenshot below is common between *Standard* and *Professional* editions; the *Enterprise* edition page includes the tabs functionality — see below).



Adding a Field to Screen

1. The "Add Field" form is located at the bottom of the "Configure Screen" page.
2. Select the field/s that you wish to add to the screen.
3. You can also specify the position in which the field will be placed.

> If you have selected multiple fields and specified a position, the topmost field selected will be placed in the corresponding position and the other fields directly below it.

4. Click the '**Add**' button.

Removing a Field from a Screen

1. From the "Configure Screen" page, select the checkboxes next the the fields you wish to remove.
2. Click the '**Remove**' button located at the bottom of the table.
3. The fields will be removed from the Screen and will become available in the "Add Field" form at the bottom of the screen.

Reordering Fields on a Screen

To change the vertical display order of fields:

1. In the text box in the "Move to Position" column next to the desired field, specify the position you wish to move the field to.
2. You can repeat this for multiple fields specifying a different position for each field.
3. Click the '**Move**' button located at the bottom of the table in the "Move to Position" column.
4. All the fields will be placed in the specified positions.

> Alternatively, you can click on the arrows next to the desired field to move the field up, down, to the first position or to the last position.

Adding a Screen

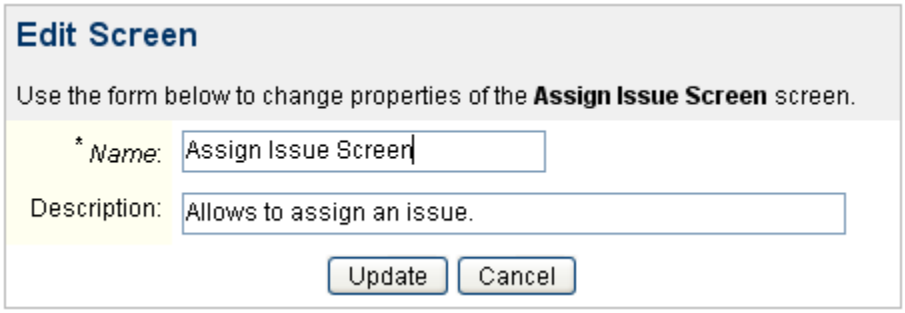
1. The "Add Screen" form is located at the bottom of the "View Screens" page (see 'Configuring a Screen's Fields', above).
2. Enter the name of the new Screen.
3. You can optionally add a Description.
4. Click the '**Add**' button. The page will automatically update the Screen list with the new Screen.

> A newly created Screen is not usable until it has been associated with either an issue operation (via a Screen Scheme) or a workflow transition. See ' Activating Screens ' (below).

Editing a Screen's Details

To change Screen's name and/or description:

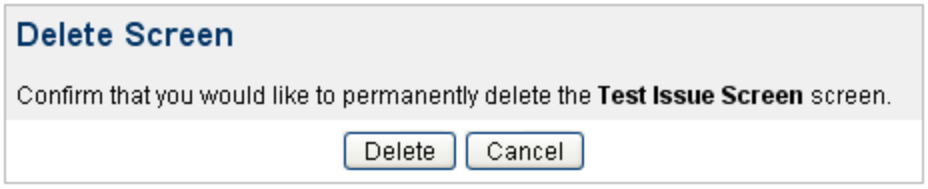
1. On the "View Screens" page (see 'Configuring a Screen's Fields', above), click the '**Edit**' link next to the appropriate screen.
2. You will now be directed to the "Edit Screen" page where you can edit the name and/or description of the Screen.
3. Click the '**Update**' button. You will be brought back to the "View Screens" page with your updates now applied to the Screen.



Deleting a Screen

To entirely remove a Screen from the system:

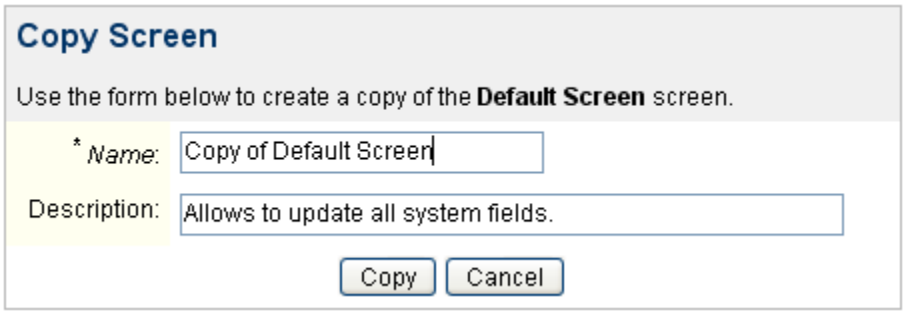
1. On the "View Screens" page (see 'JIRADOC:Configuring a Screen's Fields', above), click the '**Delete**' link next to the Screen you wish to delete.
2. Click the '**Delete**' button to confirm this action. You will be brought back to the "View Screens" page with the Screen removed from the list of Screens.



Screens that are associated with at one or more Screen Schemes, or one or more workflow transitions, cannot be deleted.

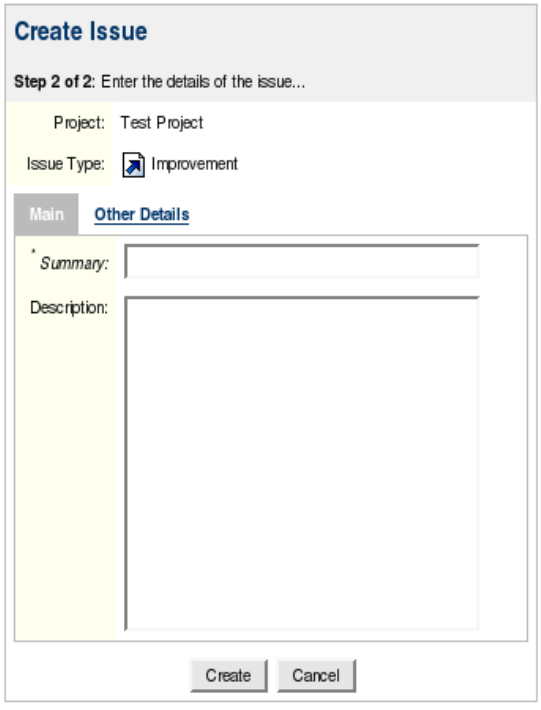
Copying a Screen

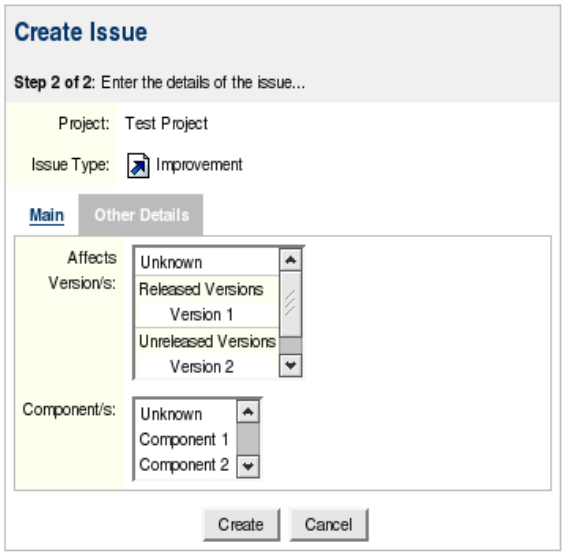
1. On the "View Screens" page (see 'Configuring a Screen's Fields', above), click on the '**Copy**' link next to the Screen you wish to copy. You will be directed to the "Copy Screen" page.
2. Enter a name and a description for the new Screen.
3. Click the '**Copy**' button. You will be brought back to the "View Screens" page, and the newly added Screen will have the same issue fields and field positions as the original field screen.



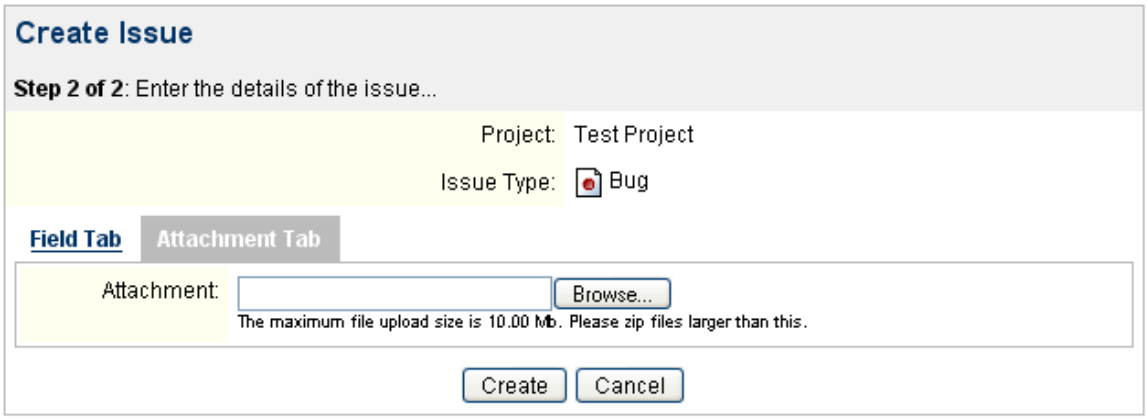
Configuring Tabs

Splitting a Screen into multiple tabs can help to group related fields. For example, the following screenshot shows a simple Screen that only shows the issue 'Summary' and 'Description' on the first tab ('Main'), and 'Affected Versions' and 'Components' on the second tab ('Other Details'):





This functionality is very useful for organising complex Screens, as you can place less used fields, for example, 'Attachment' and 'Environment', onto separate tabs:



Screen tabs are available from the "Configure Screens" page (see ' Configuring a Screen's Fields ', above).

Adding a Tab

1. The '**Add Tab**' is located on the bottom right of the "Configure Screen" page.
2. Enter the name of the new tab on this form.
3. Click the '**Add**' button.

Moving fields between Tabs

1. In the "Move to Tab" column next to the field you wish to move, select the desired tab.
2. Repeat this for all the fields you wish to move.
3. Click the '**Move**' button located at the bottom of the table in the "Move to Tab" column.
4. All the selected fields will be moved to the appropriate tabs.

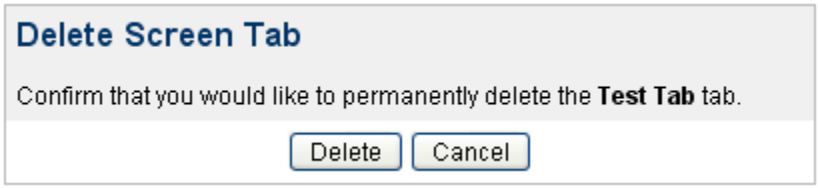
> Please note that the system fields on the default 'View Issue' screen (e.g. Summary, Security Level, Issue Type, etc.) are fixed and cannot be moved onto a separate tab. However, any custom fields that have been added to the 'View Issue' screen can be moved onto a separate tab.
> This restriction only applies to the screen associated with the 'View Issue' operation, i.e. system fields can be moved onto other tabs for screens associated with operations such as 'Create Issue', 'Edit Issue', etc.

Navigating between Tabs

To navigate between the Tabs of a Screen, simply click on the links on the top left of the "Configure Screen" form.

Deleting a Tab

1. Navigate to the Tab you wish to remove.
2. Click the '**Delete**' tab link. You will now be directed to the "Delete Tab" confirmation page.
3. Click the '**Delete**' button to confirm. You will be returned to the "Configure Screen" Page.



Renaming a Tab

1. Navigate to the desired Tab.
2. The '**Rename**' text field is located in the top left of the "Configure Screen Tab" form.
3. Enter the new name of the Tab.
4. Click '**Enter**'.

Reordering Tabs

It is possible to configure the horizontal order of Tabs by clicking on the arrows to move the selected Tab left or right.

Activating a Screen

To make a Screen available to users, you can either:

- Associate the Screen with an **issue operation** (e.g. 'Create Issue'), via a **Screen Scheme** — see 'Associating Screens with Issue Operations', or
- Associate the Screen with a **Workflow Transition** (e.g. 'Resolve Issue') — see 'Configuring Workflow'.

Associating a Screen with an Issue Operation

On this page:

- What is a 'Screen Scheme'?
- Configuring a Screen Scheme
  - Associating an Issue Operation with a Screen
  - Editing an Association
  - Deleting an Association
- Managing Multiple Screen Schemes
  - Adding a Screen Scheme
  - Editing a Screen Scheme's details
  - Deleting a Screen Scheme
  - Copying a Screen Scheme
- Activating a Screen Scheme

What is a 'Screen Scheme'?

A **Screen Scheme** allows you to choose which Screen will be shown to a JIRA user when they perform a particular *issue operation.* There are three issue operations for which you can choose a Screen:

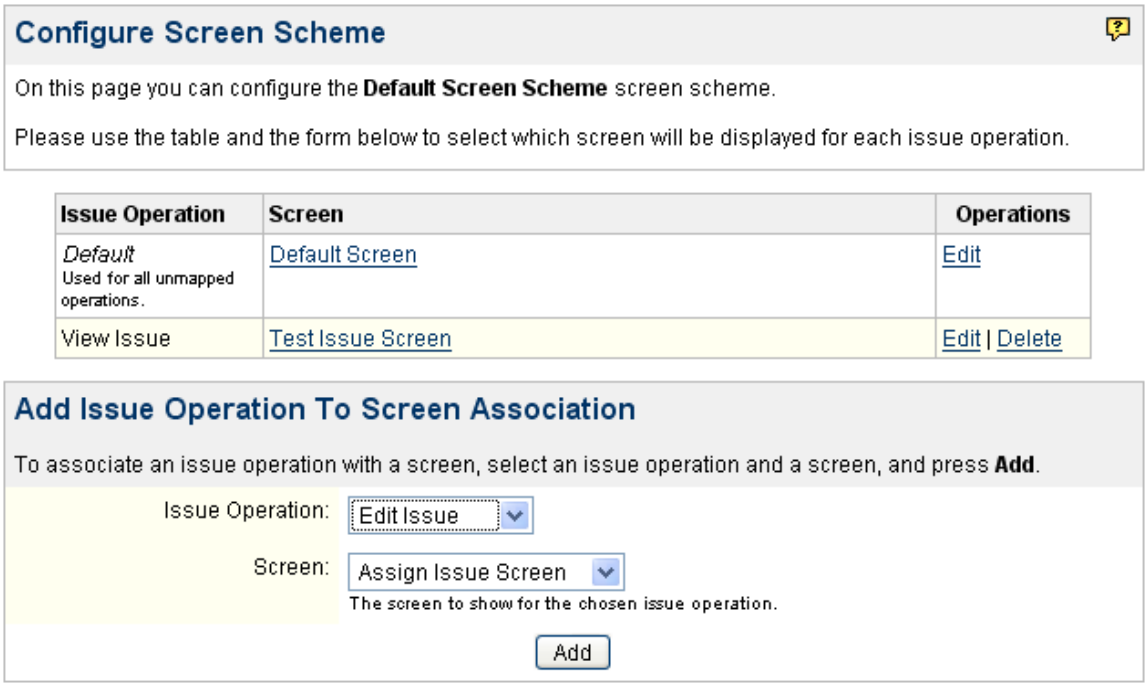
- 'Create Issue' — choose the Screen that is shown when an issue is being created .
- 'Edit Issue' — choose the Screen that is shown when an issue is edited.
- 'View Issue' — choose the Screen that is shown when a user views an issue.

You can specify the same screen for each of these issue operations, or choose different screens for each operation.

Once you have created your Screen Scheme, you will need to activate it by associating it with an Issue Type Screen Scheme.

Configuring a Screen Scheme

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**' sub-menu on the left hand side if it is not open already, and choose 'Screen Schemes' from the list.
4. The 'View Screen Schemes' page will be displayed (see 'Managing Multiple Screen Schemes' below). Locate the Screen Scheme in which you are interested, and click the '**Screens**'
   link next to it.
5. You will be brought to the 'Configure Screen Scheme' page:



Associating an Issue Operation with a Screen

1. The '**Add Issue Operation to Screen Association**' is located at the bottom of the 'Configure Screen Scheme' page (see above).
2. Select the Issue Operation with which you wish to associate a Screen.
3. Select the desired Screen.
4. Click the '**Add**' button and the new association will be added to the list of associations.

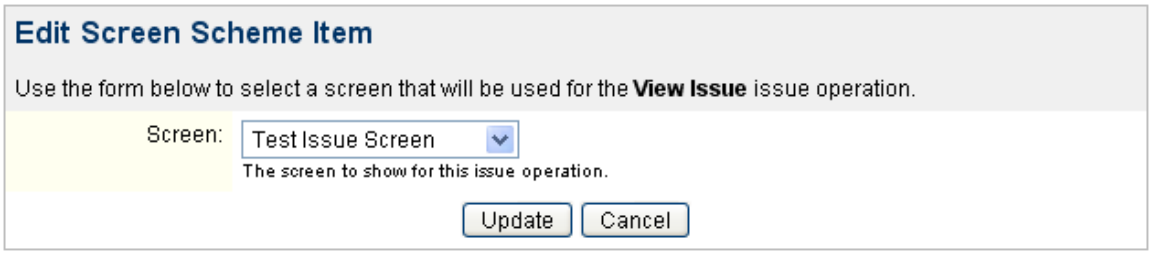
The 'View Issue' operation only allows you to control the layout of **[custom fields]** in the middle of the 'View Issue' page. The 'View Issue' page ignores all the non-custom fields on the Screen.

There can only be one association for an issue operation per Screen Scheme. If all operations have been associated with a Screen, use the 'Edit' link next to each operation to change the Screen it is associated with.

If an issue operation does not have a specific mapping to a Screen, the screen that is associated with the *Default* entry will be used for that operation. The *Default* entry cannot be deleted from a Screen Scheme. You can use the "Edit" link next to the *Default* entry to change the Screen that is associated with it.

Editing an Association

1. On the 'Configure Screen Scheme' page, click the '**Edit**' link next to the issue operation you wish to edit.
2. You will be brought to the "Edit Screen Scheme Item" page.
3. Select the screen you wish to change the association to.
4. Click the '**Update**' button and you will be returned to the screen scheme page.



Deleting an Association

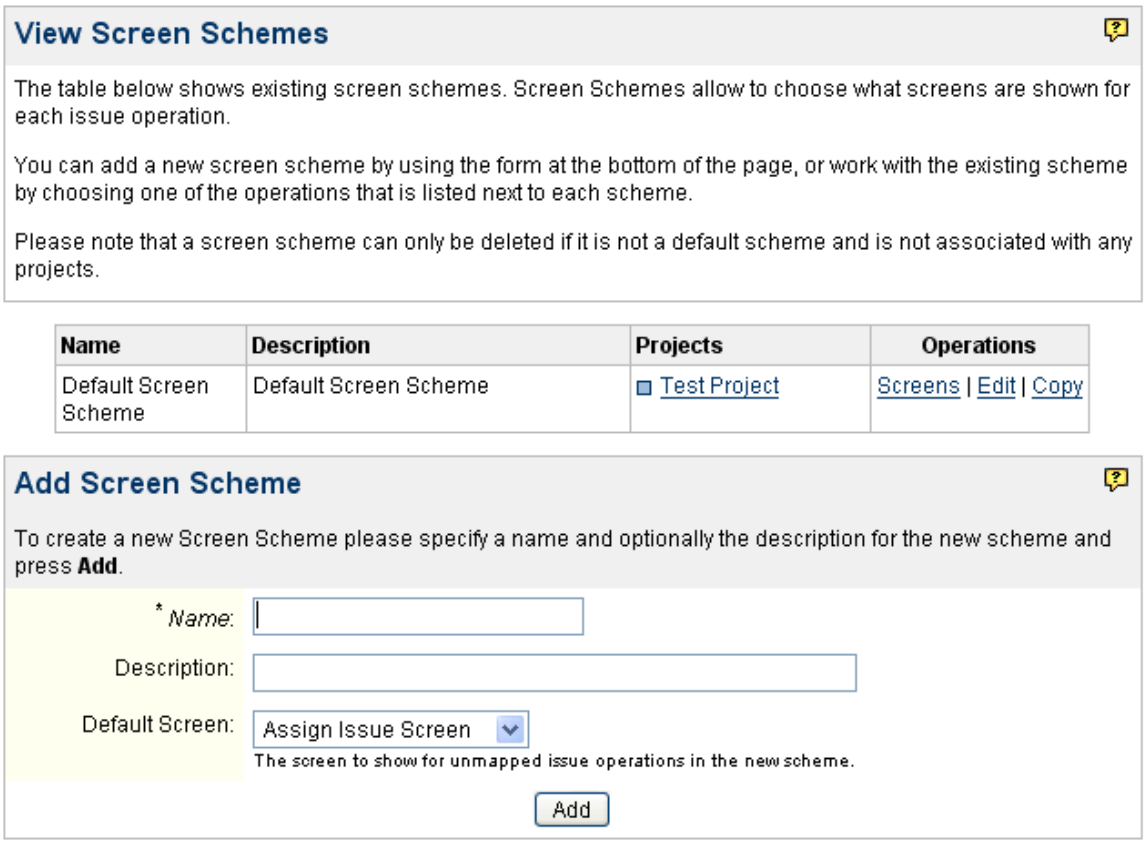
1. On the 'Configure Screen Scheme' page, click the '**Delete**' link next to the issue operation you wish to remove.

2. The association will be automatically removed from the list.

Managing Multiple Screen Schemes

Depending on your requirements, you may want to create multiple Screen Schemes, and associate them with different projects/issue types. To manage multiple Screen Schemes:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**' sub-menu on the left hand side if it is not open already, and choose '**Screen Schemes**' from the list.
4. This will bring you to the 'View Screen Schemes' page.

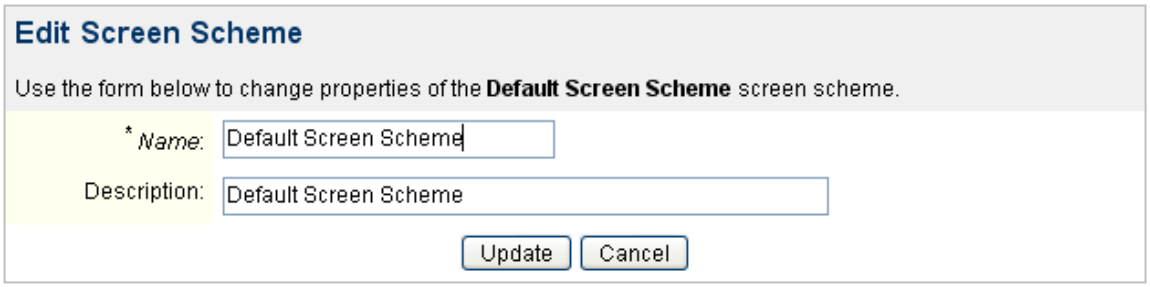


Adding a Screen Scheme

1. The '**Add Screen Scheme**' form is located at the bottom of the 'View Screen Schemes' page.
2. Enter the name of the new field screen.
3. You can optionally add a description of the field screen.
4. Select a default field screen. The default screen will be used for views that do not have an association.
5. Click the '**Add**' button. The screen will automatically update the field screen schemes list with the new field screen scheme.

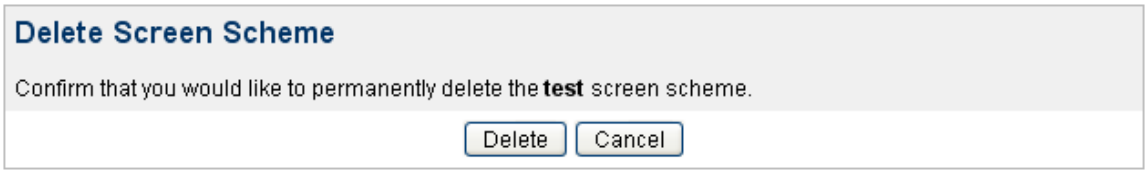
Editing a Screen Scheme's details

1. On the 'View Screen Schemes' page, click the '**Edit**' link next to the selected Screen Scheme.
2. You will now be directed to the "Edit Screen Scheme" page where you can edit the Screen Scheme's name and description and the Screen that is associated with the *Default Entry* of the scheme.
3. Click the '**Update**' button.
4. You will be brought back to the Screen Schemes page with your updates now applied to the Screen Schemes list.



Deleting a Screen Scheme

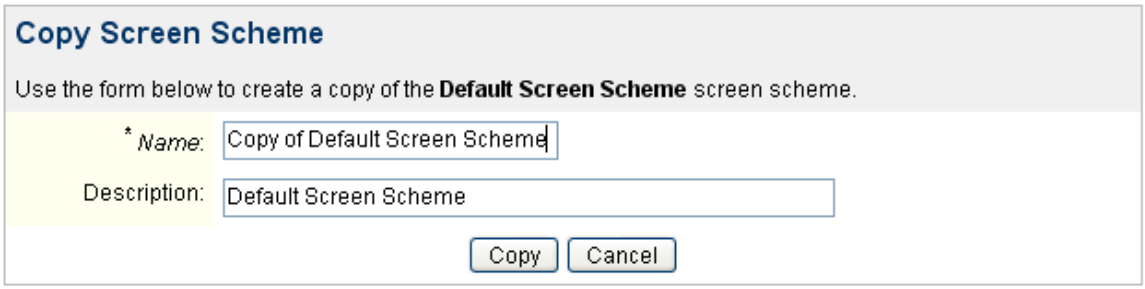
1. On the 'View Screen Schemes' page, click the '**Delete**' link next to the Screen Scheme you wish to delete.
2. Click the '**Delete**' button to confirm this action.
3. You will be brought back to the Screen Schemes page with the Screen Scheme removed from the screen schemes list.



Screen Schemes that are associated with an Issue Type Screen Scheme cannot be deleted.

Copying a Screen Scheme

1. On the 'View Screen Schemes' page, click the '**Copy**' link next to the Screen Screen you wish to copy.
2. You will now be directed to the "Copy Screen Scheme" page.
3. Enter the name and description of the new Screen Scheme.
4. Click the '**Copy**' button.
5. You will be brought back to the Screen Schemes page, and the newly added Screen Scheme will have the same settings as the original Screen Scheme.



Activating a Screen Scheme

To activate a Screen Scheme, you need to associate it with one or more projects and issue types, using Issue Type Screen Schemes. To activate a Screen Scheme,

1. configure an Issue Type Screen Scheme to use the Screen Scheme; then
2. associate the Issue Type Screen Scheme with a project.

For details of both procedures, see 'Associating screens with Issue Types'.

Associating a Screen with an Issue Type

On this page:

- What is an 'Issue Type Screen Scheme'?
- Adding an Issue Type Screen Scheme
- Editing an Issue Type Screen Scheme
- Deleting an Issue Type Screen Scheme

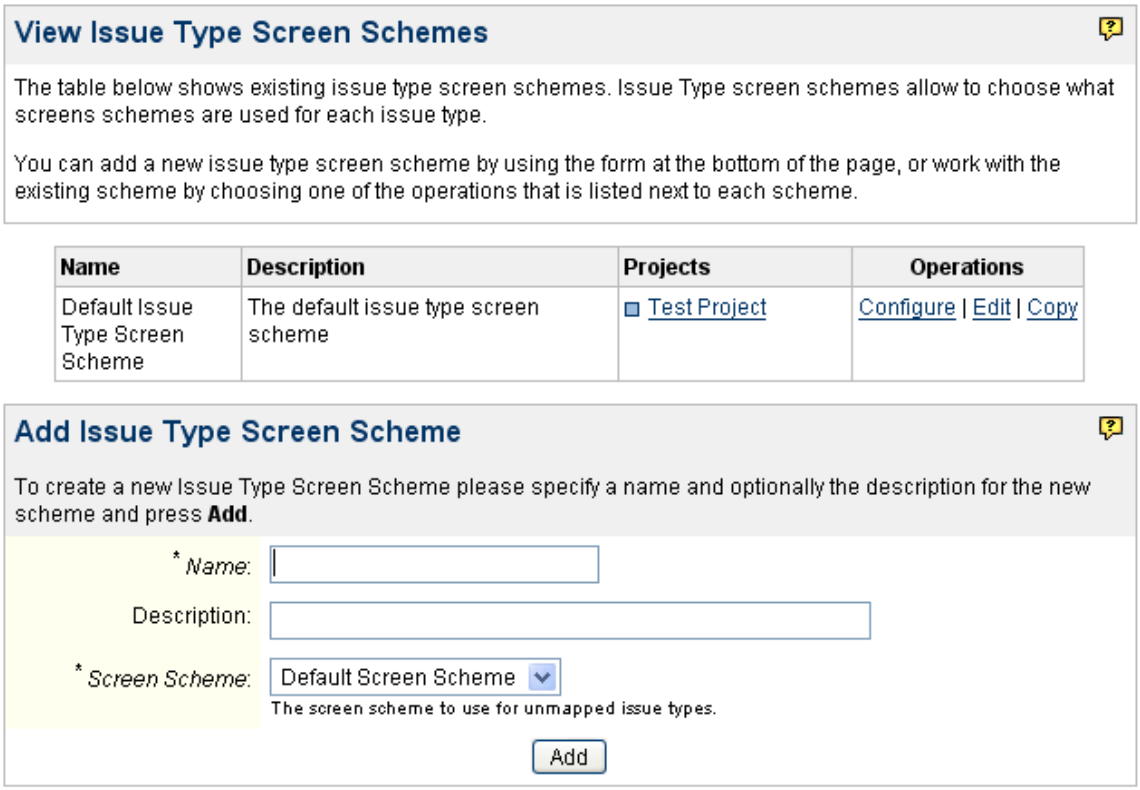
- Copying an Issue Type Screen Scheme
- Configuring an Issue Type Screen Scheme
  - Associating an Issue Type with a Screen Scheme
  - Editing an Association
  - Deleting an Association
- Associating an Issue Type Screen Scheme with a Project

What is an 'Issue Type Screen Scheme'?

An **Issue Type Screen Scheme** associates a Screen Scheme with issue types, allowing you to specify different Screens for the same operation (e.g. 'Create Issue') in the same project for issues of different types.

Adding an Issue Type Screen Scheme

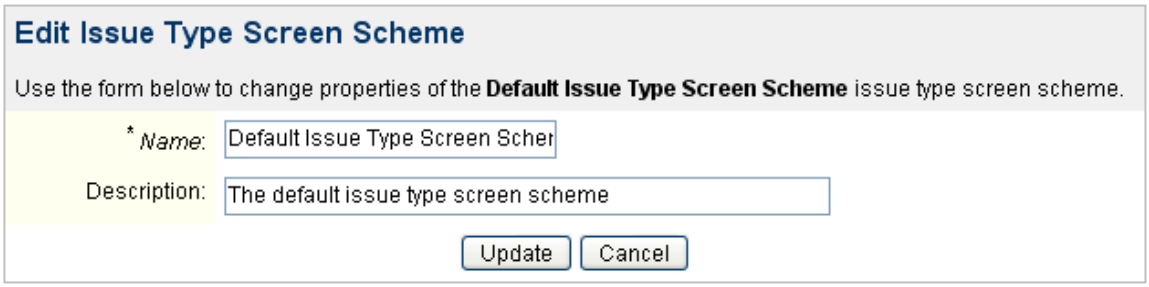
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the '**Issue Fields**' sub-menu on the left hand side if it is not open already, and choose '**Issue Type Screen Schemes**' from the list.
4. This will bring you to the '**View Issue Type Screen Schemes**' page:



5. The 'Add Issue Type Screen Scheme' form is located at the bottom of the 'View Issue Type Screen Schemes' page.
6. Enter the name for the new scheme.
7. You can optionally add a description.
8. Select a Screen Scheme for the *Default* entry in the new scheme. The Screen Scheme mapped to the *Default* entry will be used for issue types that do not have a specific mapping in the scheme.
9. Click the '**Add**' button. The screen will automatically update the Issue Type Screen Schemes list with the new Issue Type Screen Scheme.

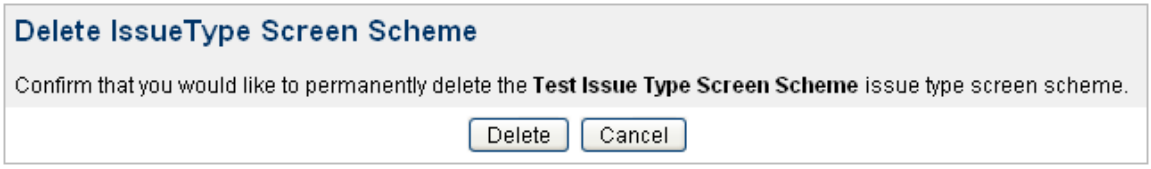
Editing an Issue Type Screen Scheme

1. On the 'View Issue Type Screen Schemes' page, click the '**Edit**' link next to the selected Issue Type Screen Scheme.
2. You will now be directed to the 'Edit Issue Type Screen Scheme' page where you can edit the Issue Type Screen Scheme's name and description as well as the Screen Scheme of the *Default* entry.
3. Click the '**Update**' button.
4. You will be brought back to the 'View Issue Type Screen Schemes' page, with your updates now applied to the Issue Type Screen Schemes list.



Deleting an Issue Type Screen Scheme

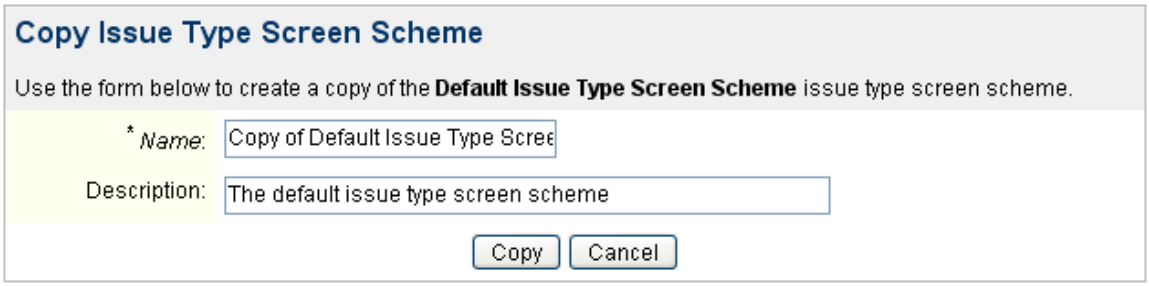
1. On the 'View Issue Type Screen Schemes' page, click the '**Delete**' link next to the Issue Type Screen Scheme you wish to delete.
2. Click the **'Delete**' button to confirm this action.
3. You will be brought back to the 'View Issue Type Screen Schemes' page, with the Issue Type Screen Scheme removed from the Issue Type Screen Schemes list.



Issue Type Screen Schemes that are associated with a project cannot be deleted.

Copying an Issue Type Screen Scheme

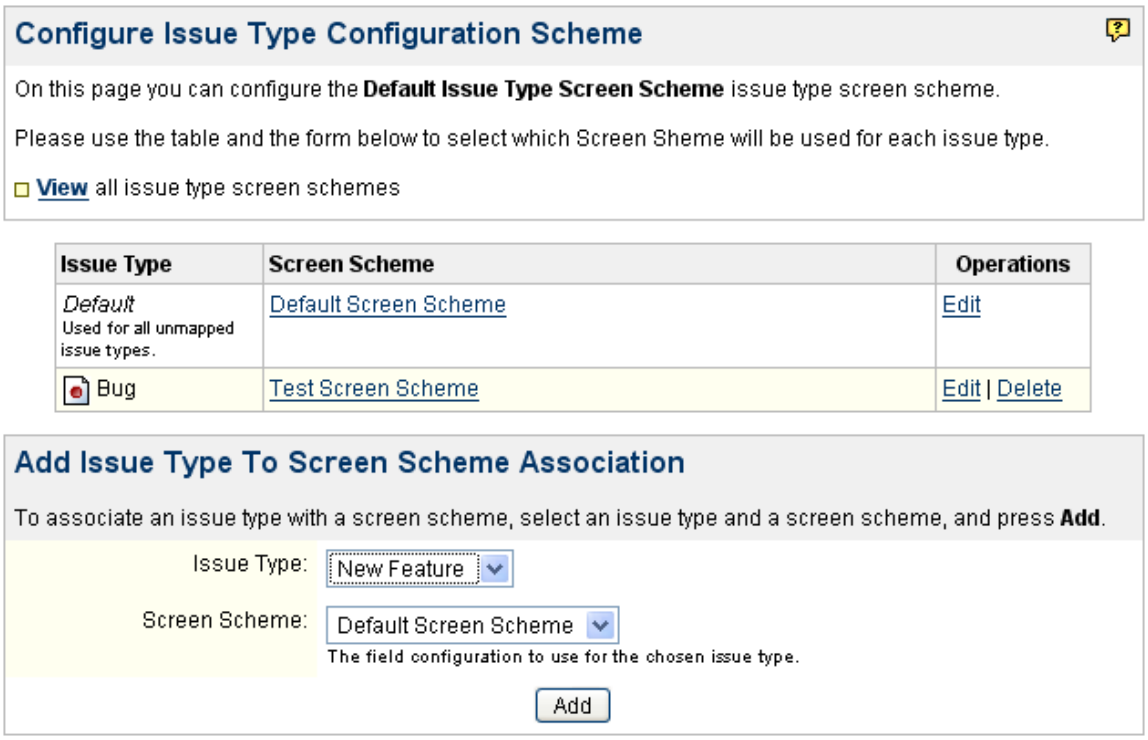
1. On the 'View Issue Type Screen Schemes' page, click the '**Copy**' link next to the field screen you wish to copy.
2. You will now be directed to the 'Copy Issue Type Screen Scheme' page.
3. Enter the name and description of the new Issue Type Screen Scheme.
4. Click the '**Copy**' button.
5. You will be brought back to the 'View Issue Type Screen Schemes' page, and the newly added Issue Type Screen Scheme will have the same scheme settings as the original Issue Type Screen Scheme.



Configuring an Issue Type Screen Scheme

The configuration of an Issue Type Screen Scheme involves associating an issue type(s) with a particular Screen Scheme. For example, associating the 'Bug' issue type with the 'Default Screen Scheme' and then associating the 'Improvement' issue type with the 'Improvement Screen Scheme'.

To configure a given Issue Type Screen Scheme click the '**Configure**' link next to the selected Issue Type Screen Scheme on the 'View Issue Type Screen Schemes' page. You will then be directed to the 'Configure Issue Type Screen Scheme' page:



Associating an Issue Type with a Screen Scheme

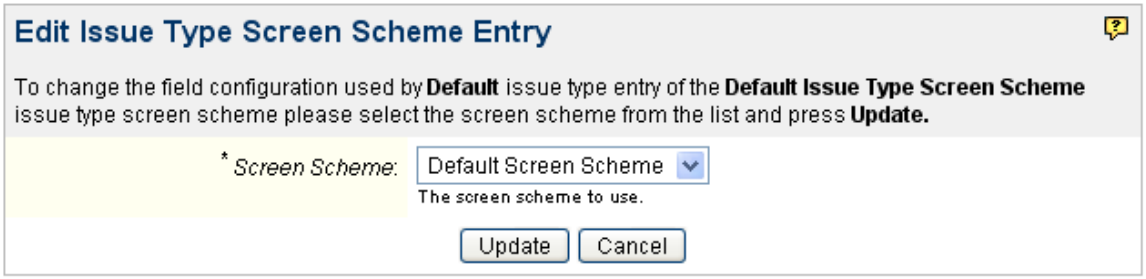
1. The 'Add Issue Type to Screen Scheme Association' is located at the bottom of the 'Configure Issue Type Screen Scheme' page.
2. Select an issue type you wish to associate a Screen Scheme with.
3. Select the desired scheme.
4. Click the '**Add**' button and the new association will be added to the association list above.

There can only be one association for each issue type. If all issue types have been associated with a Screen Scheme you can use the '**Edit**' link next to each entry to change the Screen Scheme that is associated with it.

If there is no specific entry for an issue type, the Screen Scheme associated with the *Default* entry will be used.

Editing an Association

1. On the 'Configure Issue Type Screen Scheme' page, click the '**Edit**' link next to the issue type you wish to edit.
2. You will be brought to the 'Edit Issue Type Screen Scheme Entry' page:



3. Select the screen whose association you wish to change.
4. Click the '**Update**' button and you will be returned to the 'Configure Issue Type Screen Scheme' page.

Deleting an Association

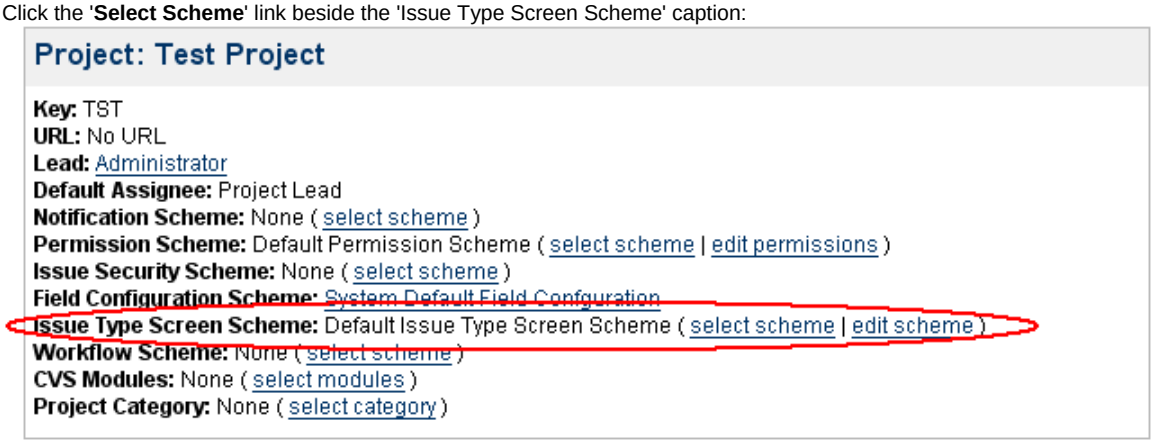
1. On the 'Configure Issue Type Screen Scheme' page, click the '**Delete**' link next to the issue operation you wish to remove.
2. The association will be automatically removed from the table.

The *Default* entry is used for all issue types that do not have a specific entry in the scheme. It cannot be deleted.

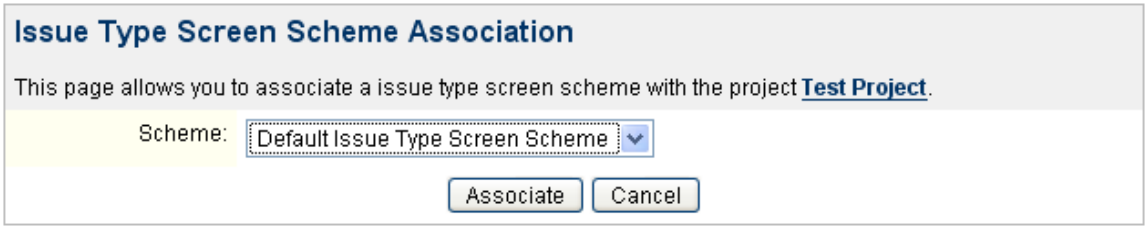
Associating an Issue Type Screen Scheme with a Project

Once you have created and configured an Issue Type Screen Scheme to your desired settings, you can now associate the scheme with a Project. This will apply your chosen Screen Scheme to each issue type within the selected project.

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the 'Administration' page is a list of projects which this user can manage. Select the project of interest.
4. Click the '**Select Scheme**' link beside the 'Issue Type Screen Scheme' caption:



5. Select the screen scheme you wish to associate with this project.



6. Click the '**Associate**' button.

To control which issue types apply to a project, please see 'Associating Issue Types with Projects'.

Configuring Workflow

A JIRA *workflow* is the set of *steps* and *transitions* an issue goes through during its lifecycle. Workflows typically represent business processes.

JIRA ships with a default workflow. The default workflow cannot be edited, but you can customise the issue lifecycle by creating additional workflows. Each workflow can be associated with particular projects and (optionally) particular issue type(s).

On this page:

- Creating a workflow
- Editing a workflow
  - Editing an inactive workflow
  - Editing an active workflow
    - Limitations
- About steps and transitions
  - A note about 'open' and 'closed' issues
- Adding a step
  - Using step properties
- Deleting a step
- Adding a transition
  - Using a screen
  - Adding a condition

        - Combining conditions into groups
    - Adding a validator
    - Adding a post function
        - Using a post function to set a field
        - Using a post function to send a notification
        - Working with transition properties
- Using 'common transitions'
- Using XML to create a workflow
    - Copying a workflow between systems

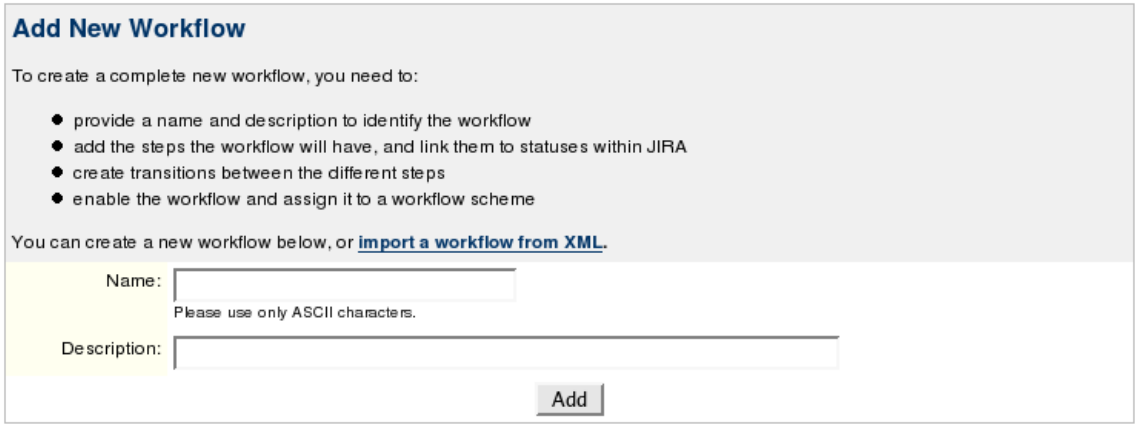
See also:

- Activating Workflow
- Adding a Custom Event

Creating a workflow

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed, listing all the workflows that are currently defined in your JIRA system:

Name	Description	Active?	Schemes	Steps	Operation
jira (Read-only System Workflow)	The default JIRA workflow.	**Active**	Used by projects with no associated workflow shceme and by workflow schemes with unassigned issue types.	5	Steps | XML | Copy
Copy of jira	Workflow for customisation.	**Inactive**		5	Steps | XML | Copy | Delete



5. To create a new workflow in JIRA, either:
    - **Create a 'blank workflow** by using the **'Add New Workflow'** form at the bottom of the page:
        a. In the **'Name'** field, type a name (usually 2-3 words) to identify your new workflow.
        b. *(Optional)* In the **'Description'** field, type a detailed description of your new workflow.
        c. Click the **'Add'** button.Your new workflow will contain one step, called **'Open',** which has an incoming transition called **'Create'.**
    - **Copy an existing workflow** (this is useful if you already have a workflow that is similar to what you need) by clicking the **'Copy'** link next to an existing workflow:
        a. In the **'Name'** field, type a name (usually 2-3 words) to identify your new workflow.
        b. *(Optional)* In the **'Description'** field, type a detailed description of your new workflow.
        c. Click the **'Copy'** button.Your new workflow will contain the same steps and transitions as the workflow you copied.

> If you are copying the default JIRA workflow and wish to rename the transitions, you will need to delete the **'jira.i18n.title'** and **'jira.i18n.description'** properties from all of the transitions. Otherwise, the default names will persist. Read more about transition properties.

6. Once you have created your new workflow you may want to customise it by adding or editing steps and transitions (see below) — especially if you have created a blank workflow.
7. When you have finished customising your new workflow, see *Activating Workflow* for how to use it.

Editing a workflow

Editing a workflow means that you are modifying the steps and transitions that make up a workflow. Read more about modifying steps and

transitions on this page.

The process for editing a workflow differs depending on whether you are editing an **inactive workflow** or an **active workflow**. Restrictions are placed on the modifications you can make to an active workflow, due to the impact the changes will have on projects and/or issue types that the workflow is applied. to.

Editing an inactive workflow

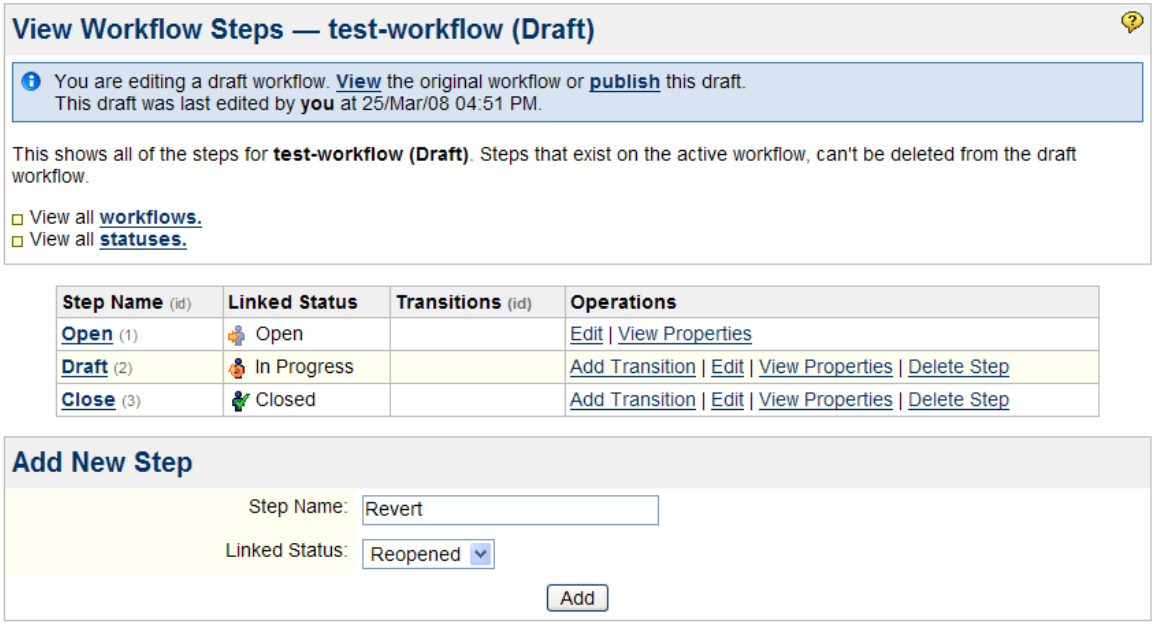
An inactive workflow is a workflow that is not currently being used by any projects . Because there are no issues currently transitioning through an inactive workflow, you can simply edit the workflow's steps and transitions as described below.

Editing an active workflow

To edit an active workflow, you will need to create a 'draft'. You will be able to make quick edits to your live draft with the benefit of real-time validations. Once you publish your changes, you also have the option of saving your old workflow as an inactive backup.

To edit an active workflow:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow that you wish to edit.
5. The **'Workflow Steps'** page will be displayed. Click the **'Create a draft workflow'** link in the information message displayed at the top of the screen.
6. The **'Workflow Steps'** page will be reloaded, as shown below. You will now be able to edit a draft of the workflow as described in the sections above. Any changes that you make to this draft will not affect the active workflow until you publish your draft.



7. When you have completed your changes, click the **'publish this draft'** link in the information message displayed at the top of the screen.
8. A confirmation screen will display, as shown below:
   !workflow-publishdraft.png!Select whether you wish to save the original workflow as an inactive copy. If you choose to retain the original workflow, enter a name for the inactive copy. Click **'Publish'** to publish your draft (i.e. commit your changes to the active workflow).

Limitations

Please note that the following limitations apply when editing an active workflow:

- Existing workflow steps cannot be deleted.
- The associated Status for an existing step cannot be edited.
- If an existing step has no outgoing transitions, it can't have any new outgoing transitions added.
- Step IDs for existing steps cannot be changed.

If you wish to make any of the modifications listed above, then you will need to copy the workflow (see *'Creating a Workflow'* above), modify the copy and then activate it. Please note, this method will be **significantly slower than editing an active workflow,** particularly for large instances of JIRA.

About steps and transitions

A workflow consists of **steps** and **transitions:**

- A **step** represents a stage in a workflow for an issue. An issue can exist in only one step at any point in time. Each workflow step corresponds to (and is usually named after) a 'linked' status. When an issue is moved into a particular step, its **'Status'** field is updated to the value of the step's 'linked' status.
  When defining a step, you can optionally specify **properties** — these allow you to make an issue uneditable while it is in this step.
- A **transition** is a link between two steps. A transition allows an issue to move from one step to another step. For an issue to be able to progress from one particular step to another, a transition must exist that links those two steps. Note that a transition is a *one-way* link, so if an issue needs to move back and forth between two steps, two transitions need to be created.The available workflow transitions for an issue are listed on the issue's **'View Issue'** page. A user can execute a transition (i.e. move the issue through workflow) by clicking one of the available links, e.g.:
  !workflow-avail_transitions.png!When defining a transition, you can optionally specify:
  - A **screen** to be displayed to the user — this is useful if you need the user to provide input before completing the transition.
  - **Conditions** — these control who can perform a transition (i.e. who can see the transition link on the **'View Issue'** page).
  - **Validators** — these check that any user-supplied input is valid before performing the transition.
  - **Post Functions** — these perform particular actions after the transition is complete, e.g.:
    - Assign the issue to a particular user.
    - Send an email notification.
    - Update a field in the issue.
      In the diagram of the default workflow, the five boxes represent steps/statuses **('OPEN', 'IN PROGRESS', 'CLOSED',** etc) and the arrows represent transitions.

A note about 'open' and 'closed' issues

Within JIRA (e.g. in the *'Assigned To Me'*portlet and other portlets), an issue is determined to be 'open' or 'closed' based on the value of its **'Resolution'** field — not its **'Status'** field.

- An issue is determined to be 'open' if its **'Resolution'** field has not been set.
- An issue is determined to be 'closed' if its **'Resolution'** field has a value (e.g. **'FIXED', 'CANNOT REPRODUCE').**
  This is true regardless of the current value of the issue's **'Status'** field **('OPEN', 'IN PROGRESS',** etc).

So if you need your workflow to force an issue to be 'open' or 'closed', you will need to set the issue's **'Resolution'** field during a transition. There are two ways to do this:

- Set the **'Resolution'** field automatically via a post function.
- Prompt the user to choose a **'Resolution'** via a screen.

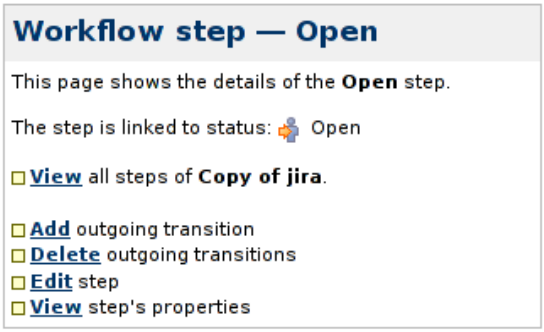
Adding a step

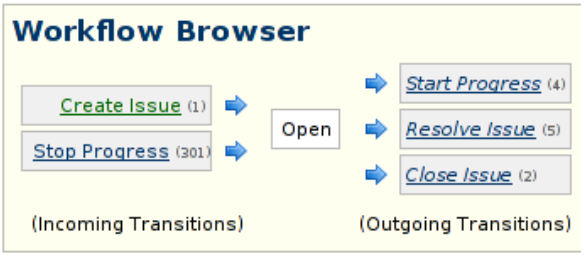
To add a new step to a workflow:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a step.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**. The **'Add New Step'** form appears below the list of steps. (Note: this form will only be shown if the workflow is inactive or you are JIRADOC:editing an active workflow.)

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	*Start Progress* (4) >> In Progress *Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open *Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
Resolved (4)	Resolved	Close Issue (701) >> Closed *Reopen Issue* (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties
Reopened (5)	Reopened	*Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed *Start Progress* (4) >> In Progress	Add Transition | Delete Transitions | Edit | View Properties
Closed (6)	Closed	*Reopen Issue* (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties

6. In the **'Step Name'** field, type a short name for the step. (Note: it is often useful to use the name of the corresponding status.)
7. In the **'Linked Status'** field, select the status that corresponds to this step. Note that each status can only correspond to one step in each workflow, so if all the statuses are already linked to steps in this workflow, you may need to define a new status.
8. Click the **'Add'** button. The **'Workflow Steps'** page will now show your new step in the list.
9. If you wish to view the details of your new step, click the step name. The **'View Workflow Step'** page will be displayed, showing the step's:
    - **Linked Status ('Open'** in the screenshot below).
    - **Incoming Transitions** — that is, transitions whose **Destination Step** is this step.
        - To allow issues to move into this step, there must be at least one incoming transition.
    - **Outgoing Transitions** — that is, transitions whose **Originating Step** is this step.
        - To allow issues to move out of this step, there must be at least one outgoing transition.





10. From this page you can:
    - Edit the step's **Name** or **Linked Status,** by clicking the **'Edit'** link.
    - View and edit the step's **Properties** (see *'Using step properties'* below).
    - View and edit any of the step's **Incoming Transitions** or **Outgoing Transitions,** by clicking the name of a transition. See *'Adding a condition', 'Adding a validator'* and *'Adding a post function'* (below).
    - Add an **Outgoing Transition** to the step (see *'Adding a transition'* below).
    - Delete an **Outgoing Transition.**

Using step properties

You can use step properties to prevent issues from being edited when they are in a particular workflow step(s). For example, in the default JIRA workflow, issues in the **'Closed'** step/status cannot be edited, even by users who have the *'Edit Issue'*permission. Note that issues which cannot be edited cannot be updated using Bulk Edit either.

To stop issues from being editable in a particular step, set the **'jira.issue.editable'** property of the step to **'false'** as follows:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown in *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow whose step you wish to make uneditable.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow.
6. Click the **'View Properties'** link that corresponds to the relevant step.
7. The **'View Workflow Step Properties'** page will be displayed, showing the step's existing properties (if any). The **'Add New Property'** form appears below the list of steps. (Note: this form will only be shown if the workflow is inactive or you are JIRADOC:editing an active workflow.)
8. In the **'Property Key'** field, type: `jira.issue.editable` .
9. In the **'Property Value'** field, type: `false` .
10. Click the **'Add'** button.

Deleting a step

Note: a step can only be deleted if it has no incoming transitions.

To delete a step from a workflow:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow from which you wish to delete a step.
5. The **'Workflow Steps'** page will be displayed.
6. Click the **'Delete'** link that corresponds to the relevant step. (Note: this link will only be shown if the step has no incoming transitions. A workflow step cannot be deleted if it is the destination of a transition.)

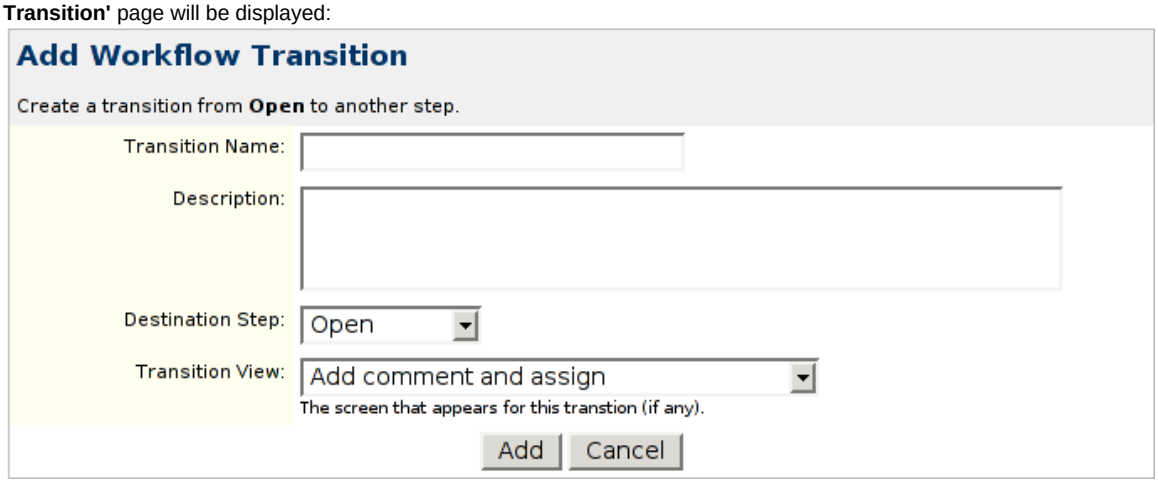
Adding a transition

To add a new transition to a workflow:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a transition.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions:**

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	*Start Progress* (4) >> In Progress *Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open *Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
Resolved (4)	Resolved	Close Issue (701) >> Closed *Reopen Issue* (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties
Reopened (5)	Reopened	*Resolve Issue* (5) >> Resolved *Close Issue* (2) >> Closed *Start Progress* (4) >> In Progress	Add Transition | Delete Transitions | Edit | View Properties
Closed (6)	Closed	*Reopen Issue* (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties

6. Identify the step from which your new transition will originate, and click the **'Add Transition'** link next to the step. The **'Add Workflow Transition'** page will be displayed:



7. In the **'Transition Name'** field, type a short name for the transition. (Note: this name will be shown to users as the transition link in the list of **'Available Workflow Actions'** on the ' **View Issue** ' page.)
8. *(Optional)* In the **'Description'** field, type a short description of the purpose of the transition.
9. In the **'Destination Step'** field, choose the step to which issues will move when this transition is executed.
10. In the **'Transition View'** field, choose either:
    - **'No view for transition'** — choose this if you don't need to prompt the user for input before the transition is executed (i.e. the transition will occur instantly when the user clicks the transition link).
    - The name of a screen that will be shown to users, asking for input before the transition is executed. You can choose one of JIRA's default screens (note: many of these are used in the default workflow and are named after its transitions, e.g. **'Start Progress'** and **'Resolve Issue'),** or any other screen you have created. If no existing screen is suitable, you may want to create a new screen.

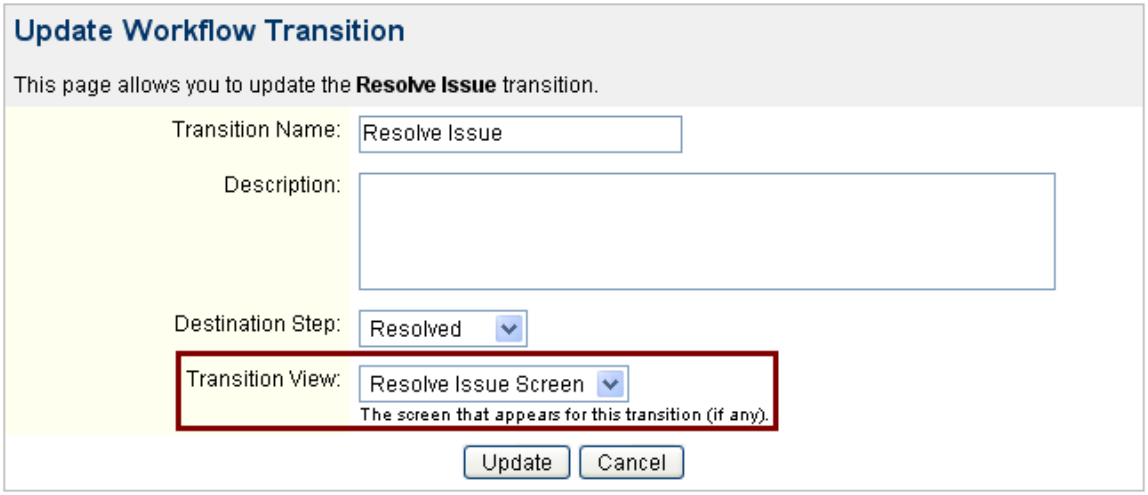
Using a screen

You can use a screen to gather input from a user before a particular transition is executed.

Example: using a screen to set the 'Resolution' field

For a particular step in a workflow, you might need to create a transition that will move the issue to a 'closed' status (e.g. **'CLOSED', 'RESOLVED',** etc) - see 'open' and 'closed' issues. As part of this transition, you might need the user to set the **'Resolution'** field. To do this:

1. Create a screen, e.g. named '`Resolve Issue Screen`', that contains the **'Resolution'** field (and any other fields you want to display).
2. Create/edit your transition, and choose '`Resolve Issue Screen`' in the **'Transition View'** field:



Adding a condition

Conditions control who can perform a transition, and under what circumstances. If a condition fails, the user won't see the transition link on the *View Issue* page.

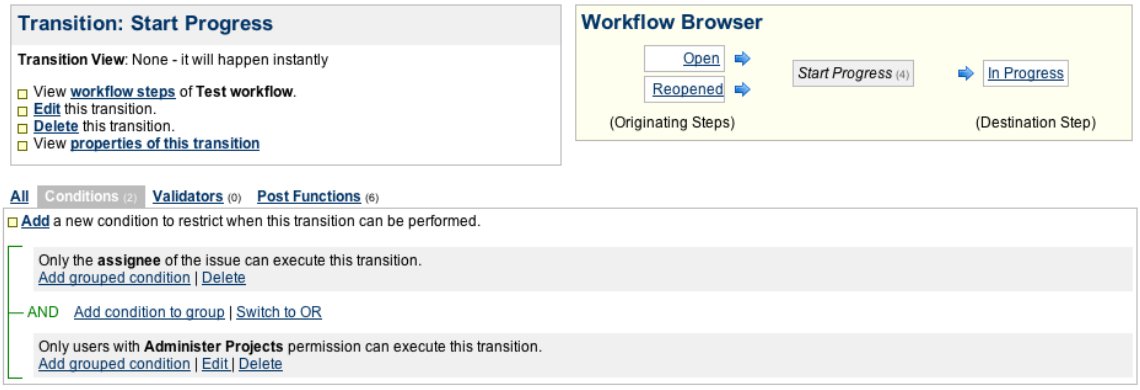
JIRA ships with the following built-in conditions, which are available for you to add to transitions:

Condition	Explanation
Only Assignee Condition	Only allow the issue's current assignee to execute the transition.
Only Reporter Condition	Only allow the issue's reporter to execute the transition.
Permission Condition	Only allow users with a given permission to execute the transition.
Sub-Task Blocking Condition	Block the parent issue transition depending on sub-task status.
User Is In Group	Only allow users in a given group to execute the transition.
User Is In Group Custom Field	Only allow users in a given custom field (of type "Group") to execute a transition.
User Is In Project Role	Only allow users in a given project role to execute a transition.

(You can also create your own conditions via the plugin system. See the *Workflow Plugin Guide* for details.)

To add a condition to a transition:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a condition.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions.**
6. Click the name of the transition to which you wish to add a condition. The **'View Workflow Transition'** page will be displayed:



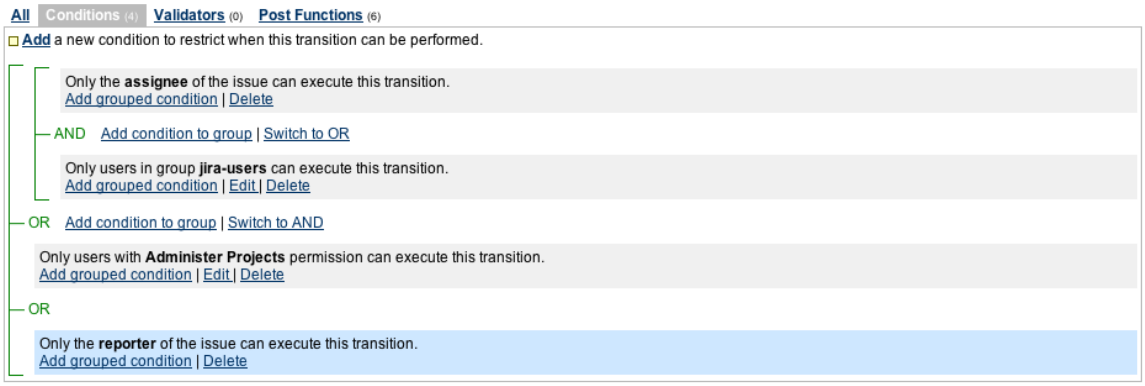
7. Click the **'Conditions'** tab. A list of the transition's existing conditions will be displayed.
8. Click the **'Add'** link. A list of all available conditions will be displayed.
9. Select a condition from the list and click the **'Add'** button.
10. If the condition requires one or more configuration parameters (e.g. the name of a group or project role), the **'Add Parameters To Condition'** page will be presented. Enter your criteria and click the **'Add'** button.
11. The **'Conditions'** tab will be displayed, showing your new condition at the bottom of the list of conditions.Note: from here you can:
    - Click the **'Edit'** link next to the condition's name to edit its configuration parameters (if there are any).
    - Click the **'Delete'** link next to the condition's name to remove the condition
    - Combine your conditions into 'AND'/'OR' groups (see below).

Combining conditions into groups

You can construct complex conditions by combining individual conditions together to form 'condition groups', using a boolean *AND* or *OR.* For example, the following condition group could be constructed:

- Only the assignee of this issue can execute this transition
- **AND**
- Only users in group **jira-users** can execute this transition
  The condition will pass if the user is the assignee of the issue **AND** the user is in the group **jira-users.**

Multiple condition groups can be combined to construct even more complex conditions. Each pair of condition groups can be combined using a boolean **AND** or **OR.** Depending on the structure of the overall condition and its groups, the condition will pass once one or all condition groups have been satisified, e.g:



Adding a validator

Validators check that any user-supplied input is valid before performing the transition. For example, a validator can be used to ensure that the comment entered by a user on the transition's screen meets a certain criteria. If a validator 'fails', the Post Functions of the transition will not be executed and the issue will not progress to the destination step of the transition.

JIRA ships with a number of default validators, which are available for you to add to your transitions. You can also create your own validators via the plugin system.

What is the difference between conditions and validators? Conditions are used to determine whether a transition is 'allowed' to be executed. Conditions cannot validate input parameters that are provided by the user on the transition's screen, since if the condition fails the user is not allowed to start executing the transition and so will not see the transition's screen. Validators have access to the input that has been gathered from the user via the transition's screen, and thus can validate that input.

To add a validator to a transition:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a condition.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions.**
6. Click the name of the transition to which you wish to add a validator. The **'View Workflow Transition'** page will be displayed.
7. Click the **'Validators'** tab.
8. Click the **'Add'** link. A list of all available validators will be displayed.
9. Select a validator from the list and click the **'Add'** button.
10. If the validator requires one or more configuration parameters (e.g. the name of a group or project role), the **'Add Parameters To Validator'** page will be presented. Enter your criteria and click the **'Add'** button.
11. The **'Validators'** tab will be displayed, showing your new validator at the bottom of the list of validators.Note: from here you can:
    - Click the **'Edit'** link next to the validator's name to edit its configuration parameters (if there are any).
    - Click the **'Delete'** link next to the validator's name to remove the validator.

Adding a post function

Post functions carry out some processing immediately after a transition is executed (hence the name *post* function), such as updating an issue's fields, generating change history for an issue, adding a comment to an issue, generating an event (e.g. an email notification).

The JIRA default workflow includes a number of default transitions. Additionally, JIRA ships with the following 'essential' post functions, which are automatically added to every newly-created transition:

Essential post function
Set issue status to the linked status of the destination workflow step.
Add a comment to an issue if one is entered during a transition.
Update change history for an issue and store the issue in the database.
Re-index an issue to keep indexes in sync with the database.
Fire an event that can be processed by the listeners.

The 'essential' post functions cannot be deleted, or reordered relative to each other, as this could compromise other functionality. However, you can insert other post functions between them.

JIRA ships with four built-in post functions which you can optionally add to your transitions:

Optional post function	Explanation
Assign to Current User	Assigns the issue to the user who is executing the transition. (Note: This post function will be ignored unless the user has the *'Assignable User'*permission. You may want to use a condition to ensure that the logged-in user has this permission before executing the transition.)
Assign to Lead Developer	Assigns the issue to the component lead (if one exists) or project lead.
Assign to Reporter	Assigns the issue to the user who created the issue.

Update Issue Field	Updates one of the issue's fields to a given value. Updateable fields are: • **'Assignee'** • **'Description'** • **'Environment'** • **'Priority'** • **'Resolution'** • **'Summary'** • **'Original Estimate'** • **'Remaining Estimate'** Note that this post function cannot update custom fields.

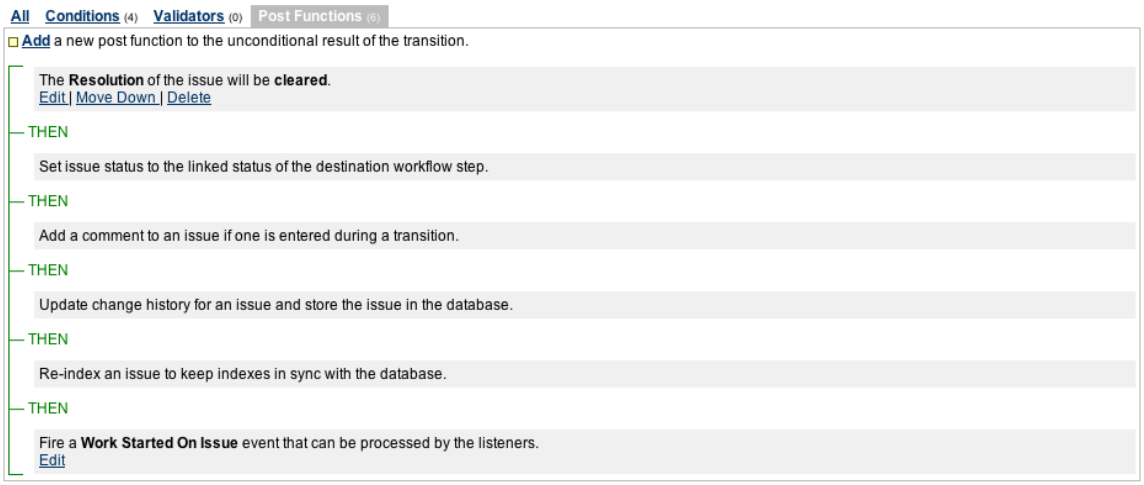
(You can also create your own post functions via the plugin system. See the *Workflow Plugin Guide* for details.)

Note that the four optional post functions must be positioned *before* the **'Update change history for an issue and store the issue in the database'** post function, except when used in the **'Create'** transition.

A note regarding the 'Create' transition:
Sometimes it is useful to perform particular processing (e.g. set a particular field) when an issue is first created. You can do this by adding post functions to the workflow's 'initial transition', which is executed whenever a user creates an issue, and puts the newly-created issue into the workflow's 'initial step'. The 'initial step' is simply the first step in a workflow; every workflow has one, and only one, initial step (called **'Open'** by default, i.e. if you created a blank workflow or copied the default workflow). The 'initial transition' (called **'Create'** by default) is the first incoming transition of the 'initial step'.
When adding one of the optional post functions to the workflow's **'Create'** transition (e.g. you might use the **'Update Issue Field'** transition to set the **'Assignee'** field to a particular user when an issue is created), note that you need to put it *before* the **'Create'** transition's default **'Creates the issue originally'** post function.
Special case:
If you need to set the **'Resolution'** field when creating an issue, put the **'Update Issue Field'** post function *after* the default **'Creates the issue originally'** post function, and use the **'Issue Store'** post function after that. Note that use of the **'Issue Store** post function (which is available only for the **'Create'** transition) should be kept to a minimum, as it does not generate change history and is incapable of persisting fields that have a one-to-many relationship with the issue (e.g. **'Version'** or **'Component').** However, for setting the **'Resolution'** field during issue creation, this post function is useful.

To add a post function to a transition:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a condition.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions.**
6. Click the name of the transition to which you wish to add a post function. The **'View Workflow Transition'** page will be displayed.
7. Click the **'Post Functions'** tab. A list of the transition's existing post functions (if any) will be displayed. For example, the default workflow has the following built-in post functions for the **'Start Progress'** transition:



8. Click the **'Add'** link. A list of all available post functions will be displayed.
9. Select a post function from the list and click the **'Add'** button.
10. If the post function requires one or more configuration parameters (e.g. the name of an event), the **'Add Parameters To Post Function'**

page will be presented. Enter the appropriate information and click the **'Add'** button.

11. The **'Post Functions'** tab will be displayed, showing your new post function at the bottom of the list of post functions.Note: from here you can:
    - Click the **'Edit'** link next to the post function's name to edit its configuration parameters (if there are any).
    - Click the **'Delete'** link next to the post function's name to remove the post function.
    - Click the **'Move Up'** link to move the post function higher up in the list (i.e. it will be executed earlier).
    - Click the **'Move Down'** link to move the post function lower down in the list (i.e. it will be executed later).

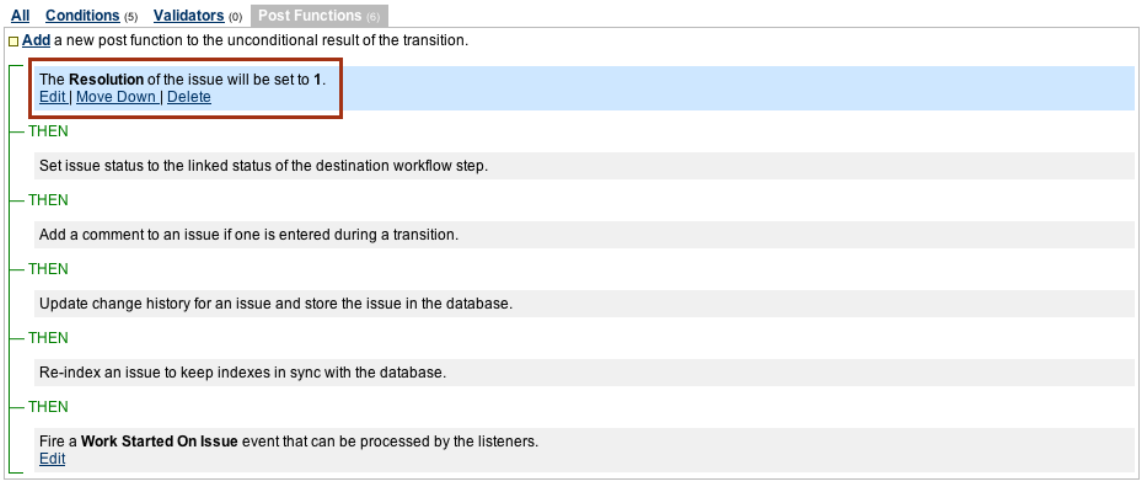
Using a post function to set a field

You can use a post function of type **'Update Issue Field'** to set the value of an issue's field(s) after a particular transition is executed.

Example: using a post function to set the 'Resolution' field

For a particular step in a workflow, you might need to create a transition that will move the issue to a 'closed' status (e.g. **'CLOSED', 'RESOLVED',** etc) - see 'open' and 'closed' issues. As part of this transition, you might want to automatically set the **'Resolution'** field. To do this:

1. Create/edit your transition. In the **'Transition View'** field, either select **'No View For Transition'** or choose a screen that does not contain the **'Resolution'** field (e.g. the **'Add Comment And Assign'** screen).
2. Add a new post function of type **'Update Issue Field'.** Select **'Resolution'** from the **'Issue Field'** select list and select a suitable resolution from the **'Field Value'** select list.
3. Once completed, the transition's list of post functions will appear as follows:



To create a transition that unsets the **'Resolution'** field, follow the same steps but select **'None'** from the **'Field Value'** select list when adding the post function. The list of post functions for this transition will include the following statement:

- **The Resolution of the issue will be cleared.**

Each time one of these transitions is executed, the **'Resolution'** of the issue is automatically set or unset as specified in these post functions.

Using a post function to send a notification

You can use a post function of type **'Fire an event that can be processed by the listeners'** to fire the **'Generic Event'.** The **'Generic Event'** is a built-in JIRA event whose purpose is to allow you to to send email notifications after a particular transition is executed.

Alternatively, you could fire a custom event that you have created specifically for this transition.

When a transition is performed, JIRA will:

- Look up the notification scheme associated with the issue's project, and identify the users associated with the fired event;
- Send an email notification to each user.
  (Note that the fired event is also propagated to all registered listeners.)

Example: using a post function to fire the 'Generic Event'

You can use the **'Generic Event'** to send email notifications. To do this:

1. Create/edit your transition.
2. Go to the transition's **'Post Functions** tab and edit the **'Fire an event that can be processed by the listeners'** post function.
3. On the **'Add Parameters To Post Function'** page, select **'Generic Event'** from the list of events.

Working with transition properties

Properties are key-value pairs that are can be used to further customise transitions. For example, transition properties help to extend the default workflow to allow language translations.

To view the properties of a transition:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown under *'Creating a workflow'* above. Click the **'Steps'** link next to the workflow to which you wish to add a condition.
5. The **'Workflow Steps'** page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions.**
6. Click the name of the transition for which you wish to view the properties. The **'View Workflow Transition'** page will be displayed.
7. Click the **'View properties of this transition'** link. The **'View Workflow Transition Properties'** page will display listing the properties currently set up for the transition. You can also add and delete properties for this transition on this page.

Using 'common transitions'

A 'common transition' is a transition that is defined only once in the workflow, but can be used more than once. That is, a common transition can have more than one originating step. The advantage of common transitions is that if a transition needs to be updated, the update only has to be done in one place.

You can edit common transitions in JIRA, but they cannot be created by the method described in *'Adding a transition'* (above). Instead, to create common transitions, you can either:

- **Copy the default workflow** — the default workflow contains common transitions. Although you cannot edit the default workflow, you can copy it and then edit its steps and transitions to suit your requirements.
- **Create your workflow in XML** — see *'Using XML to create a workflow'* (below).

Using XML to create a workflow

JIRA uses OSWorkflow, a flexible and customisable workflow engine. JIRA's workflow editor generates OSWorkflow XML definition files that are stored in JIRA's database. If you need to take advantage of some OSWorkflow feature that is not available in JIRA's workflow editor (such as 'common' transitions - see above), you can define the workflow in XML and then import it into JIRA as described below.

Once the XML workflow has been imported, JIRA's workflow editor should be able to display most OSWorkflow definitions even if it does not support creating or editing them. For example, conditional results of workflow transitions are displayed in the **'Other'** tab on the **'View Workflow Transition'** page. The **'Other'** tab is only visible if a transition has elements that the editor does not directly support.

To import an XML workflow into JIRA:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **'Global Settings',** click the **'Workflows'** link.
4. The **'View Workflows'** page will be displayed as shown in *'Creating a Workflow'* (above). Locate the **'Add New Workflow'** form at the bottom of the page.
5. Click the **'Import a workflow from XML'** link. The **'Import Workflow'** page will be displayed.
6. In the **'Name'** field, type a name (usually 2-3 words) to identify your new workflow.
7. *(Optional)* In the **'Description'** field, type a detailed description of your new workflow.
8. Either:
    - In the **'Workflow Definition (XML)'** field, paste the contents of the workflow XML file; or
    - In the **'File'** field, type the full path to the file (note that the path must be local, i.e. you will need to first copy the file to your JIRA server).
9. Click the **'Import'** button.

Copying a workflow between systems

Sometimes it is useful to create a workflow in a test system and then copy it into a production system. To do this:

1. In the test system, export the workflow to XML by clicking the **'XML'** link next to the workflow in the list shown on the ' **View Workflows** ' page, and save the output into a file.
2. In the production system, import the file by clicking the 'Import a workflow from XML' link as described in ' *Using XML to create a workflow* ' (above).

Warning
When importing an XML workflow into JIRA:
JIRA's XML workflow definitions contain references to JIRA meta attributes. For example, the id of the linked JIRA status of each workflow step is stored as a 'jira.status.id' meta attribute in the step's definition. Therefore, when manually creating workflows in XML, please ensure that all referenced external entities exist before you import the workflow into JIRA.

Warning
When copying a workflow between systems:
Please note that conditions, validators and post functions can have parameters that might be valid in one system and not in another. For example, different systems might contain different sets of values for the **'Resolution'** field (since it is possible to define your own values). This would be a problem if the **'Update Issue Field'** post function is used to set the **'Resolution'** field to a value that exists in one system but not the other.

Activating Workflow

- How is Workflow 'Activated'?
  - What is a 'Workflow Scheme'?
- Creating a Workflow Scheme
- Associating a Workflow Scheme with a Project
- Editing a Workflow Scheme
- Disassociating a Workflow Scheme from a Project
- Additional Resources

How is Workflow 'Activated'?

Once you have created a new workflow or modified an inactive workflow, you will need to *activate* it. To activate a workflow, you need to:

1. Create a Workflow Scheme that references your workflow, and (optionally) associate it with the relevant issue type(s).
2. Associate the Workflow Scheme with the relevant project(s).

Some terminology:

- **"Active"** workflows are those that are currently being used.
- **"Inactive"** workflows are those that are not associated with any Workflow Schemes, or are associated with Workflow Schemes that are not associated with any projects.

Please note that if you edit an active workflow, it does not need to be re-activated after your changes. Read more about editing active workflows.

What is a 'Workflow Scheme'?

A **Workflow Scheme** associates a particular workflow with particular projects and (optionally) particular issue types. Therefore it is possible to use a different workflow for every project/issue type combination, if you wish.

Creating a Workflow Scheme

To create a workflow scheme:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **"Schemes",** click the **"Workflow Schemes"** link.
4. The **"Workflow Schemes"** page will be displayed, showing a list of all existing workflow schemes in your system. Click the **"Add Workflow Scheme"** link.
5. The **"Add Workflow Scheme"** page will be displayed. Type a **Name** and (optionally) a short **Description** for the new workflow scheme, then click the **"Add"** button. This will create the scheme.
6. The **"Edit Workflows"** page will be displayed, showing your newly-created scheme. Click the **"Assign a workflow to an issue type"** link.
7. The **"Add Workflow To Scheme"** page will be displayed.

- In the **"Issue Type"** drop-down list, select an issue type that is relevant to your workflow. Note: you can also select "All Unassigned Issue Types" to associate your workflow with all issue types that do not have a specific association in this workflow scheme.
- In the **"Workflow"** drop-down list, select the name of your new workflow.
- Click the **"Add"** button.

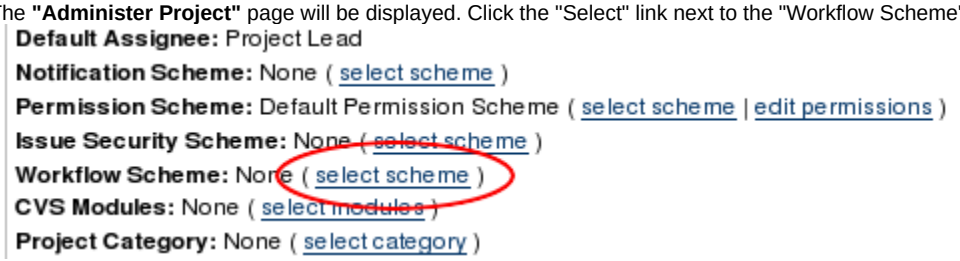
8. Repeat the previous step until your new workflow has been associated with all the relevant issue types. Note that you can choose different workflows for some issue types if you wish.

Once a Workflow Scheme is fully defined you need to associate it with one or more projects (see below) so that the scheme's workflows are actually used by your JIRA system.

Associating a Workflow Scheme with a Project

To associate a workflow scheme with a project:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, click the **"Projects"** link.
4. A list of projects will be displayed. Click the name of the project in which you are interested.
5. The **"Administer Project"** page will be displayed. Click the "Select" link next to the "Workflow Scheme" property of the project.



6. The **"Associate Workflow Scheme to Project"** page will be displayed. Select the relevant scheme from the list and click the **"Associate"** button.
7. Follow the wizard, which will guide you through migrating all the project's issues to the new scheme's workflows.

> You can associate a workflow scheme with more than one project.

Editing a Workflow Scheme

To edit a workflow scheme, i.e. to change which workflows are associated with which issue types:

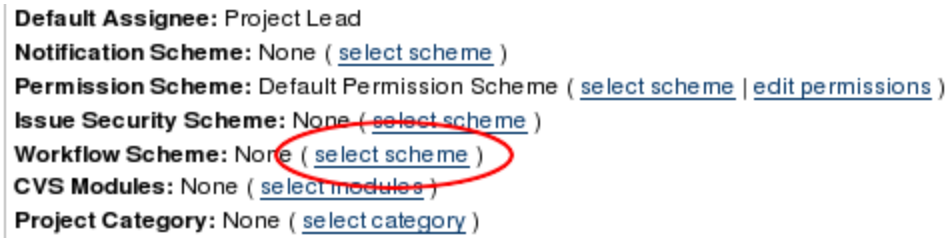
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, under **"Schemes",** click the **"Workflow Schemes"** link.
4. The **"Workflow Schemes"** page will be displayed, showing a list of all existing workflow schemes in your system. Click the **"Workflows"** link next to the workflow scheme in which you are interested.
5. The **"Edit Workflows"** page will be displayed.
   - To associate a workflow with an issue type, click the **"Assign a workflow to an issue type"** link.
   - To disssociate a workflow from an issue type, click the **"Delete"** link.

> It is not possible to edit an active workflow scheme, that is, a workflow scheme that is currently associated with one or more projects. Instead: copy it, edit the new copy and then associate all the relevant projects with the new copy.

Disassociating a Workflow Scheme from a Project

To disassociate a workflow scheme from a project:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, click the **"Projects"** link.
4. A list of projects will be displayed. Click the name of the project in which you are interested.
5. The **"Administer Project"** page will be displayed. Click the "Select" link next to the "Workflow Scheme" property of the project.



6. Select **"None"** from the presented list and click the **"Associate"** button.
7. Follow the wizard, which will guide you through migrating all of the project's issues to the default workflow.

> All projects that do not have an associated workflow scheme use JIRA's default workflow.

Additional Resources

- Workflow scheme overview tutorial video — Watch this short tutorial video to see how to create a new workflow scheme and associate it with a project in JIRA. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Adding a Custom Event

On this page:

- Overview of JIRA Events
  - Event Types
    - System Events
    - Custom Events
- Configuring Notifications for a Custom Event
  - Step 1. Add a Custom Event
  - Step 2. Configure Notification Scheme to send mail on Custom Event
  - Step 3. Configure Workflow Transition Post-Function to Fire Custom Event
- Updates to Workflows on Disk

Overview of JIRA Events

JIRA uses an event-listener mechanism to alert the system that something has happened, and to peform appropriate action (e.g. send an email notification) based on the event that has occurred. Every *issue operation* within JIRA is associated with a particular *event* - e.g. the `Issue Created` event is fired when an issue has been created.

A Listener can execute a specified action once it has been notified that a particular event has been fired. For example, the MailListener can send an `Issue Created` email to a list of recipients defined in the appropriate Notification Scheme, whenever an issue is created.

Some events are fired by JIRA internally — e.g. an `Issue Updated` or `Issue Moved` event. Other events are fired from within workflow transition post-functions — e.g. an `Issue Resolved` event, or a Custom Event (see below).

Event Types

There are two types of events within JIRA:

- **System** — System events are used throughout JIRA internally, and cannot be added or deleted. You can, however, make them Inactive (see below).
- **Custom** — ***(JIRA Enterprise & Professional editions only)*** Custom events are used to generate an email notification (or invoke a listener) from a particular workflow transition's post-function. You can add/delete as many custom events as you need. Note that only *inactive* custom events can be deleted.

An event can be in either of the following states:

- **Active** — the event is associated with at least one notification scheme or workflow transition post-function
- **Inactive** — the event is not associated with any notification schemes or workflow transition post-functions.
  Note that the event state does not indicate whether the event is able to be fired. A **custom event** will only be fired if it is associated with a transition post-function for an active workflow (see 'Activating Workflow').

System Events

JIRA's built-in system events are:

Issue Created:	An issue has been entered into the system.
Issue Updated:	An issue has had its details changed.
Issue Assigned:	An issue has been assigned to a new user.
Issue Resolved:	An issue has been resolved (usually after being worked on and fixed).
Issue Closed:	An issue has been closed. (Note that an issue may be closed without being resolved; see Statuses).
Issue Commented:	An issue has had a comment added to it.
Issue Comment Edited:	An issue's comment has been modified.
Issue Reopened:	An issue has been re-opened.
Issue Deleted:	An issue has been deleted.
Issue Moved:	An issue has been moved into this project.
Work Logged On Issue:	An issue has had hours logged against it (i.e. a worklog has been added).
Work Started On Issue:	The Assignee has started working on an issue.
Work Stopped On Issue:	The Assignee has stopped working on an issue.
Issue Worklog Updated:	An entry in an issue's worklog has been modified.
Issue Worklog Deleted:	An entry in an issue's worklog has been deleted.
Generic Event:	The exact nature of this event depends on the workflow transition post-function(s) which invoke it. As with Custom Events, you can use the Generic Event to generate an email notification (or invoke a listener) from a particular workflow transition's post-function (see Workflow and Notifications).

Custom Events

You can fire a **custom event** from a custom transition post-function in a custom workflow. The appropriate listeners will be alerted of the custom transition by the firing of this event. For example, the associated notification scheme can be configured to notify users of the workflow transition based on the firing of this custom event.

Configuring Notifications for a Custom Event

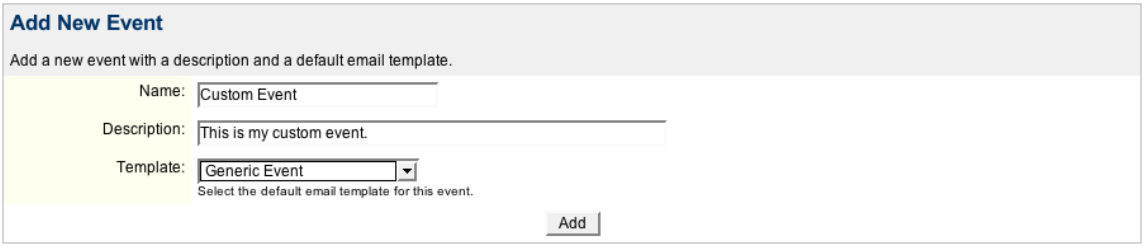
Custom events are most commonly used to generate notifications for custom workflow transitions. For example, your organisation might need you to modify the default workflow by adding a workflow step called "QA_Inspection" (e.g. between "Resolve Issue" and "Close Issue"). You would typically also need to generate an email notification to the QA team whenever an issue progresses to the "QA_Inspection" step of the workflow.

There are three overall steps to achieve this:

1. Add a custom event to the system (e.g. "Issue Awaiting QA").
2. Configure the notification scheme to send an email when the custom event is fired.
3. Configure the workflow transition post-function to fire the custom event.

Step 1. Add a Custom Event

1. Navigate to the **'Administration'** link from the main menu.
2. Select the **'Events'** link under the **'Global Settings'** section in the sub-menu.
3. Add a name and description for the new event.
4. Select a default email template to be associated with the event.
5. Click **'Add'**.

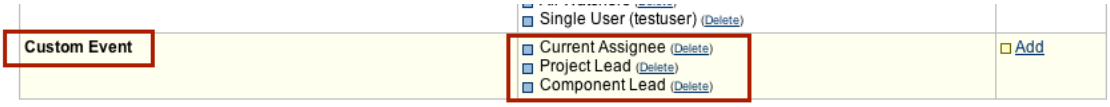


The custom event must be associated with a default email notification template. A notification scheme configured to notify users of this event will use this email template when sending the notification.

The custom event will appear in the list of events defined within the system. Initially, the event will be marked as **inactive** as it is not associated with a notification scheme or workflow post-function.

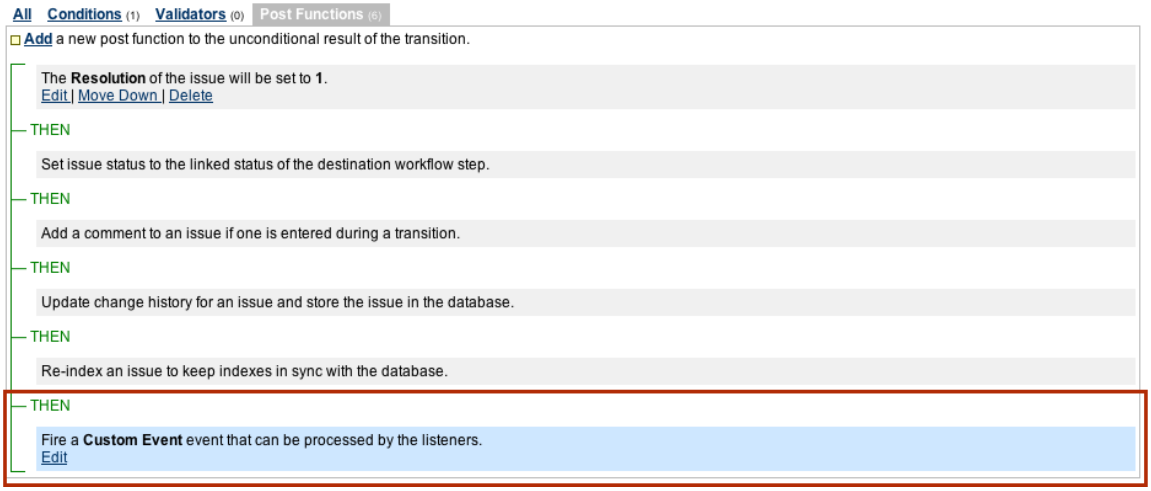
Step 2. Configure Notification Scheme to send mail on Custom Event

1. Navigate to the **'Administration'** link from the main menu.
2. Select the **'Notification Schemes'** link under the **Schemes** section in the sub-menu.
3. Select the notification scheme to edit.
4. Add the recipients for the custom event as required - further details available here.



Step 3. Configure Workflow Transition Post-Function to Fire Custom Event

1. Navigate to the **'Administration'** link from the main menu.
2. Select the **'Workflows'** link under the **'Global Settings'** section in the sub-menu.
3. Navigate to workflow transition post-function screen to be edited - further details available here.
4. Update the post-function to fire the custom event.
5. Activate or associate the workflow (and scheme) with the appropriate project (see 'Activating Workflow').



Updates to Workflows on Disk

As of JIRA 3.6, all event references are made through the `EVENT ID`. For pre-JIRA 3.6 data, all database tables (Workflow, Notification, etc.) are

updated automatically. However, it is necessary to manually update event references in workflows saved to disk. This upgrade guide provides details on the changes required.

Configuring Email Notifications

JIRA can send email notifications to users when significant events occur.

On this page:

- Enabling Email Notifications
- Disabling Email Notifications
- Configuring a Project's Email Address
- Email Recipients
- Email HTML Formatting

Enabling Email Notifications

To enable email notifications in JIRA,

1. Configure an SMTP Mail Server.
2. Configure a notification scheme and associate it with the appropriate projects.

It is possible to customise your email content. The email address from which notifications are sent can also be configured for each project.

Disabling Email Notifications

To disable email notifications for a project, you can remove the notification scheme from the project by editing the project and selecting 'None' as the project's notification scheme.

Alternatively, you can edit the notification scheme so that no emails are sent.

Configuring a Project's Email Address

It is possible to configure the project email address that notifications are sent from.

By setting the 'Sender' email address for a project, all notifications will be sent from this address. This setting is specific to the project selected and will not affect the configuration of the other projects. The default address specified in the SMTP Mail Server configuration is used as the default "sender" address for all projects.

The "sender" email address can be configured as follows:

1. From the Administration view, select "Projects" to view all projects. Select the project to be configured.
2. Select "Edit Configuration" from the "Mail Configuration" entry in the project detail list.
3. Enter a valid email address in the "sender" field and click "Confirm" to complete the process. This email address will now be used as the "sender" address in all notifications for this project.
4. The default email address as specified in the SMTP Mail Server can be reinstated by clicking the "Reset" button.

This option is not accessible unless a SMTP Mail Server has been previously configured.

Email Recipients

For each event notification, JIRA will only send the first encountered email intended for a recipient. Hence, in the case where a user is included in two or more recipient lists (e.g. Project Lead and Current Reporter) for one event notification, the user will only receive the first encountered email notification. JIRA will log the fact that this user was on multiple recipient lists.

Email HTML Formatting

Each JIRA user can specify in their Profile Preferences whether to send outgoing emails in text or HTML format; JIRA administrators can specify a defaut email format under 'User Defaults' in the Administration menu.

Since Jira 3.6.1, the HTML email format was improved to accomodate internationalised words in the 'Issue Details' section. However, due to Internet Security Settings, which prevent automatic download of images, the HTML e-mail may not be correctly formatted. For example, the summary column on the left may appear too wide. It is possible to correct the formatting by accepting to download these images. On some e-mail

clients it is possible to do this in two different ways:

1. per each email;
    - **Mozilla Thunderbird** — by clicking on the *"Show Images"* button above the e-mail
    - **Microsoft Outlook 2003** — by clicking on the *"Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message."* message above the e-mail
    - **Microsoft Outlook 2000** — does not have this option, it always downloads images
    - **Microsoft Outlook Express 6** — by clicking on the *"Some pictures have been blocked to help prevent the sender from identifying your computer. Click here to download pictures."* message above the e-mail
2. configuring the e-mail client;
    - **Mozilla Thunderbird 1.5** — Navigate to **Tools -> Options -> Privacy -> General** tab and ensure that *"Allow remote images if the sender is in my:"* option is checked and note which address book is selected. Then return to the e-mail sent from JIRA, right-click on the sender's e-mail address and choose *"Add to address book..."* option, adding this contact to the same address book as was selected in the Privacy options
    - **Microsoft Outlook 2003 and Outlook Express 6** — Navigate to **Control Panel -> Internet Options.** On the Security tab, add JIRA's base URL to the trusted sites

Creating a Notification Scheme

JIRA can generate email notifications for various events that happen during the issue lifecycle. Notifications are defined within a *notification scheme* (see below), which associates particular events with particular email recipients. The notification scheme is then assigned to a particular project; note that you can use the same notification scheme for more than one project.

The events which can generate email notifications are:

Issue Created:	An issue has been entered into the system.
Issue Updated:	An issue has had its details changed.
Issue Assigned:	An issue has been assigned to a new user.
Issue Resolved:	An issue has been resolved (usually after being worked on and fixed).
Issue Closed:	An issue has been closed. (Note that an issue may be closed without being resolved; see Statuses).
Issue Commented:	An issue has had a comment added to it.
Issue Comment Edited:	An issue's comment has been modified.
Issue Reopened:	An issue has been re-opened.
Issue Deleted:	An issue has been deleted.
Issue Moved:	An issue has been moved into this project.
Work Logged On Issue:	An issue has had hours logged against it (i.e. a worklog has been added).
Work Started On Issue:	The Assignee has started working on an issue.
Work Stopped On Issue:	The Assignee has stopped working on an issue.
Issue Worklog Updated:	An entry in an issue's worklog has been modified.
Issue Worklog Deleted:	An entry in an issue's worklog has been deleted.
Generic Event:	The exact nature of this event depends on the workflow transition(s) from it was fired.
Custom Event(s) :	The exact nature of these events depends on the workflow transition(s) from which they were fired.

Note that email notifications will only be sent to people who have permission to view the relevant issue - that is, people who:

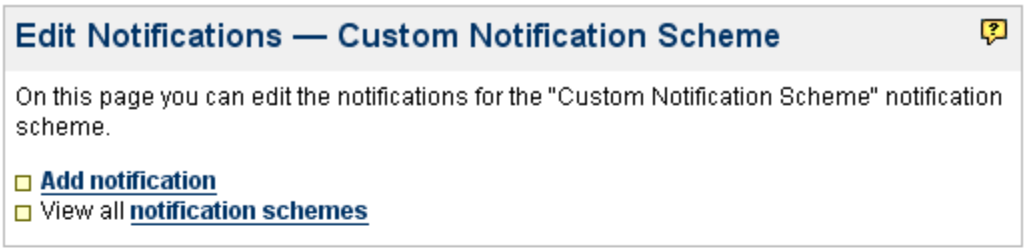
- have the 'Browse Project' permission for the project to which the issue belongs; and
- are members of any Issue security levels that have been applied to the issue.

Also note that JIRA can only send email notifications if SMTP email has been enabled (see Email Overview).

Creating a Notification Scheme

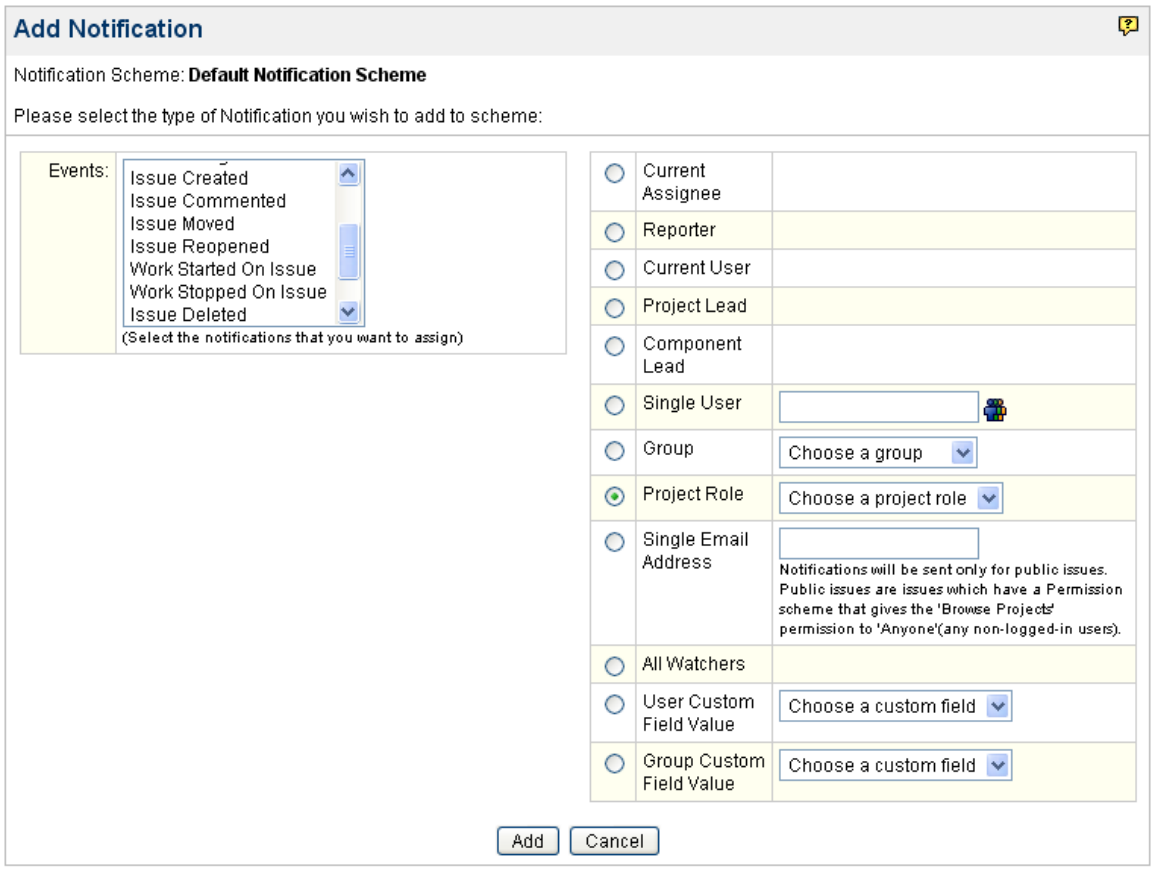
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Schemes, click the link labelled '**Notification Schemes**'.

4. This will display the 'Notification Schemes' page. This page lists all of the notification schemes that JIRA currently has. Click the '**Add Notification Scheme**' link.
5. In the 'Add Notification Scheme' form, enter a name for the notification scheme, and a short description of the scheme. Click the '**Add**' button.
6. You are then shown the 'Edit Notifications' page. This page lists all of the above mentioned issue life cycle events, along with whom should be notified. It is currently empty.
7. Click the '**Add**' link in the appropriate life cycle event row.



Event	Notifications	Operations
Issue Created (System)		Add
Issue Updated (System)		Add
Issue Assigned (System)		Add
Issue Resolved (System)		Add
Issue Closed (System)		Add
Issue Commented (System)		Add
Issue Reopened (System)		Add
Issue Deleted (System)		Add
Issue Moved (System)		Add
Work Logged On Issue (System)		Add
Work Started On Issue (System)		Add
Work Stopped On Issue (System)		Add
Generic Event (System)		Add

8. This will display the 'Add New Notification' page. Here you can choose who to notify, from the list of alternatives.



Where:

- **Current Assignee** is the user assigned to the issue.
- **Current Reporter** is the user who originally created the issue
- **Current User** is the user who performed the action triggering the event in question.
- **Single Email Address** is any email address that you wish to alert.

> A Single Email Address notification will only be sent if the issue is publicly viewable (as the email address of a non-JIRA user could be specified, in which case a security check is not possible). Publicly viewable issues are issues which have a Permission scheme that gives the 'Browse Projects' permission to 'Anyone' (any non-logged-in users).

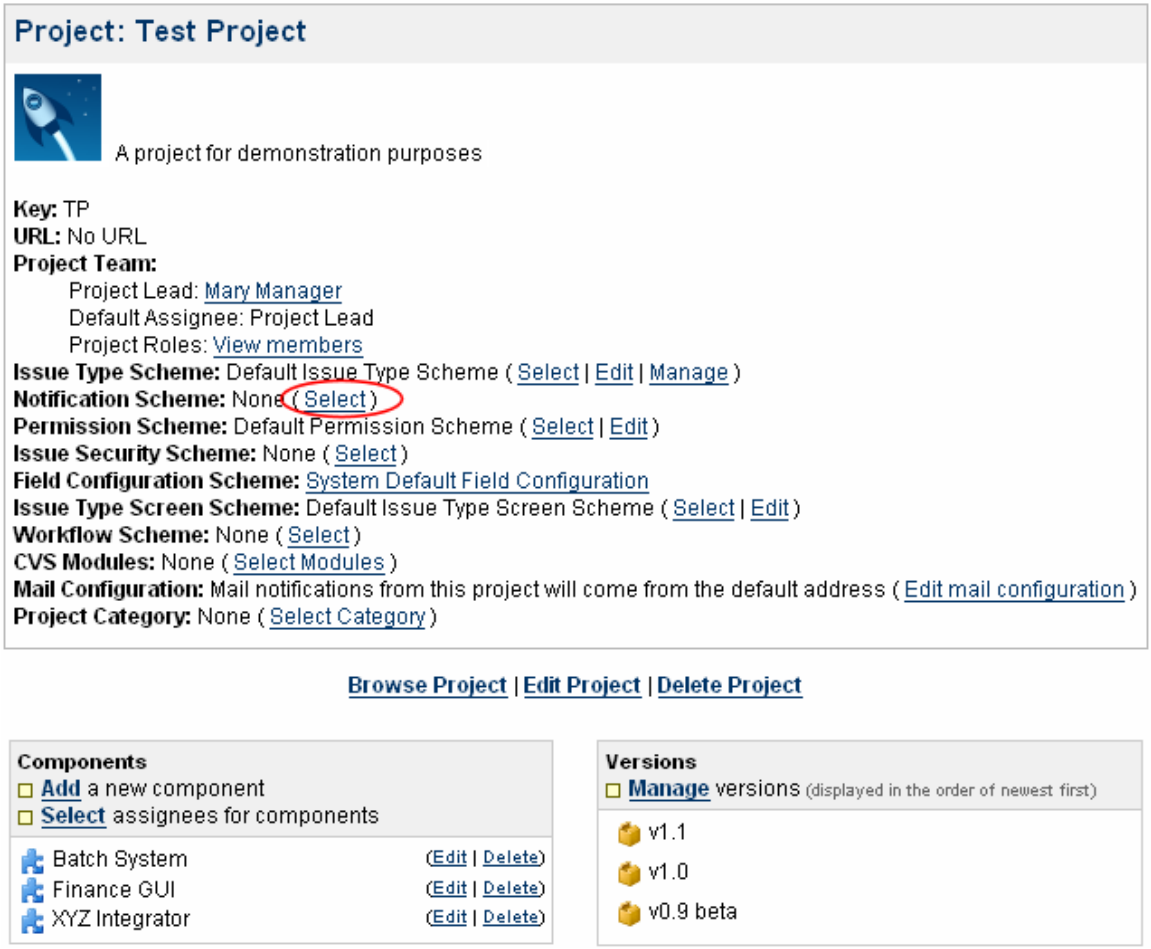
- **User Custom Field Value** is any custom field value of type *User Picker* or *Multi User Picker* that may have been associated with issues. An example of where this can be useful, you have a custom User field called Tester, you have the tester notified when an issue is resolved.
- *(the rest are hopefully self-evident)* Note that project roles are useful for defining specific team members for each project. Referencing project roles (rather than groups) in your notifications can help you minimise the number of notification schemes in your system.

9. After selecting the appropriate option, and filling in any required information for that option, click the '**Add**' button.
10. You will be taken back to the 'Edit Notifications' page, with the notification you just specified now listed against the appropriate issue life cycle event.
11. Repeat steps 7 through 11 until you have specified all the notifications you want to happen.
12. If you make a mistake, or you would like to remove who is being notified, simply do so by clicking the '**delete**' link beside the person/group/role.

Assigning a Notification Scheme to a Project

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. A list of projects is displayed

4. Select the project you want by clicking on the project name. This will display the project details
5. Click the '**Select**' link beside the Notification Scheme caption.



6. This will bring up a list of notification schemes. Select the notification scheme that you want to associate with this project.
7. Click the '**Associate**' button to associate the project with the notification scheme.

See also Minimising the number of Permission Schemes and Notification Schemes.

Customising Email Content

JIRA generates emails in reaction to events using a templating engine. The templating engine is Apache Jakarta's Velocity. This is a relatively easy to use templating language that can pull apart java objects in useful ways. The mails are generated inside JIRA by invoking Velocity with a set of objects of relevance to the event.

To customise email content, please follow this procedure.

1. Open up your JIRA distribution, and navigate to the following paths:
   - **Standalone:** atlassian-jira-2.0\atlassian-jira\WEB-INF\classes\templates\email\
   - **Source:** jira\src\etc\java\templates\email\
   - **WAR:** webapp\WEB-INF\classes\templates\email\
2. Under this directory there are two directories, html and text. The html subdirectory contains the templates used to create emails in html, while the text directory the plain text mail outs. The templates are named after the event that will trigger the email.

3. Bring the template up in your favourite text editor. Referring to the JIRA template documentation and Velocity Users Guide, make the customisations you want.
4. Restart JIRA.

See also Adding Custom Fields to Email.

Creating Issues and Comments from Email

JIRA can be configured to automatically create issues or comments based on incoming emails. This is especially useful in a helpdesk or support scenario, where users send support queries via email, which you wish to track with JIRA. Subsequent emails about the issue, for example responses to Email Notifications, can be automatically recorded as comments. Additionally, any attachments in the emails can be automatically attached to the issue (with appropriate configuration).

On this page:

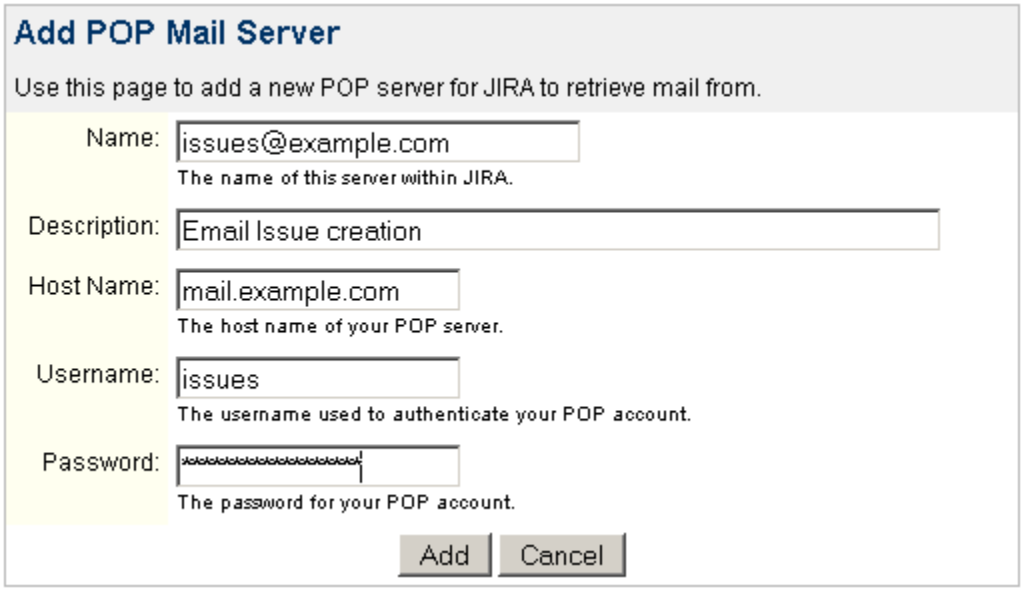
- Mail Service Configuration
- Issue/Comment Creation
- Handler Parameters
- Other Handlers
- Email pre-processing
- Troubleshooting
- Additional Resources

Mail Service Configuration

To set up issue and comment creation via email, you will need to create a mail account on your server (usually one mail account for each project). For example, for the 'ABC' project, you might establish an account `abc-issues@yourcompany.com`. This mail box should be accessible via POP, IMAP, or on the local filesystem. JIRA will periodically scan this mail box, and appropriately create issues or comments for any emails it finds, and - optionally - create new user accounts for senders not previously seen (note that this is not possible if you are using External User Management).

Once you have established a mail account, here is how to configure JIRA to periodically scan it (POP access assumed):

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Global Settings'**, click the link labelled **'Mail Servers'**.
4. Click the **'Configure new POP mail server'** link.
5. This will bring up the "Add POP Mail Server" page.



Fill in as follows:

- '**Name:**' put a short descriptive name, possibly just the email address that will be collected by this service
- '**Description:**' put a short phrase that describes this service, probably 'Email Issue Creation/Comments for <Project>'.
- '**Hostname:**' put the name of your POP server
- '**Username**' and '**Password**' use the email account details as created in step 1.
  Note that the use of SSL is specified later in the service, not here in the Mail Server.
  If you need to set a non-standard port, this will need to be done by setting a `-Dmail.pop3.port=<port>` property (instead of

pop3 you can specify `pop3s`, `imap` or `imaps`). See JRA-11037 for more on this.

6. This will bring you back to the 'Email Servers' page, where you should see a new POP server listed. You can edit and delete this server here.
7. On the panel on the left, under the title 'System', click the link labelled **'Services'**.
8. This will bring up the 'Services' page. It lists the current services running on this system. On a vanilla system there should be one service running — **Mail Queue Service.** You cannot delete the **Mail Queue Service.** Additionally, if you have enabled the option to automatically backup JIRA's data, you will also see the **'Backup Service'** listed here too.
9. Fill in the 'Add Service' form as follows:
   !issue_creation_email-addservice.png|title="Add service"!#* **'Name'** — enter a descriptive name, eg "Create Issue/Comment Service for <Project>".
   - **Class** — select the appropriate option presented in the drop down list, or enter **com.atlassian.jira.service.services.pop.PopService,**
   - **Delay** — this is best left as 1 minute.
10. Click **'Add Service'**.
11. This will bring up the "Edit Service" screen to configure the service.
    !issue_creation_email-editservice.png|title="Editing a service"!#* **'Handler'** — select "Create Or Comment Handler" from the drop down box. Set Handler parameters to something like:

```
project=JRA, issuetype=1, createusers=true, bulk=forward
```

Further details on the **handler parameters** are available below. If you choose to connect over SSL, you will need to import and verify the server's SSL key before JIRA will be able to connect. See Connecting to SSL Services for more information.

- **Forward Email** — this parameter specifies an email address to which error notifications and (optionally) unhandled emails can be forwarded (see "bulk" parameter below). Any unhandled mails or failures encountered in this process are logged and forwarded in an email to this address.

1. Click the **'Update'** button and the service will be in effect.

Issue/Comment Creation

JIRA examines the email subject and the in-reply-to message for an existing issue reference to determine whether a new issue or comment should be created. A new issue is created if an existing issue reference is not found — otherwise, a comment is added to the issue referenced in the email. The email to foo@atlassian.com will be processed as follows:

- Issue Creation:
  - The subject of the email will become the issue summary.
  - The body of the email will be the issue description.
  - A bug (since issue type has been set to 1 in this example) will now be created for project "JRA" with the above information.
  - Any attachments to the email will become attachments to the issue (assuming attachments have been enabled in JIRA). Note that, to ensure compatibility with various operating systems, any of the following characters in the filename will be replaced with an underscore character: \, /, ", %, :, $, ?, *, <, |, >.
- Comment Creation:
  - The body of the email will become a comment on the issue
  - Any attachments to the email will become attachments to the issue (assuming attachments have been enabled in JIRA)

> The **Subject** of the email becomes the issue summary. As all issues require a summary, each email intended for issue creation should include a **Subject.**

Handler Parameters

project parameter is the project key.

> The **project** parameter is only relevant for issue creation, not for issue commenting. If an email contains an issue key in the email subject, and that issue exists in the JIRA instance, the handler will add the email as a comment on the issue, regardless of which project the issue is in.

These are the numbers associated with the default **issue types:**

- **Bug:** `issuetype=1`
- **New Feature:** `issuetype=2`
- **Task:** `issuetype=3`
- **Improvement:** `issuetype=4`
- **Sub-task:** `issuetype=5`

You can use the method described here to determine what numbers are mapped to your issue types.

Besides **project** and **issuetype,** the following parameters are allowed:

- **createusers** — if **createusers** is set to true, people who don't currently have an account in JIRA will have it created for them. This allows the creator to be notified of subsequent updates to the issue, by configuring the notification scheme to notify the 'Reporter' of updates.
- **reporterusername** — this sets which user will be the "reporter" of created issues, for emails whose sender does *not* match that of an existing user. Normally JIRA will ignore emails from addresses not matching an existing user. For instance, to allow anonymous users to create issues via email, you can create an anonymous user or dummy account on JIRA and set the **reporterusername** to point to this account. When the "reporterusername" parameter is specified, the "from" address of the email is added at the end of the comment of the issue, so you can identify the sender.
- **notifyusers** — this parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added.
- **ccassignee** — if the To, Cc, or Bcc field of an email contains the address of a user already present in JIRA, then by default JIRA will **assign** the issue created from the email to that user. JIRA will attempt to assign the issue to a user from the To field first, then the Cc field and finally the Bcc field, if it cannot find a match in the To or Cc fields. If you do not wish JIRA to automatically assign issues in this way, then set **ccassignee** to **false.**
- **bulk** — this parameter determines how to handle "bulk" emails (those sent by an automated service, notably JIRA itself), indicated by a "Predecence: bulk" header or an "Auto-Submitted" header that is not set to "no". Possible values are:
  - *ignore* — Ignore the email and do nothing
  - *forward* — Forward the email to the address set in the "Forward Email" text field
  - *delete* — Delete the email permanentlyIt is generally a good idea to set **bulk=forward** and set a Forward Email address, to prevent mail loops between JIRA and another automated service (eg. another JIRA installation).
- **catchemail** — this causes JIRA to only process emails sent *to* the specified email address. All other emails are ignored. This is useful if you have multiple aliases for the same email Inbox, eg. `foo-support@example.com` and `bar-support@example.com` aliases for `support@example.com`, and you want one email service each, eg. to create issues in FOO and BAR projects respectively. Please note that **this parameter is rarely useful,** and should not be confused with the more common **reporterusername.**
  You can only specify one catch email address and one issue type per listener.

Other Handlers

For more information on other handlers that are shipped with JIRA please refer to this document.

Email pre-processing

For production use, we recommend that you set up the following email pre-processing:

- Ensure mail is sent to a backup folder, so there is a record of what JIRA processed.
- If the POP box contains email replies to JIRA notifications, set up rules filtering out email auto-replies and bounces.
  If you do not do this, there is a strong possibility of mail loops between JIRA and autoresponders like vacation scripts. JIRA sets a 'Precedence:bulk' header (unless you've disabled this) and an 'Auto-Submitted' header on outgoing email, but some autoresponders ignore it.
  There is no bulletproof way of detecting whether an email is a bounce or autoreply. The following rules (in procmail format) will detect most autoreplies:

```
^From:.*mailer-daemon@
^Auto-Submitted:.auto-
^Content-Type:\ multipart/report;\ report-type=delivery-status
^Subject:\ Delivery\ Status\ Notification
^Subject:\ Undeliverable
^Subject: Returned Mail:
^From:\ System\ Administrator
^Precedence:\ auto_reply
^Subject:.*autoreply
^Subject:.*Account\ signup
```

  Even with these rules, you may encounter autoreplies with nothing in the headers to distinguish it from a regular mail, In these cases you will just need to manually update the filters to exclude that sender.

- Set up a filter to catch email with huge attachments. JIRA uses the standard JavaMail library to parse email, and it quickly runs out of memory on large attachments (eg. > 50Mb given 512Mb heap). As the unhandled mail isn't deleted, it will be reprocessed (causing another OutOfMemoryError) each time the mail service runs.
  In practice this problem is rarely seen, because most mail servers are configured to not accept email with huge attachments. Unless you're sure yours won't pass a huge attachment on to JIRA, it is best to configure a filter to prevent JIRA encountering any huge attachments.
- Set up spam filtering rules, so JIRA is not having to process (and possibly create issues from) spam.

Troubleshooting

A useful tip for debugging mail-related problems in JIRA is to set the `-Dmail.debug=true` property on startup. This will cause protocol-level details of JIRA's email interactions to be logged. Additionally, turning up JIRA's log level will show when the service is running and how mails are processed.

If you find some incoming emails simply disappear, check that you haven't accidentally started a second copy of JIRA (eg. in a staging environment) which is downloading and deleting mails. See the Restoring Data page for flags you should set to prevent mail being processed.

If you receive email with non-ASCII attachment names, particularly from Thunderbird users, you will need to configure JavaMail to support RFC 2231-encoded attachments. See JRA-12525 for details.

Additional Resources

- Creating comments and issues via email tutorial video — Watch this short tutorial video to see how to create a comments and issues in JIRA via email. Please note the JIRA version and JIRA edition of the tutorial video before watching.

Configuring JIRA to Send SMTP Mail

To enable JIRA to send notifications about various events, you need to first configure JIRA to send SMTP email.

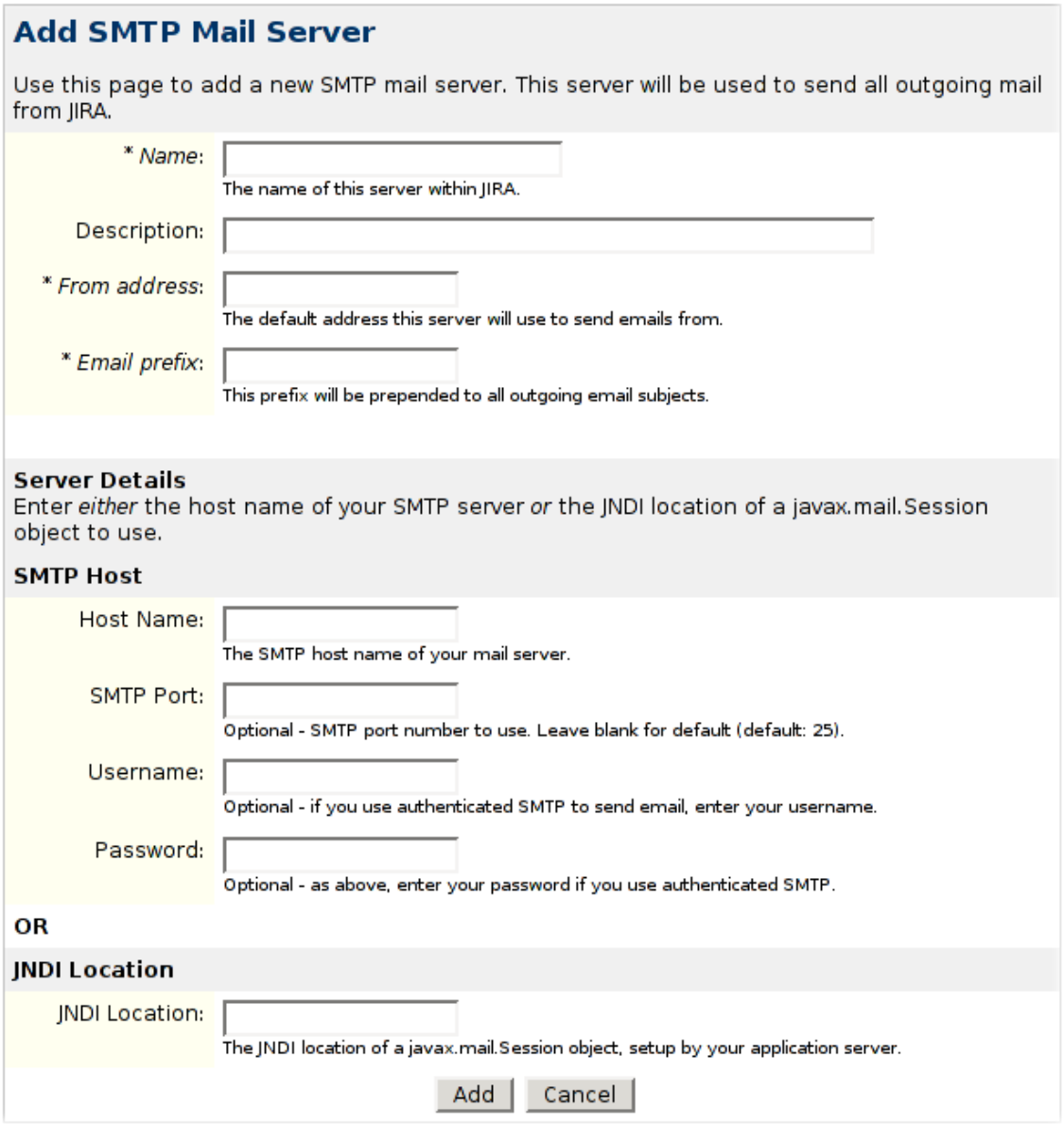
On this page:

- 1. Define the SMTP Mail Server
- 2. Specify the Host Name or JNDI Location
    - To specify the Host Name,
    - To specify and configure a JNDI Location,
        - SMTP over SSL

1. Define the SMTP Mail Server

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click "Mail Servers" in the left-hand column (under "Global Settings").
4. Click "Configure new SMTP mail server".
5. This will display the "Add SMTP Mail Server" screen. Complete the top section as follows:

Name	An arbitrary name to associate with this email server configuration
Description	(Optional) Email server description
From address	The email address that outgoing mails will appear to have come from (unless overridden per project in JIRA Enterprise). Note that this is just the address part ("jira@company.com"). JIRA will use it in constructing the full From header based on the current user ("Joe Bloggs (JIRA) <jira@company.com>"). To change the From header, go to the "Administration" menu, select "General Configuration" (under "Global Settings") and edit the **Email From Header** field.
Email prefix	The subject of emails sent from this server will use this string as a prefix. This is useful for your users so that they can filter their email.



2. Specify the Host Name or JNDI Location

The second part of the screen specifies the **Server Details** of the SMTP server to which JIRA will send mail. There are two ways you can do this. Either:

- specify the **Host Name** of your mail server;
  or:
- specify the **JNDI Location** - that is, use JNDI to look up a mail server that you have preconfigured in your application server. This has the following advantages:
  - Better security: the mail details are not available to JIRA administrators through the JIRA administration interface, and are not stored in JIRA backup files.
  - More SMTP options: if you want to use SMTP over **SSL** (see below), you will need to use JNDI.
  - Centralised management: mail details are configured in the same place as database details, and may be configured through your application server administration tools.

To specify the Host Name,

Most people configure SMTP details directly in JIRA. The form fields are as follows:

Host Name	Hostname or IP address of your SMTP server. Eg. `mail.yourcompany.com`

SMTP Port	The SMTP port, usually 25
Username	Username to connect as, if your SMTP host requires authentication. (Most company servers require authentication to relay mail to non-local users.)
Password	Password for username (if required by your SMTP host).

If your server's startup script uses the **"-Dmail"** system properties (e.g. **"mail.smtp.host"** or **"mail.smtp.port"**), they will override the settings that you specify in the above form. Additionally, if necessary you can manually specify the host name that JIRA reports itself as to the SMTP server by setting **-Dmail.smtp.localhost**

Once done, click 'Update' and then "Send a Test Email" to test the connection details.

To specify and configure a JNDI Location,

As an alternative to specifying mail details directly in JIRA, you can configure them in your application server, and then look up a preconfigured mail session via JNDI.

Complete the following form field

JNDI Location	The JNDI location of a javax.mail.MailSession object to use when sending email.

The **JNDI Location** will depend on your application server and configuration. For example, in Tomcat 5.5 (the default application server that is bundled with JIRA Standalone), your **JNDI Location** would be **java:comp/env/mail/JiraMailServer,** and you would add the following section in `conf/server.xml` , inside the <Context> node:

```
<Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
  ....

  <Resource name="mail/JiraMailServer"
    auth="Container"
    type="javax.mail.Session"
    mail.smtp.host="mail.yourcompany.com"
    mail.smtp.port="25"
    mail.transport.protocol="smtp"
    mail.smtp.auth="true"
    mail.smtp.user="jirauser"
    password="mypassword"
    />
    </Context>
```

Or if you don't require authentication (e.g. if you are sending via localhost, or only internally within the company):

```
<Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
  ....

  <Resource name="mail/JiraMailServer"
    auth="Container"
    type="javax.mail.Session"
    mail.smtp.host="localhost"
    mail.smtp.port="25"
    mail.transport.protocol="smtp"
    />
    </Context>
```

The format for other application servers will be similar. For details please see the Transaction Factory documentation.

If you have problems connecting, add a **mail.debug="true"** parameter, which will let you see SMTP-level details when testing the connection.

You will also need to ensure that the JavaMail classes are present in your application server's classpath, and do not conflict with JIRA's copy.

Most J2EE application servers (eg. **JBoss, Orion, Weblogic, Websphere)** come with JavaMail, and this may conflict with JIRA's copy, resulting in errors like:

```
java.lang.NoClassDefFoundError: javax/mail/Authenticator
```

or:

```
java.lang.IllegalArgumentException: Mail server at location [java:comp/env/mail/JiraMailServer] is not
          of required type javax.mail.Session.
```

To fix this, **remove** `WEB-INF/lib/javamail-1.3.2.jar` and `WEB-INF/lib/activation-1.0.2.jar` from the JIRA webapp.

Lighter app servers **(Tomcat, Resin, Jetty (but not JettyPlus))** do not come with JavaMail. For these, you should **move** `WEB-INF/lib/javamail-1.3.2.jar` and `WEB-INF/lib/activation-1.0.2.jar` into the application server's lib/ directory, eg. `common/lib/` for Tomcat. This is necessary because the application server is establishing the SMTP connection, not JIRA, and the application server won't see the jars in JIRA's classloader.

SMTP over SSL

You can encrypt email communications between JIRA and your mail server via SSL, provided your mail server supports SSL.

To do this, edit your mail server connection properties and specify `starttls` and `SSLSocketFactory` , e.g.:

```
  <Resource name="mail/GmailSmtpServer"
    auth="Container"
    type="javax.mail.Session"
    mail.smtp.host="smtp.gmail.com"
    mail.smtp.port="465"
    mail.smtp.auth="true"
    mail.smtp.user="myusername@gmail.com"
    password="mypassword"
    mail.smtp.starttls.enable="true"
    mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
  />
```

Please note that there is a known bug in some versions of Tomcat 5.5.x (please see JRA-12180).

Additionally, as you are connecting to an SSL service, you will need to **import the SMTP server certificate** into a Java keystore. The process is described on the Connecting to SSL Services page.

For example, on Linux, you could import a certificate as follows:

```
$JAVA_HOME/jre/bin/keytool -import -alias jiramailserver -keystore ~/.keystore  -file
/etc/exim/exim.cert

Enter keystore password:  changeit
Owner: O=Atlassian, L=Sydney, ST=NSW, C=AU
Issuer: O=Atlassian, L=Sydney, ST=NSW, C=AU
Serial number: 0
Valid from: Wed Dec 29 13:02:52 EST 2004 until: Sat May 15 12:02:52 EST 2032
Certificate fingerprints:
         MD5:  91:EC:6E:EA:73:7A:7C:4F:88:92:A2:A0:2B:F7:BC:CC
         SHA1: D8:7C:09:8A:8D:D8:7D:59:C2:28:2A:09:85:90:82:46:78:06:38:D5
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

You would also need to tell Tomcat where the keystore file is located by adding the following to `bin/setenv.sh` :

```
export JAVA_OPTS="-Djavax.net.ssl.trustStore=$HOME/.keystore"
```

Migrating from Other Issue Trackers

When migrating from other issue trackers, you may wish to take your data with you. Depending on what issue tracker you are migrating from, you are recommended to use one of the methods listed below to import data into JIRA:

Whichever method you use, you will need to reindex JIRA after you have imported.

If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

1. Built-in importers

JIRA ships with built-in importers for Importing Data from Bugzilla, Importing Data From Mantis and Importing Data From FogBugz. You are recommended to use the relevant built-in importer, if you are migrating from one of these issue tracking systems.

2. CSV Importer

If you are migrating from a system which JIRA does not provide a built-in importer for, you may be able to import your data into JIRA via CSV format instead. Your system must be able to export your data into a CSV (comma-separated value) file. You can then import the CSV file into JIRA using JIRA's CSV importer. There is also a workaround for importing comments.

3. Third-party scripts

A number of third-party scripts are available on the Internet that support the importing of data into JIRA. These may be a better option for importing data than using the CSV importer.
Third-party scripts are currently available for the following import processes:

- Importing data from Trac into JIRA
- Migrating Trac to JIRA
- yet another Trac 2 JIRA import

Please note, third-party scripts are not supported by Atlassian.
The Atlassian forums may provide helpful information on the third-party scripts that you wish to use.

4. Jelly script

Another approach is to write a Jelly script that will import your data. JIRA ships with some Jelly tags that make operations like creating issues in JIRA easy.

5. RPC services

JIRA ships with an RPC plugin which enables limited remote access to JIRA. It is available through SOAP and XML-RPC interfaces. We recommend using the SOAP interface when possible as it is more complete and will be our primary focus in the future. This page provides a starting point for all your remote procedure call needs.

You can find the latest news on the RPC plugin in the JIRA Extensions space. The full source of the plugin is available and you are free to modify and the extend the source. We'd also be happy to accept code contributions to the project, as Simon Mittag has done in the past. Check out the RPC Endpoint Plugin Module for more information.

6. Your own method

It is possible to use whatever tools you feel comfortable with, to import the data directly into JIRA's database. JIRA's database schema is described in XML format in the `WEB-INF/classes/entitydefs/entitymodel.xml` file under the JIRA web application. When using this approach please take care to maintain database integrity.

See also:

- ClearQuest Import Forums Discussion
- Comparison of JIRA with other issue trackers
- Commercial migrations by Atlassian Partners. A number of partners (Consulting Toolsmiths and others) have provided custom migrations from Remedy, TeamTrack, ClearQuest, GNATS and Bugzilla in the past.

Importing Data From CSV

The CSV importer provides a powerful and flexible way to import data from a comma-separated file, which is a format supported by most applications (e.g. Microsoft Excel).

> Please note that there a number of import methods available for importing data into JIRA from other issue tracking systems. It may be more appropriate to use a method other than the CSV importer, depending on what system you are importing data from. Details on other methods of importing data are available here.

Importing from a CSV file is a three step process. First, you need to prepare and verify your CSV file. Next, create a mapping file by running the CSV import wizard. The mapping file is a plain text properties file that you can also manually edit. It will map your CSV fields to fields in JIRA. Finally, to perform the import, simply enter the location of your import file and your configuration file.

- 1. Preparing your CSV file
  - Valid file format
  - CSV file encoding
  - Importing Comments from CSV
- 2. Running the CSV Import Configuration Wizard
  - 2.1 Project Configuration
  - 2.2 Issue Field Mappings
    - System Fields
    - Time Tracking Fields
    - Custom Fields and the importer
  - 2.3 Map Field Values
    - Value Mappings
  - 2.4 Miscellaneous Information
  - 2.5 Saving the Configuration File
- 3. Importing the CSV file
- Known Issues

Note: Before you begin, please backup your JIRA data.

1. Preparing your CSV file

The first thing you need to do is to ensure that your CSV is a valid CSV format. A good way to check is to import your file into a spreadsheet (e.g. Microsoft Excel, Open Office) and see if the data is as expected. This is also a good opportunity to do any massaging of the data, if you wish.

If you have values that signify a blank value (e.g. <blank> or None), it's best if you simply remove them in this step.

For *built-in JIRA fields* (e.g. **Fix-for version, Affects version, Component),** if you wish to set more then one value for an issue, you will need to have a value per column in your CSV, with each column given a distinct name. For example:

```
IssueType,Summary,FixVersion_1,FixVersion_2
bug,"First issue",v1,
bug,"Second issue",v2,
bug,"third issue",v1,v2
```

In this example, the third imported issue will have its Fix-for version set to multiple values.

For *custom fields* the situation is different, and multiple values are comma-separated. See below for details.

Valid file format

The CSV importer assumes a Microsoft Excel styled CSV file. Fields are separated by commas, and enclosed in quotes if they contain commas or quotes. Embedded quotes are doubled.

There are two requirements of the CSV, in addition to being well-formed in general:

- The CSV file **must contain a heading row.** The CSV configuration wizard uses the CSV header row extensively. The header values should **not have any punctuation** (beyond the commas separating records) such as apostrophes or the importer may not work correctly.
- As a minimum, the CSV file must contain a column for **Summary** data.

You can also have multi-lined CSV. For example, here is a valid file with a single record:

```
Summary,Description,Status
Login fails,"This is on
a new line",Open
```

Commas cannot be omitted. For example, this is valid:

```
Summary,Assignee, Reporter, Issue Type, Description, Priority
test, admin, admin, 1, ,
```

... but this is not valid:

```
Summary, Assignee, Reporter, Issue Type, Description, Priority
test, admin, admin, 1
```

CSV file encoding

JIRA will read the CSV file using the **system encoding,** which can be seen in Administration -> System Info. Make sure that you either save the CSV file with this encoding, or set -Dfile.encoding on startup to force the system encoding to be what you're using (utf8 is best).

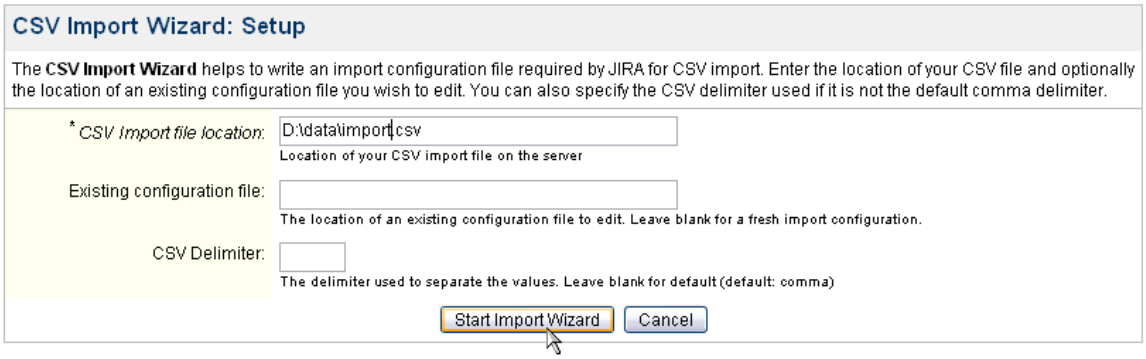
Importing Comments from CSV

If a row contains more columns than there are header columns, then the excess columns will be added as comments (though see JRA-7672).

2. Running the CSV Import Configuration Wizard

The next step of the import process is to run the import configuration wizard to determine how the CSV data can be mapped to JIRA fields.

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Import & Export',** click **'External System Import.** '
4. The **'Import Data'** page will be displayed. Select **'Comma-separated values (CSV)'.**
5. The **'Import issues from CSV file'** page will be displayed. Click the **'CSV Import Wizard'** link.
6. The **'CSV Import Wizard: Setup'** page will be displayed:



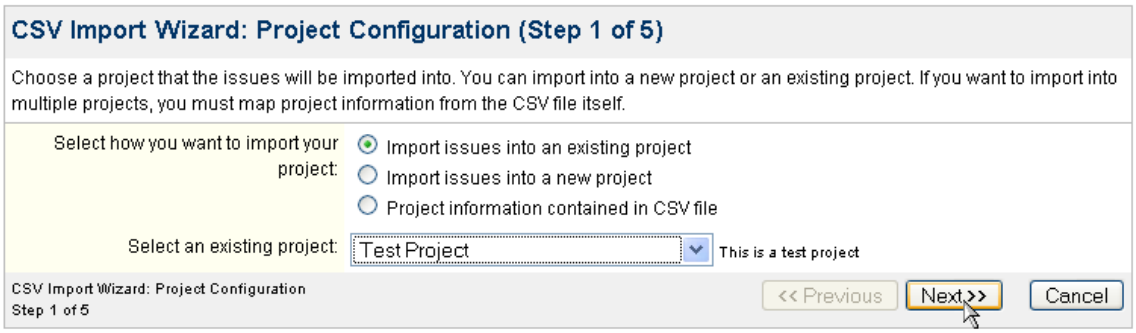
   You can optionally specify the delimiter your CSV file used. Leave the field blank if you wish to use the (default) comma delimiter. Please note that the delimiter can only be one character long.
7. Click the link **'Start Import Wizard.**

2.1 Project Configuration

The first step is to choose which project the issues will be imported into. You can import into a new project or an existing project. If certain project details (e.g. **name** and **key)** match an existing project, then the issues will be imported into an existing project. Note that if you are creating a new project, no validations are performed at this time - invalid data will result in a failure later, at import time.

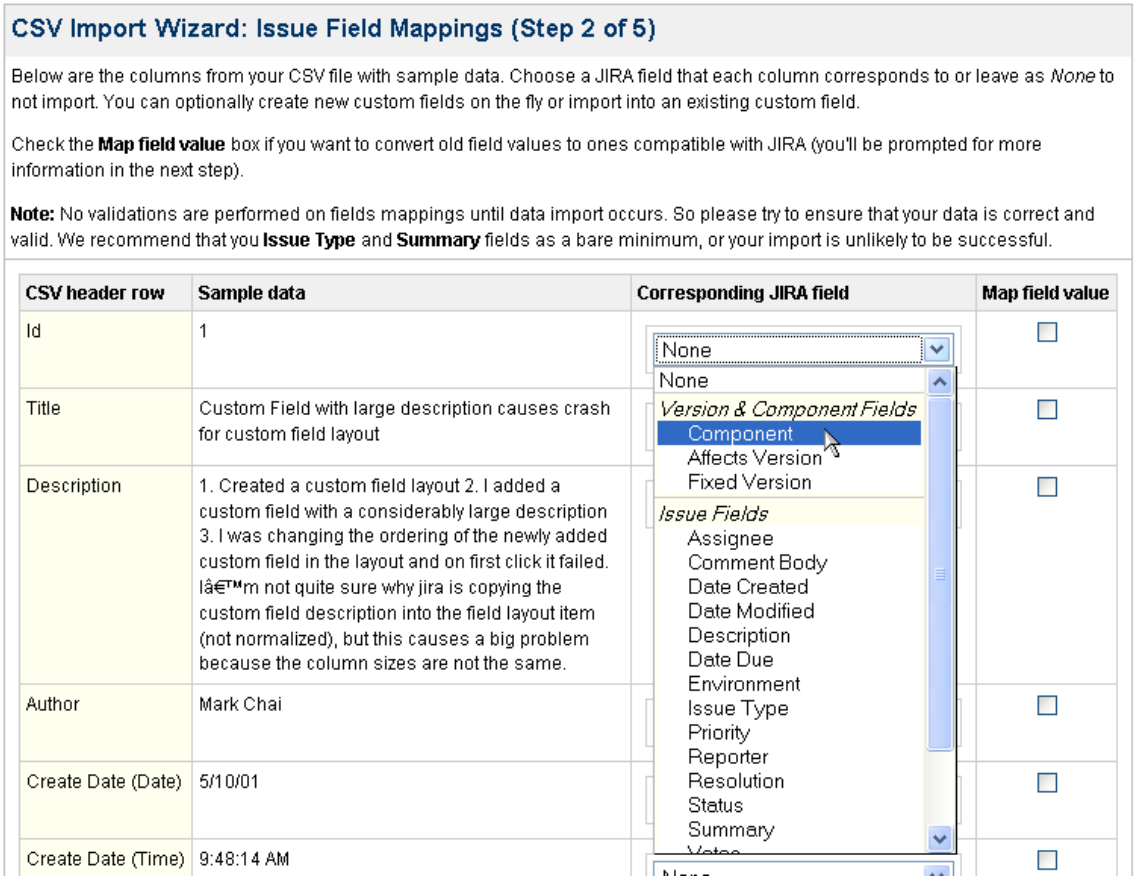
If you want to import into multiple projects, you must map project information from the CSV file itself. That means that all rows must have the project information in them.

The recommended import method is a *single project* per CSV file, imported into an *existing project.*



2.2 Issue Field Mappings

The second step is to decide which CSV fields you want to import. The screen shows all the columns that are found in your CSV file, and a sample data row. On this screen you can map each column of your CSV file to system fields in JIRA, or leave as **None** to not import. You can optionally create new custom fields on the fly or import into an existing custom field.



As no validations are performed on field mappings until data import occurs, please try to ensure that your data is correct and valid.

System Fields

You can select multiple fields to be combined into Version and Component fields. For example, you can import from **'Raised Version'** and **'Found in Version'** into the **'Affects Versions'** field. For Versions and Components, the importer will detect if the version exists in JIRA for the project. If it doesn't exist, then it will automatically created.

User fields **(Assignee** and **Reporter)** are assumed to be in a 'FirstName LastName' format. New users will be created with the username as 'FirstNameLastName'; spaces, apostrophes and brackets are stripped out. If the name only has one word, that one word will be used as the

username.

> If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import.

> If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

In most cases when importing system fields values like **Priority, >Issue Type, Status** and **Resolution,** you will need to JIRADOC:map the field values. The mapping needs to be done even if the imported CSV file has the values set to 'valid' names, e.g. **Issue Type** set to 'Bug' or 'New Feature'. The only alternative to mapping the values is to change the CSV file such that it contains the exact IDs of JIRA's **priorities, issue types, statuses** and **resolutions** instead of their names. This requires you to determine the correct IDs and then update the whole CSV file, so it is easier to map the values during the import.

Time Tracking Fields

As of JIRA 3.7 you can now import into the 'Original Estimate', 'Remaining Estimate', and 'Time Spent' fields. For these fields to be available you must have enabled time tracking within JIRA. The importer expects the estimate or time spent to be expressed in seconds. A value that is not able to be converted to a numeric value will be ignored and not imported.

The portion of the importer that converts the raw string to a java.lang.Long which represents number of seconds is customizable. If you are trying to import data that needs to more intelligently process the value (more than just converting the string to a numeric value) you can write your own java class. It needs to implement the `com.atlassian.jira.imports.csv.mapper.TimeEstimateConverter` interface and you can direct the importer to use your class by specifying in the csv.properties configuration file the '`settings.advanced.mapper.time.estimate.converter`' property (i.e. `settings.advanced.mapper.time.estimate.converter=com.atlassian.jira.imports.csv.mappers.SimpleTimeEstimateCo` `com.atlassian.jira.imports.csv.mappers.SimpleTimeEstimateConverter`)

Custom Fields and the importer

You can also map a column to an existing custom field or create a new custom field on the fly. Currently you can only create certain custom fields on the fly. All custom fields created this way will be globally scoped. Moreover, if the name matches an existing custom field, that existing custom field will be used instead. If you are worried about how this works exactly, we recommend that you create your custom fields first before importing them.

CSV header row	Sample data	Corresponding JIRA field
Id	1	New custom field Custom field name: New ID Custom field type: Text Field
Title	Custom Field with large description causes crash for custom field layout	None
Description	1. Created a custom field layout 2. I added a custom field with a	

If you map to a select list custom field, all unique values will be created as options at import time. If you map to a multiple select field, its values should be separated by a comma. If the field values have commas in them, the commas should be escaped with a backslash. Thus the field:

```
"Wally\, I,Wally\, II"
```

would be translated into one field with multiple values:

- Wally I
- Wally II

Once again, no data validations are done at configuration time, so you should ensure that the data you are trying to import is of a compatible type.

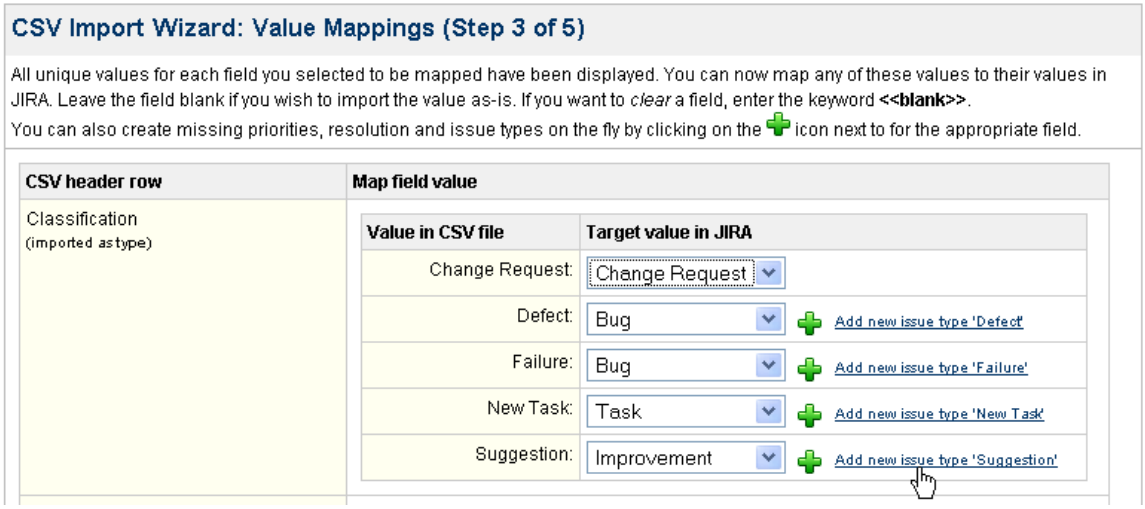
2.3 Map Field Values

You may wish to map certain values in your CSV file to a different value. For example, you might map the field **'Severity'** to JIRA's **'Priority'** field. JIRA expects the ID of priorities that exist in JIRA. Thus for this field, you'll need to check the **Map field value** check-box. This will affect the next screen that you will come to.

Value Mappings

Value mappings determine how values from your CSV importer will be 'translated' to match the values expected by JIRA. This is usually required for fields such as **'Issue Type', 'Resolution', 'Priority'** and **'Status',** but can also apply to other fields. On this screen, all unique values for each field you selected to be mapped have been displayed. You can now map any of these values to their values in JIRA. Leave the field blank if you wish to import the value as-is. If you want to clear a field, enter the keyword **<<blank>>.**

On the **'Field Mappings'** screen, each field has a checkbox under the heading **'Map values'.** If you check these boxes you will be able to map the values of these fields when you progress to the next page.



For fields mapping to **Resolution, Priority** and **Issue Type,** you will get a select list with the available values in JIRA. In addition you can quickly create values that do not exist in JIRA by clicking the green plus symbols. These will become issue constants that you can change at a later time.

For fields mapping to **Status,** you will get the select list with JIRA's available values, but no plus symbol for creating new status values.

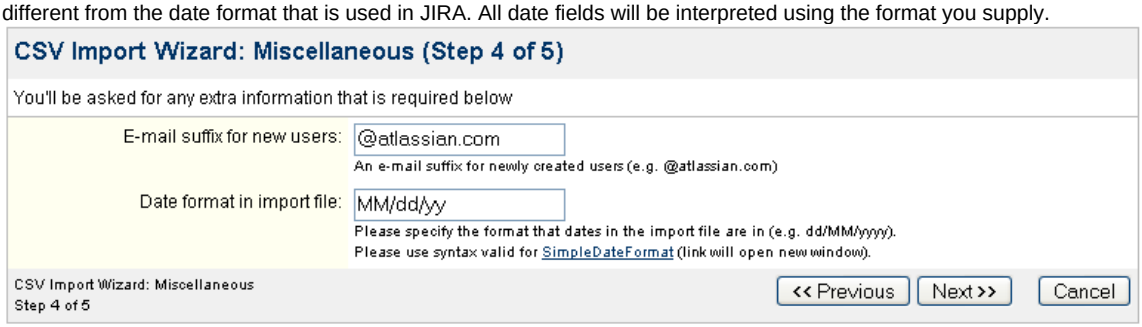
For these fields, there are two special options in the select list in addition to JIRA's available values, **'Import as blank'** and **'No mapping'. 'Import as blank'** causes the JIRA value to be blank for that field. **'No mapping'** attempts to import the value in the CSV file as-is. Note that using **'No mapping'** for a field value will result in a failed import if the value is not valid for that JIRA field.

For fields mapping to **Status** and **Issue Type,** default values are used when the **blank** option is selected.

2.4 Miscellaneous Information

You will be asked to enter some extra information on this screen, such as:

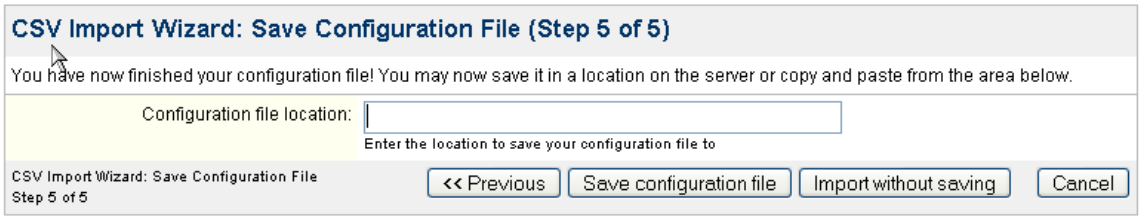
1. The domain name of the users that will be created in the system.
2. If you are importing date fields, you will also be asked to supply the date format that is used in your CSV file. Note that this could be different from the date format that is used in JIRA. All date fields will be interpreted using the format you supply.



2.5 Saving the Configuration File

The final step of the Wizard allows you to save the configuration file on your server. Saving the configuration file enables you to import more CSV files later without going through the Configuration Wizard again. Please ensure you enter a valid path. Alternatively, you can choose to continue on with the import without saving the configuration in a file.

You can also see a preview of the mapping file that will be saved.



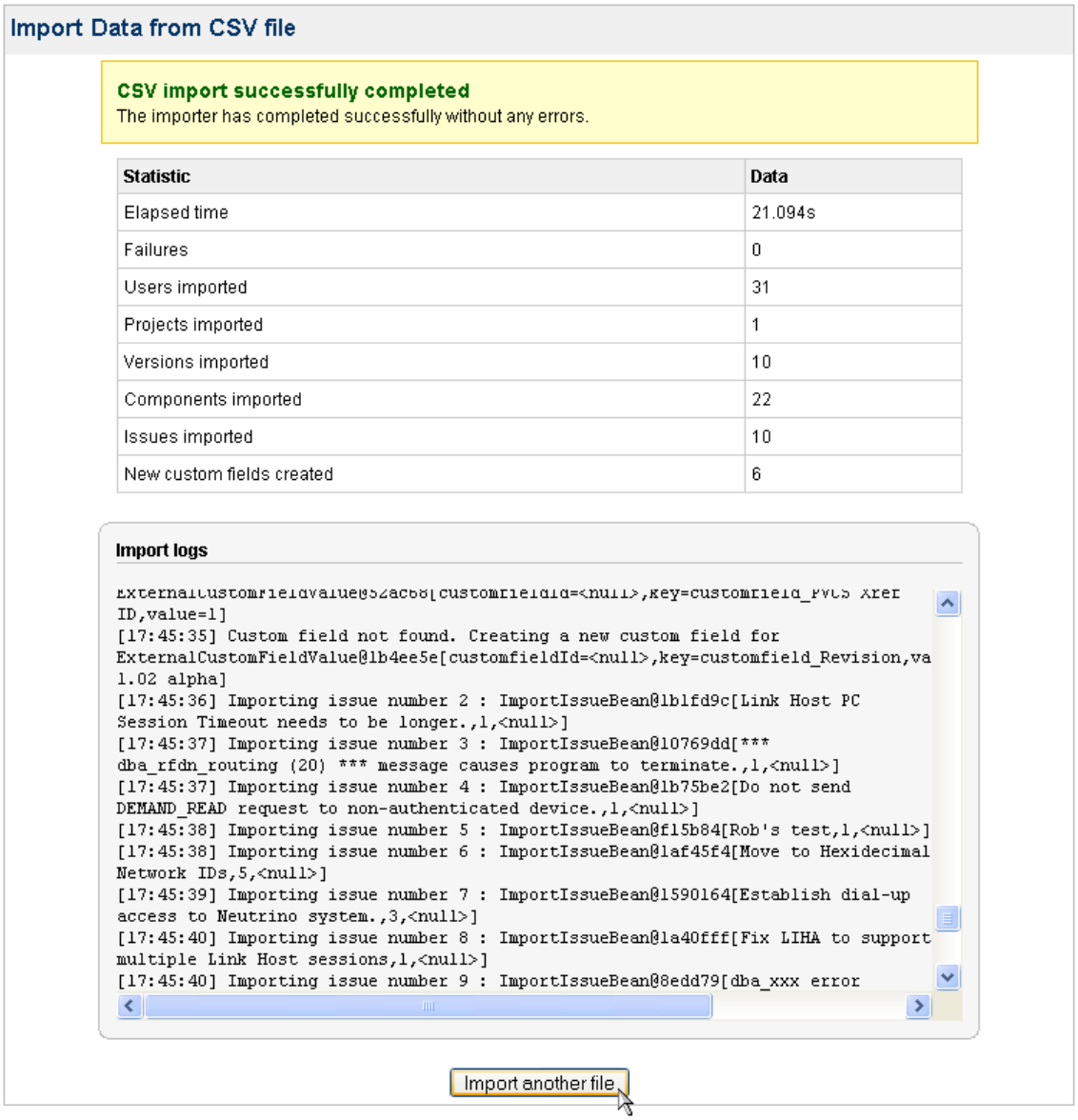
3. Importing the CSV file

Once you have your configuration file, you can then import the CSV file into JIRA.

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Import & Export',** click **'External System Import.** '
4. The **'Import Data'** page will be displayed. Select **'Comma-separated values (CSV)'.**
5. The **'Import issues from CSV file'** page will be displayed.
6. Type the location of your CSV file and your configuration file, and click the **'Import'** button.

The **'Settings'** page gives you precise control over what will be imported on each import run.

Once the import has begun you will be able to follow the progress of the import, with the screen refreshing around every 10 seconds. You can change this rate by updating the field at the bottom of the page. The importer also give you statistics about what objects have been imported and time elapsed so you can have an idea as approximately how long the import will take. You can also choose to **'Abort'** the import, which will cease importing after the current issue is done.



Known Issues

- The CSV Importer doesn't import all the objects available in JIRA at present. You can find these limitations at issue JRA-5774.
- There is a known problem that prevents the CSV Importer from being used with JIRA instances running on JBoss 4.x. This is due to a compatibility issue between the JBoss 4.x commons-collections.jar and the JIRA commons-collections.jar. The workaround is to replace the commons-collections.jar in JBoss 4.x with the more recent JIRA version. Please see JRA-6473 for further details.

Importing Data from Bugzilla

On this page:

- Importing Data from Bugzilla
  - How to import from Bugzilla
  - Importing only new bugs (repeated imports)
  - Searching by Bugzilla ID
  - Importer Source code
  - Common Customisations
    - Importing custom priorities, statuses and resolutions
    - Changing the imported username format

Importing Data from Bugzilla

JIRA can import your bugs from Bugzilla. Currently, the importer is compatible with Bugzilla 2.20 and above. Users of older Bugzilla versions will

need to first upgrade the database tables to a supported version with Bugzilla's checksetup.pl script.

> Due to a change to the database schema in Bugzilla versions 2.22.2 and later, attachments are not imported into JIRA versions prior to JIRA 3.13.x. For details please see JRA-12389. Please upgrade to JIRA 3.13+ if this is a problem for you.

The data from the Bugzilla database is appended to the existing data in JIRA. The tool imports the following data from the Bugzilla database:

In Bugzilla	In JIRA	Special Notes
Bugs	Issues	• Attachments are extracted from the Bugzilla database and saved to disk. • Statuses, Bug Severity, Issue Types, and Resolutions in Bugzilla are mapped to the defaults in JIRA. • Statuses in Bugzilla are mapped to JIRA. Bugs in Bugzilla in the 'NEW', 'UNCONFIRMED', or 'ASSIGNED' status with no assignees are 'Open' in JIRA. • Issue Types of bugs from Bugzilla are all 'Bugs' or 'Improvements'. • Bug_Severity in Bugzilla is mapped to Priorities in JIRA. Bugs with 'ENHANCEMENT' severity in Bugzilla are treated as 'Improvement' issues with 'MINOR' Priority in JIRA. Note: if you have customized the Bugzilla list of priorities, you will need to edit the Importer source and define the new mappings. • The first description for a bug in Bugzilla is stored as JIRA's Description. All other descriptions are stored as comments logged to that issue. • If a user has voted one or more times for a Bugzilla issue, a JIRA vote is stored for that user.
Product	Project	• The project key and project lead can be set by the user.
Version	Version	• Versions for imported projects are imported from Bugzilla, and set to Un-Released and Un-Archived state. • The JIRA "Fix For" Version is set to the Bugzilla bug 'milestone', if it exists. Note: this code is not well tested - please let us know if you have problems.
Component	Component	
User	User	• Users are imported 'on demand', so users who have not interacted with the system in any way are not imported. • Passwords from Bugzilla are not imported for v2.16+ of Bugzilla (as they are hashed in the database). Users from Bugzilla will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name stored in Bugzilla will get the portion of their email address (login name) before the "@" character as their Full Name in JIRA. If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import (this way, votes etc can be imported correctly). If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

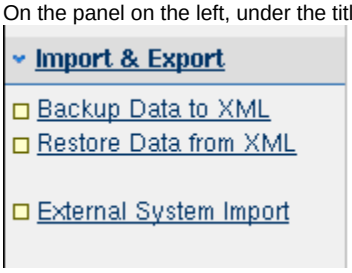
How to import from Bugzilla

Note: Before you begin, please backup your JIRA data.

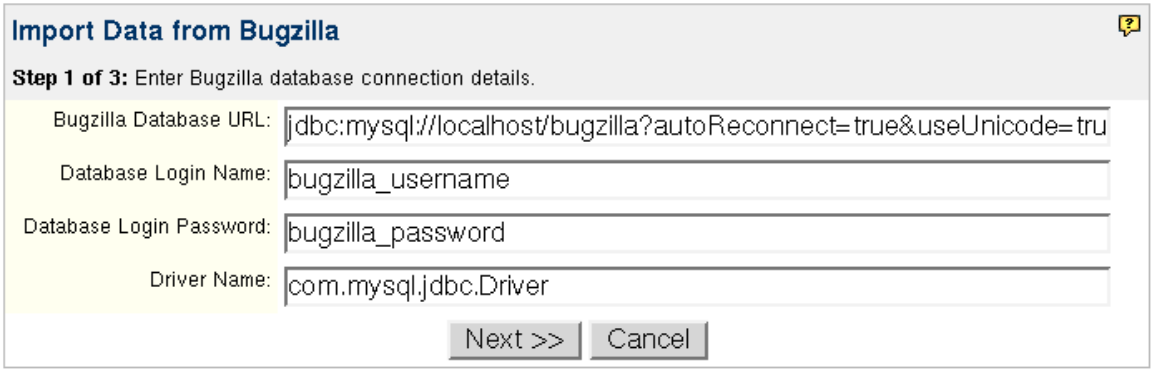
1. In your Bugzilla system, run the Bugzilla 'Sanity Check' to ensure your data is error-free.
2. Make sure that the Bugzilla database you wish to import from is running on MySQL.
3. Download and install the MySQL JDBC driver into JIRA. To do this, download the MySQL Connector/J driver. The package contains a file "mysql-connector-java-xxx.jar". Copy this to the `common/lib/` directory in JIRA Standalone, or equivalent "lib" directory in your app server. Restart JIRA so the driver is loaded.
4. In JIRA's default permission scheme (associated with newly imported projects), ensure that the 'Browse', 'Create' and 'Comment' permissions are granted to the group 'jira-users' (or a group with the 'JIRA Users' global permission).
5. Log in as a user with the **'JIRA System Administrators'** global permission.
6. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the

dashboard.

7. On the panel on the left, under the title 'Import & Export', click 'External System Import'



8. The 'Import Data' page will be displayed. Select 'Bugzilla'.
9. You will now be prompted for connection details to Bugzilla's MySQL database:

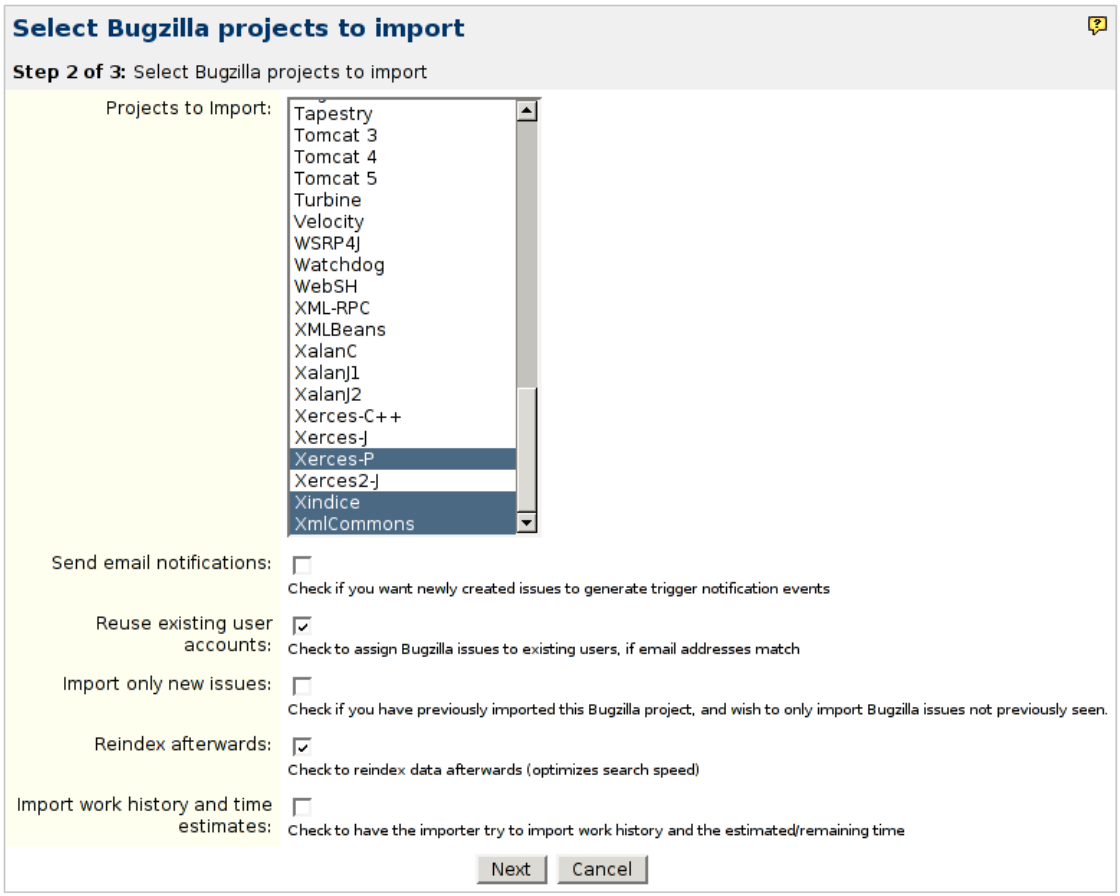


Set the JDBC URL, database username and password for your system. The JDBC URL is of the format:

```
jdbc:mysql:host[:port]/databasename?parameters
```

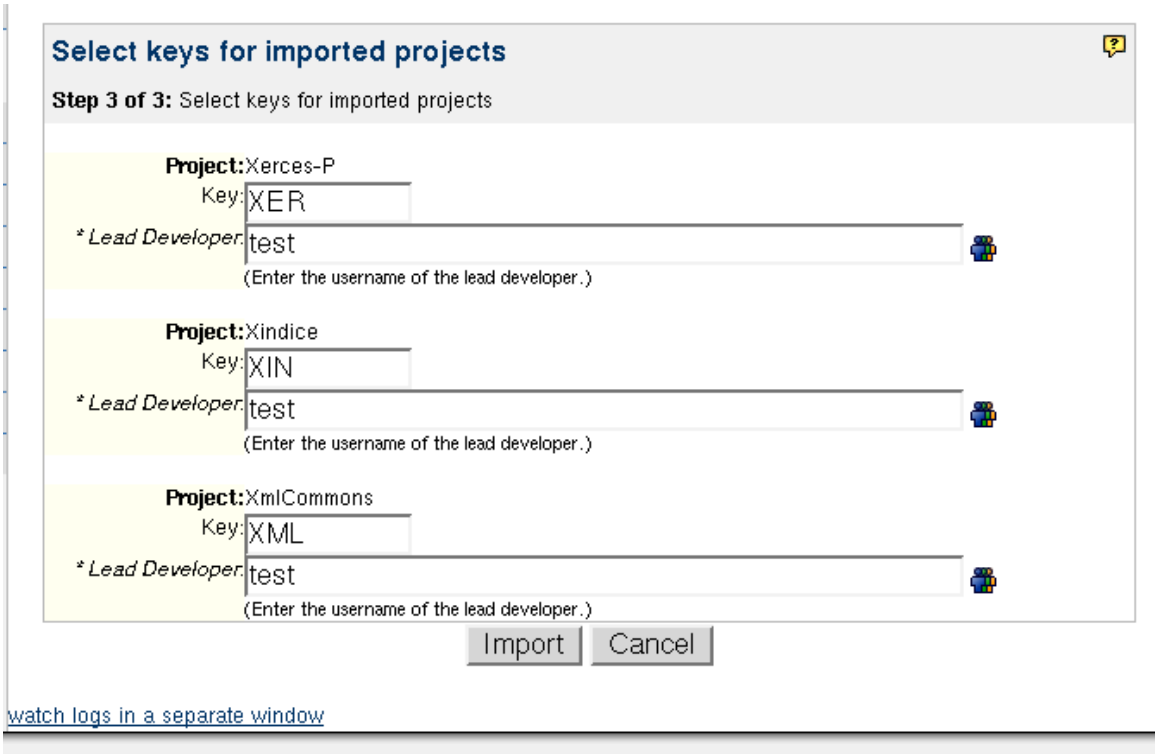
host is the server hosting JIRA, whose MySQL allows incoming TCP connections on port **port** (defaults to 3306). **databasename** is the MySQL database name (usually 'bugs').The database name, username and user password can usually be found in the 'localconfig' file in Bugzilla's root directory, or in /etc/bugzilla/.

10. Click 'Next >>' to advance to the project selection page. If you have got the connection details wrong, you may have to wait about 30s for the connection attempt to time out.
11. You will now be presented with a list of projects in Bugzilla:



Select the projects you wish to import (even if there is only one - select it!), and set import options via the checkboxes (the default checkbox settings is correct for most users). Click 'Next >>' to proceed.

12. You will now be prompted to choose a project key (the per-project prefix to attach to bug keys), and a project lead for each project to be imported. **Important:** do not use dashes or numbers in the project key!

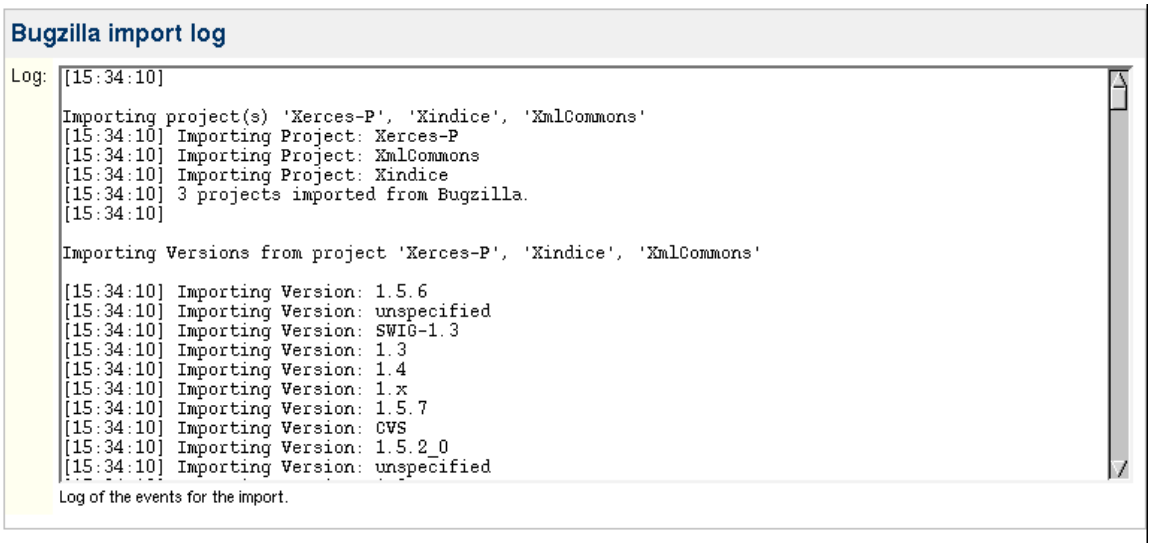


Click 'Import' when you are done to start the import. As imports frequently take a long time, you can watch the logs as they are generated by clicking the 'watch logs in separate window' link. Logs are also sent to stdout, and will appear in your app server's log:

```
Importing project(s) 'Xerces-P', 'Xindice', 'XmlCommons'
2004-02-27 15:34:10,031 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project:
Xerces-P
2004-02-27 15:34:10,052 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project:
XmlCommons
2004-02-27 15:34:10,061 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project: Xindice
2004-02-27 15:34:10,069 INFO [atlassian.jira.util.BugzillaImportBean] 3 projects imported from
Bugzilla.
2004-02-27 15:34:10,069 INFO [atlassian.jira.util.BugzillaImportBean]

Importing Versions from project 'Xerces-P', 'Xindice', 'XmlCommons'

2004-02-27 15:34:10,071 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version: 1.5.6
2004-02-27 15:34:10,080 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version:
unspecified
2004-02-27 15:34:10,087 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version:
SWIG-1.3
2004-02-27 15:34:10,093 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version: 1.3
2004-02-27 15:34:10,098 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version: 1.4
...
2004-02-27 15:34:10,228 INFO [atlassian.jira.util.BugzillaImportBean] Importing Component: SWIG
2004-02-27 15:34:10,292 INFO [atlassian.jira.util.BugzillaImportBean] Importing Component: Test
Scripts
2004-02-27 15:34:10,346 INFO [atlassian.jira.util.BugzillaImportBean] Importing Component:
Documentation
...
Importing Issues from project(s) 'Xerces-P', 'Xindice', 'XmlCommons'
2004-02-27 15:34:11,341 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue: "All
files
need Apache license"
2004-02-27 15:34:11,343 INFO [atlassian.jira.util.BugzillaImportBean] Importing User:
jason1@openinformatics.com
2004-02-27 15:34:16,430 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue: "SEGV on:
getAttributes() in list context"
2004-02-27 15:34:16,545 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue: "Sample
code
must be updated"
2004-02-27 15:34:16,680 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue: "SWIG 1.3
rohibits sub-classing of wrapped classes"
2004-02-27 15:34:16,878 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
"AttributeList in
Perl is broken (can't get attributes)"
2004-02-27 15:34:16,909 INFO [atlassian.jira.util.BugzillaImportBean] Importing User:
dgrey@internap.com
....
2004-02-27 15:35:09,364 INFO [atlassian.jira.util.BugzillaImportBean] Importing User:
ilm@doc.ic.ac.uk
2004-02-27 15:35:09,580 INFO [atlassian.jira.util.BugzillaImportBean] 144 issues imported from
Bugzilla.
2004-02-27 15:35:09,581 INFO [atlassian.jira.util.BugzillaImportBean]

Importing Votes
2004-02-27 15:35:09,747 INFO [atlassian.jira.util.BugzillaImportBean] 0 votes imported from
Bugzilla.
2004-02-27 15:35:09,748 INFO [atlassian.jira.util.BugzillaImportBean] Reindexing (this may take
a while)...
2004-02-27 15:35:11,282 WARN [jira.issue.index.AbstractDocument] Unable to index field 'created'
from ANT-4
with value: 0002-11-30 00:00:00.0
2004-02-27 15:35:29,170 INFO [atlassian.jira.util.BugzillaImportBean]
Import SUCCESS and took: 79142 ms.
```

It does not matter if your browser window times out - the import will continue regardless.

13. Once completed, you will see the 'Import SUCCESS' message, either in the logs (as above) or on the subsequent page:



Congratulations, you have successfully imported your Bugzilla projects into JIRA!If you have any questions or encounter any problems, please contact Atlassian.

Importing only new bugs (repeated imports)

It is possible to re-import a Bugzilla project, and have JIRA import only 'new' bugs not previously imported. This allows for a transition period in which the imported JIRA project can be trialled, but bugs still logged in Bugzilla need not be lost.

To import only new bugs, click the '**Import only new issues**' checkbox in the importer.

Searching by Bugzilla ID

The Bugzilla importer creates a 'Bugzilla ID' custom field for imported issues, linking back to the original Bugzilla bug URL.

If you intend to use this, you will need to configure the URL to Bugzilla in jira-application.properties. The custom field can also be made to display just the ID (unlinked) in jira-application.properties. If you don't need this custom field, delete it or hide it.

The custom field is searchable, so you can search for JIRA issues by their old Bugzilla ID. There is also a portlet which lets you search by Bugzilla ID:

Importer Source code

The Bugzilla importer source code is available here. For some customisations you may wish to make, editing this source code is required, as described below. For users of JIRA Standalone, there is a mini build-system which you can use to quickly compile and test modifications.

Common Customisations

Importing custom priorities, statuses and resolutions

Bugzilla has a standard set of priorities, statuses and resolutions, but these can be augmented with new ones by editing a Bugzilla config file ('localconfig'). If your Bugzilla has custom statuses, JIRA will set the status of affected imported issues to "Open", and log a message ("... defaulting to JIRA status Open"). For other unknown fields (priorities, resolutions) JIRA will just not set the field, which may cause problems later (eg. issues Resolved but without a resolution are listed as open in the standard filters).

To avoid problems, it is a good idea to check whether your Bugzilla uses any custom resolutions, statuses or priorities:

```
mysql> select distinct(priority) from bugs;
+----------+
| priority |
+----------+
| P2       |
| P4       |
| P3       |
| P1       |
+----------+
4 rows in set (0.00 sec)
mysql> select distinct(resolution) from bugs;
+------------+
| resolution |
+------------+
| FIXED      |
| DUPLICATE  |
| WONTFIX    |
|            |
| LATER      |
| INVALID    |
| WORKSFORME |
| REMIND     |
+------------+
8 rows in set (0.00 sec)
mysql> select distinct(bug_status) from bugs;
+-------------+
| bug_status  |
+-------------+
| RESOLVED    |
| CLOSED      |
| NEW         |
| ASSIGNED    |
| REOPENED    |
| VERIFIED    |
| UNCONFIRMED |
+-------------+
7 rows in set (0.00 sec)
```

(the above are all standard). If your bugs use anything non-standard, you will need to edit the mappings in BugzillaImportBean.java:

```
static
        {
            // bugzilla's priorities mapping to JIRA priorities
priorityMap.put("blocker", "" + IssueFieldConstants.BLOCKER_PRIORITY_ID);
            priorityMap.put("critical", "" + IssueFieldConstants.CRITICAL_PRIORITY_ID);
            priorityMap.put("major", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("normal", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("enhancement", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("minor", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("trivial", "" + IssueFieldConstants.TRIVIAL_PRIORITY_ID);

            // bugzilla resolutions mapping to JIRA resolutions
resolutionMap.put("", null);
            resolutionMap.put("FIXED", "" + IssueFieldConstants.FIXED_RESOLUTION_ID);
            resolutionMap.put("INVALID", "" + IssueFieldConstants.INCOMPLETE_RESOLUTION_ID);
            resolutionMap.put("WONTFIX", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("LATER", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("REMIND", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("DUPLICATE", "" + IssueFieldConstants.DUPLICATE_RESOLUTION_ID);
            resolutionMap.put("WORKSFORME", "" + IssueFieldConstants.CANNOTREPRODUCE_RESOLUTION_ID);
            resolutionMap.put("NEEDTESTCASE", "" + IssueFieldConstants.INCOMPLETE_RESOLUTION_ID);

            // bugzilla status mapping to JIRA status
statusMap.put("UNCONFIRMED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("NEW", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("ASSIGNED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("REOPENED", "" + IssueFieldConstants.REOPENED_STATUS_ID);
            statusMap.put("RESOLVED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("VERIFIED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("CLOSED", "" + IssueFieldConstants.CLOSED_STATUS_ID);

            // workflow Mappings
wfStepMap.put("1", new Integer("1"));
            wfStepMap.put("2", new Integer("2"));
            wfStepMap.put("3", new Integer("3"));
            wfStepMap.put("4", new Integer("5"));
            wfStepMap.put("5", new Integer("4"));
            wfStepMap.put("6", new Integer("6"));

            wfStatusMap.put("1", "Open");
            wfStatusMap.put("3", "In Progress");
            wfStatusMap.put("4", "Reopened");
            wfStatusMap.put("5", "Resolved");
            wfStatusMap.put("6", "Closed");
        }
```

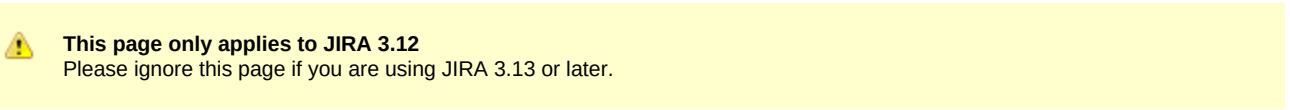
(Note: wfStepMap and wfStatusMap should usually not be touched, unless you are importing into project with a non-standard workflow).

Changing the imported username format

Bugzilla uses email addresses for usernames (eg. "joe@example.com"). You may wish to automatically strip everything after the '@' to form a shortened username ("joe"), or otherwise alter imported names (eg. read from a lookup table to conform to a company-wide standard). This requires editing the BugzillaImportBean source (see above). The relevant code to modify is included in BugzillaImportBean.java, but commented out:

```
/**
     * Given a Bugzilla 'profile' user record, infer a JIRA username from it.
     * In Bugzilla your username is your email address, and this will become your JIRA username,
unless this method
     * is overridden to implement a different scheme.
     */
    protected String getUsernameFromBugzillaProfile(ResultSet bugzillaProfileResultSet)
            throws SQLException
    {
        return TextUtils.noNull(bugzillaProfileResultSet.getString("login_name"
)).toLowerCase().trim();

        // Alternatively, use the first part ('joe' in 'joe@company.com')
//        String name = bugzillaProfileResultSet.getString("login_name");
//        name = TextUtils.noNull(name).trim();
//        int i = name.indexOf("@");
//        if (i != -1) name = name.substring(0, i);
//        return name;
}
```

Migrating from Bugzilla 3.0.3



This page only applies to JIRA 3.12
Please ignore this page if you are using JIRA 3.13 or later.

Due to a change to the database schema in Bugzilla versions 2.22.2 and later, attachments are not imported into JIRA. For details please see JRA-12389.

The instructions below will assist you in migrating attachments from Bugzilla 3.0.3 into JIRA 3.12. These instructions are not supported by Atlassian.

The attached zip file (BugzillaImportBean.zip) includes a patched up `BugzillaImportBean.java` from JIRA 3.12.2. Please move the following files from "`<JIRA installation directory>/atlassian-jira/WEB-INF/classes/com/atlassian/jira/util/`" into a backup directory:

- BugzillaImportBean$1.class
- BugzillaImportBean$BugzillaMappingBean.class
- BugzillaImportBean.class
- BugzillaImportBean$DefaultBugzillaMappingBean.class
- BugzillaImportBean$UserNameCollator.class

Then extract the contents of the attached zip file into this directory.

Based on my simple test against Bugzilla 3.0.3, after the JIRA restart the attachment import functionality should be enabled.

You will need to delete the already imported bugzilla issues and redo the process again in order to get the attachments imported into JIRA.

Modifying the Bugzilla Importer

JIRA comes with a means of importing issues from Bugzilla. Unfortunately there are some aspects of this importer that are hardcoded, and require editing of the code to change. For instance:

Changing hardcoded mappings

(related feature requests: JRA-3701, JRA-6403)

The Bugzilla importer assumes a fixed set of Bugzilla severities, resolutions, and JIRA statuses :

```
static
        {
            // bugzilla's priorities mapping to JIRA priorities
priorityMap.put("blocker", "" + IssueFieldConstants.BLOCKER_PRIORITY_ID);
            priorityMap.put("critical", "" + IssueFieldConstants.CRITICAL_PRIORITY_ID);
            priorityMap.put("major", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("normal", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("enhancement", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("minor", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("trivial", "" + IssueFieldConstants.TRIVIAL_PRIORITY_ID);

            // bugzilla resolutions mapping to JIRA resolutions
resolutionMap.put("", null);
            resolutionMap.put("FIXED", "" + IssueFieldConstants.FIXED_RESOLUTION_ID);
            resolutionMap.put("INVALID", "" + IssueFieldConstants.INCOMPLETE_RESOLUTION_ID);
            resolutionMap.put("WONTFIX", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("LATER", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("REMIND", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("DUPLICATE", "" + IssueFieldConstants.DUPLICATE_RESOLUTION_ID);
            resolutionMap.put("WORKSFORME", "" + IssueFieldConstants.CANNOTREPRODUCE_RESOLUTION_ID);
            resolutionMap.put("NEEDTESTCASE", "" + IssueFieldConstants.INCOMPLETE_RESOLUTION_ID);

            // bugzilla status mapping to JIRA status
statusMap.put("UNCONFIRMED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("NEW", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("ASSIGNED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("REOPENED", "" + IssueFieldConstants.REOPENED_STATUS_ID);
            statusMap.put("RESOLVED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("VERIFIED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("CLOSED", "" + IssueFieldConstants.CLOSED_STATUS_ID);

            // workflow Mappings
wfStepMap.put("1", new Integer("1"));
            wfStepMap.put("2", new Integer("2"));
            wfStepMap.put("3", new Integer("3"));
            wfStepMap.put("4", new Integer("5"));
            wfStepMap.put("5", new Integer("4"));
            wfStepMap.put("6", new Integer("6"));

            wfStatusMap.put("1", "Open");
            wfStatusMap.put("3", "In Progress");
            wfStatusMap.put("4", "Reopened");
            wfStatusMap.put("5", "Resolved");
            wfStatusMap.put("6", "Closed");
        }
```

Changing imported usernames

(related feature request: JRA-3890)

By default, the Bugzilla importer creates usernames which are equal to the email addresses of reporters (which act as the Bugzilla logins). If instead you wish to infer a shorter username from the email (eg. 'joe' from 'joe@company.com'), you can do this by modifying this code:

```
/**
     * Given a Bugzilla 'profile' user record, infer a JIRA username from it.
     * In Bugzilla your username is your email address, and this will become your JIRA username,
unless this method
     * is overridden to implement a different scheme.
     */
    protected String getUsernameFromBugzillaProfile(ResultSet bugzillaProfileResultSet)
            throws SQLException
    {
        return TextUtils.noNull(bugzillaProfileResultSet.getString("login_name"
)).toLowerCase().trim();

        // Alternatively, use the first part ('joe' in 'joe@company.com')
//        String name = bugzillaProfileResultSet.getString("login_name");
//        name = TextUtils.noNull(name).trim();
//        int i = name.indexOf("@");
//        if (i != -1) name = name.substring(0, i);
//        return name;
}
```

Modifying the importer code

Get the importer source.

The Bugzilla importer code can be obtained from the JIRA documentation website (the Importing Data from Bugzilla page), at the bottom. A direct link to the latest source is:

BugzillaImportBean.java

Compile the source

In JIRA Standalone:

1. Create a directory, `external-source/src`
2. Copy BugzillaImportBean.java to `external-source/src`
3. Download and install Apache Ant if you don't already have it.
4. Compile BugzillaImportBean.java into JIRA Standalone.

Please note that you may need to restart JIRA Standalone in order for your changes to be detected.

For example:

```
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone$ mkdir external-source/src
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone$ cd external-source/src
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/src$ wget -q
'http://www.atlassian.com/software/jira/docs/latest/importers/BugzillaImportBean.java'
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/src$ ls
BugzillaImportBean.java
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/src$ cd ..
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source$ ant
Buildfile: build.xml

prepare:
    [mkdir] Created dir: /tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/etc
    [mkdir] Created dir: /tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/lib

compile:
    [javac] Compiling 1 source file to
/tmp/atlassian-jira-enterprise-3.4.2-standalone/atlassian-jira/WEB-INF/classes
    [javac] Note:
/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/src/BugzillaImportBean.java uses or
overrides a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.
    [javac] Note:
/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source/src/BugzillaImportBean.java uses
unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

BUILD SUCCESSFUL
Total time: 3 seconds
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2-standalone/external-source$
```

In JIRA Webapp/WAR edition

If you have the webapp edition of JIRA, copy `BugzillaImportBean.java` into src/, and then run `build.sh`/`build.bat` as normal:

```
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2$ cd src
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2/src$ wget -q
'http://www.atlassian.com/software/jira/docs/latest/importers/BugzillaImportBean.java'
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2/src$ cd ..
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2$ ./build.sh
Buildfile: build.xml

init:

compile:
Created dir: /tmp/atlassian-jira-enterprise-3.4.2/tmp/build/war
Created dir: /tmp/atlassian-jira-enterprise-3.4.2/dist-generic
Created dir: /tmp/atlassian-jira-enterprise-3.4.2/dist-tomcat
Copying 3492 files to /tmp/atlassian-jira-enterprise-3.4.2/tmp/build/war
Copied 2 empty directories to /tmp/atlassian-jira-enterprise-3.4.2/tmp/build/war
Copying 1 file to /tmp/atlassian-jira-enterprise-3.4.2/tmp/build/war
Compiling 1 source file to /tmp/atlassian-jira-enterprise-3.4.2/tmp/build/war/WEB-INF/classes
Note: /tmp/atlassian-jira-enterprise-3.4.2/src/BugzillaImportBean.java uses or overrides a deprecated
API.
Note: Recompile with -Xlint:deprecation for details.
Note: /tmp/atlassian-jira-enterprise-3.4.2/src/BugzillaImportBean.java uses unchecked or unsafe
operations.
Note: Recompile with -Xlint:unchecked for details.

war:
Building jar: /tmp/atlassian-jira-enterprise-3.4.2/dist-generic/atlassian-jira-3.4.2.war
Building jar: /tmp/atlassian-jira-enterprise-3.4.2/dist-tomcat/atlassian-jira-3.4.2.war
Copying 1 file to /tmp/atlassian-jira-enterprise-3.4.2/dist-tomcat

-------------------------------------------------
Deployable WARs generated:

TOMCAT:  /tmp/atlassian-jira-enterprise-3.4.2/dist-tomcat/atlassian-jira-3.4.2.war
GENERIC: /tmp/atlassian-jira-enterprise-3.4.2/dist-generic/atlassian-jira-3.4.2.war


See http://www.atlassian.com/software/jira/docs/latest/servers/ for install instructions
-------------------------------------------------


BUILD SUCCESSFUL
Total time: 25 seconds
jturner@teacup:/tmp/atlassian-jira-enterprise-3.4.2$
```

Then deploy the webapp in your app server.

Importing Data From Mantis

JIRA can import data from Mantis, an open-source PHP-based bug tracker. The Mantis import is almost identical to the Bugzilla importer (just expecting a different database format), so please refer to the Bugzilla documentation for a walkthrough.

Before you begin, please backup your JIRA data.

In addition to the Bugzilla importer's features, the Mantis importer also:

1. Creates a custom field, "Mantis Link", containing a link to the old Mantis bug URL. This field can be hidden when no longer useful.
2. Creates a custom field, "Mantis ID", containing the mantis bug ID (useful for searching)
3. Creates and uses a custom "Not a bug" resolution type.
4. Converts Mantis links (#1234) to JIRA links (TST-123) in text.

Importer Source code

The Mantis importer source code is available here (see also here) — fixes and improvements welcome.

Importing Data From FogBugz

On this page:

- Overview
  - Notes about the FogBugz Importer
- Step 1. Running the Import Configuration Wizard
  - Project Configuration
  - Issue Field Mappings
  - Value Mappings
  - Issue Links
  - Saving the configuration file
- Step 2. Importing the FogBugz file

Overview

The FogBugz importer allows you to import your bugs from an existing FogBugz installation. Alternatively, you may like to compare JIRA and FogBugz here.

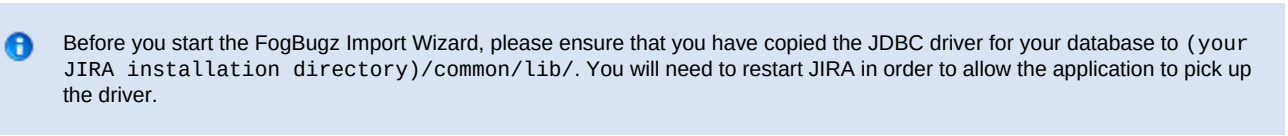
Importing from a FogBugz file is a two step process. First, you will need to create a mapping file by running the FogBugz import wizard. (The mapping file is a plain text properties file that you can also manually edit. It will map your CSV fields to fields in JIRA.) Then, to perform the import, simply enter the connection details and the location of your configuration file.

Notes about the FogBugz Importer

1. **Attachments:** JIRA will import all attachments stored in the FogBugz dB. Any e-mail issues will be parsed for attachments and the e-mail text saved as a comment. The dates and user attaching the attachments will be retained.
2. **Link rewriting:** FogBugz allows for links to other issues to be automatically generated by using the format "bug issueId" or "case issue id". After import, any string matching this pattern will be rewritten to their new JIRA key. For example, a comment "Please see case 100" may be rewritten to "Please see IMP-100".

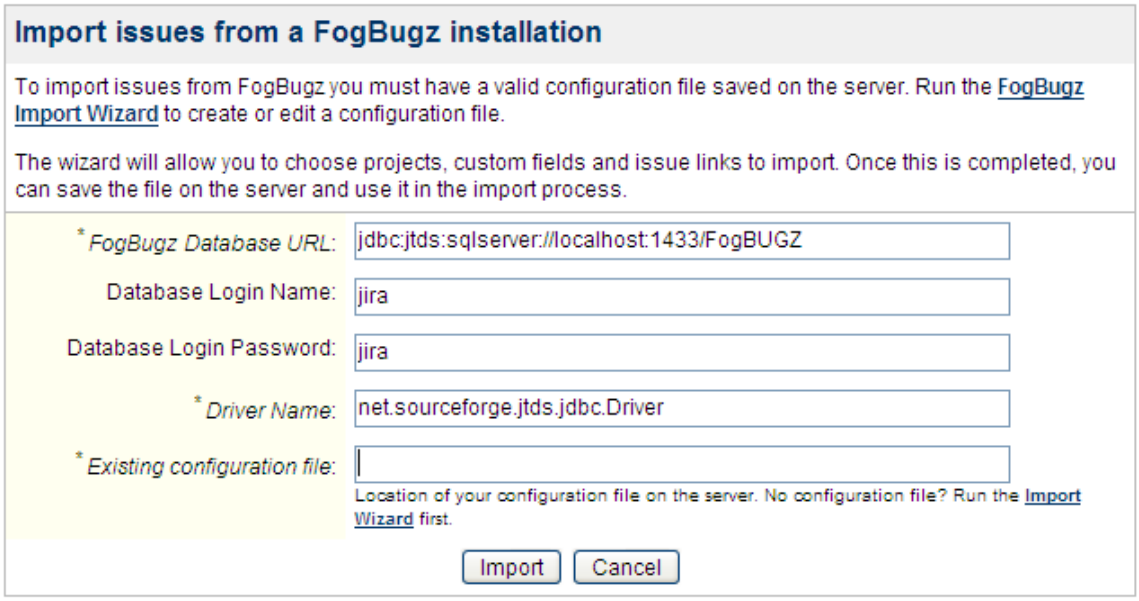
Step 1. Running the Import Configuration Wizard

The wizard allows you to choose projects, custom fields and issue links to import. Once this is completed, you can save the file on the server and use it in the import process.



To run the FogBugz Import Wizard:

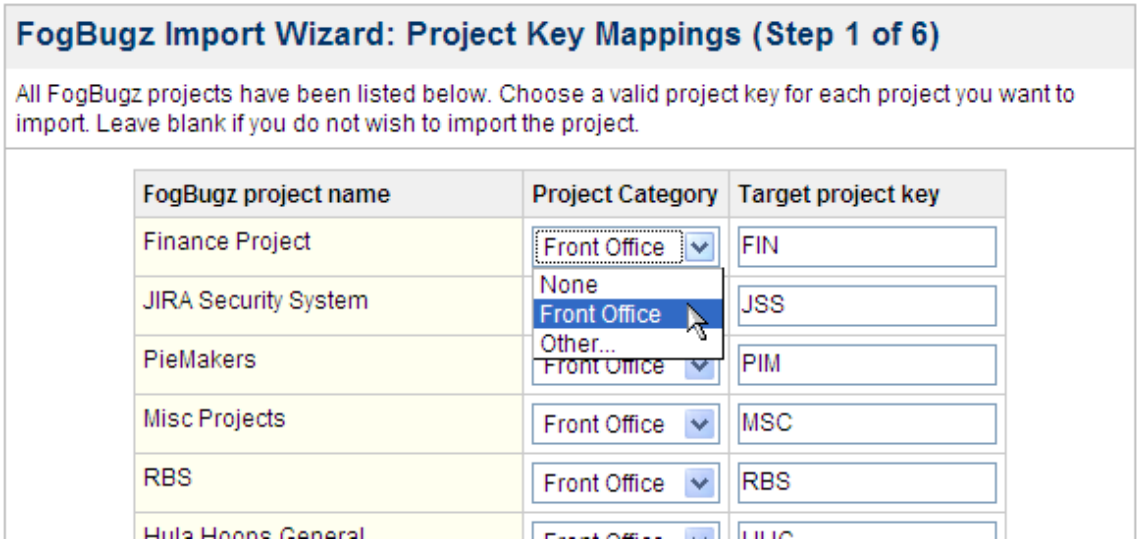
1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Import & Export',** click **'External System Import.** '
4. The **'Import Data'** page will be displayed. Select **'FogBugz'.**
5. The **'Import issues from a FogBugz installation'** page will be displayed:



6. Click the link **'Run the FogBugz Import Wizard.**

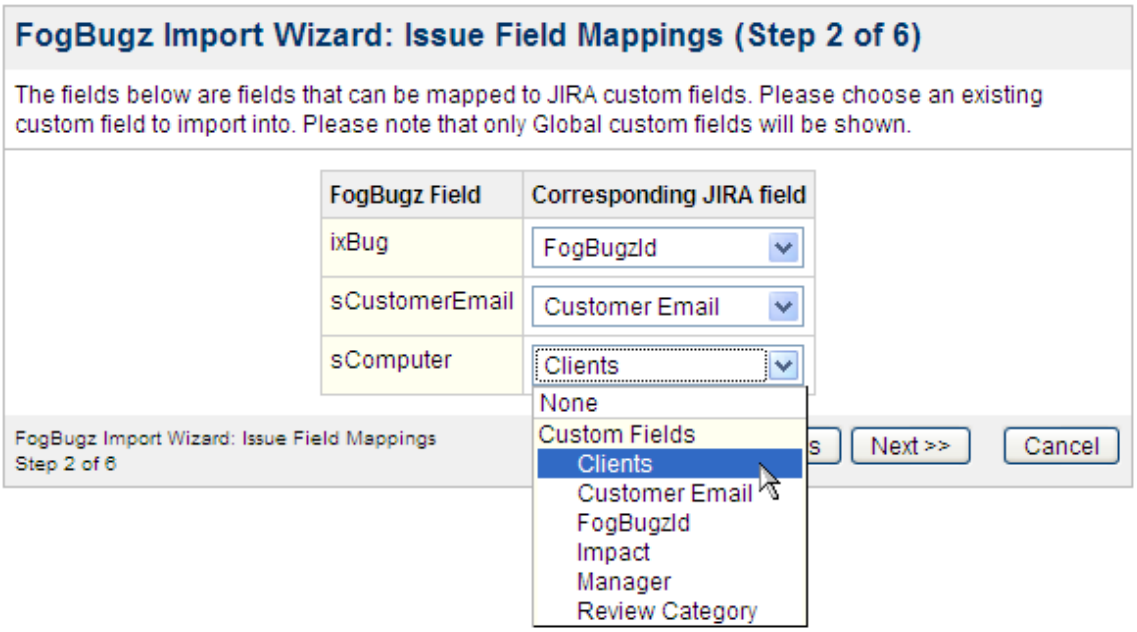
Project Configuration

The first step is to choose which projects are to be imported. Choose a valid project key for each project you want to import, and leave blank if you do not wish to import the project. In JIRA Enterprise Edition, you can also choose a project category that the project should be created in. This will only apply if the project key does not exist and a new project is created.



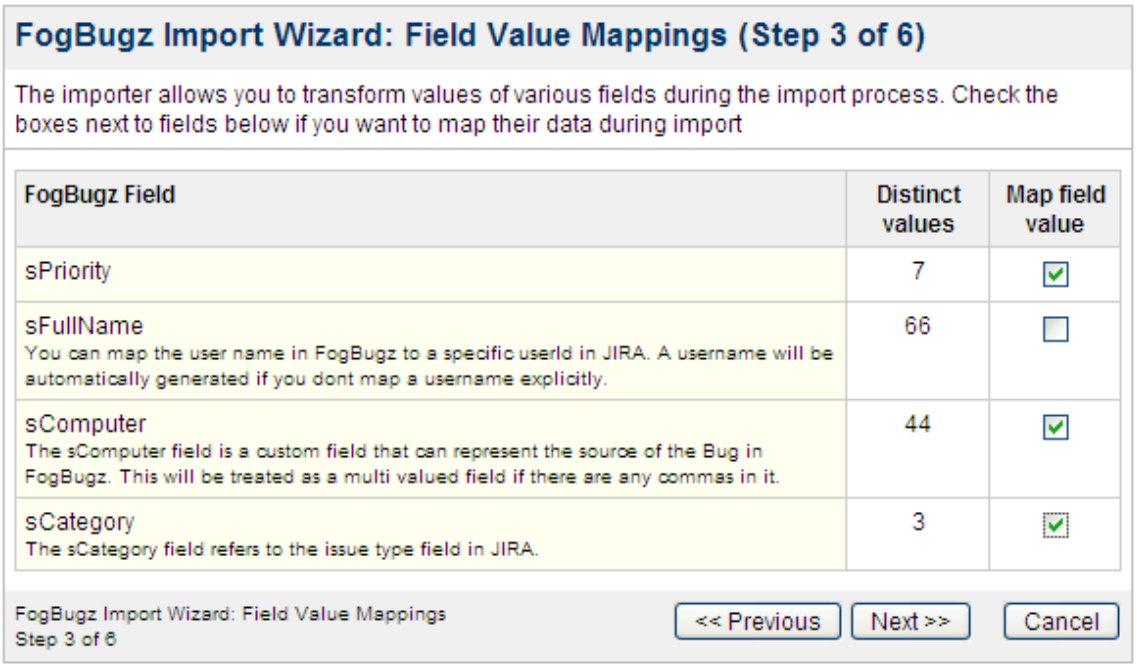
Issue Field Mappings

The second step is to decide which of the optional FogBugz fields you wish to import. Most of the fields in FogBugz are automatically mapped to appropriate JIRA fields. You can choose to map the fields **ixBug** (internal FogBugz issue id), **sCustomerEmail** (e-mail address of the issue if created by e-mail) and **sComputer** (custom field) to any existing global JIRA custom field.



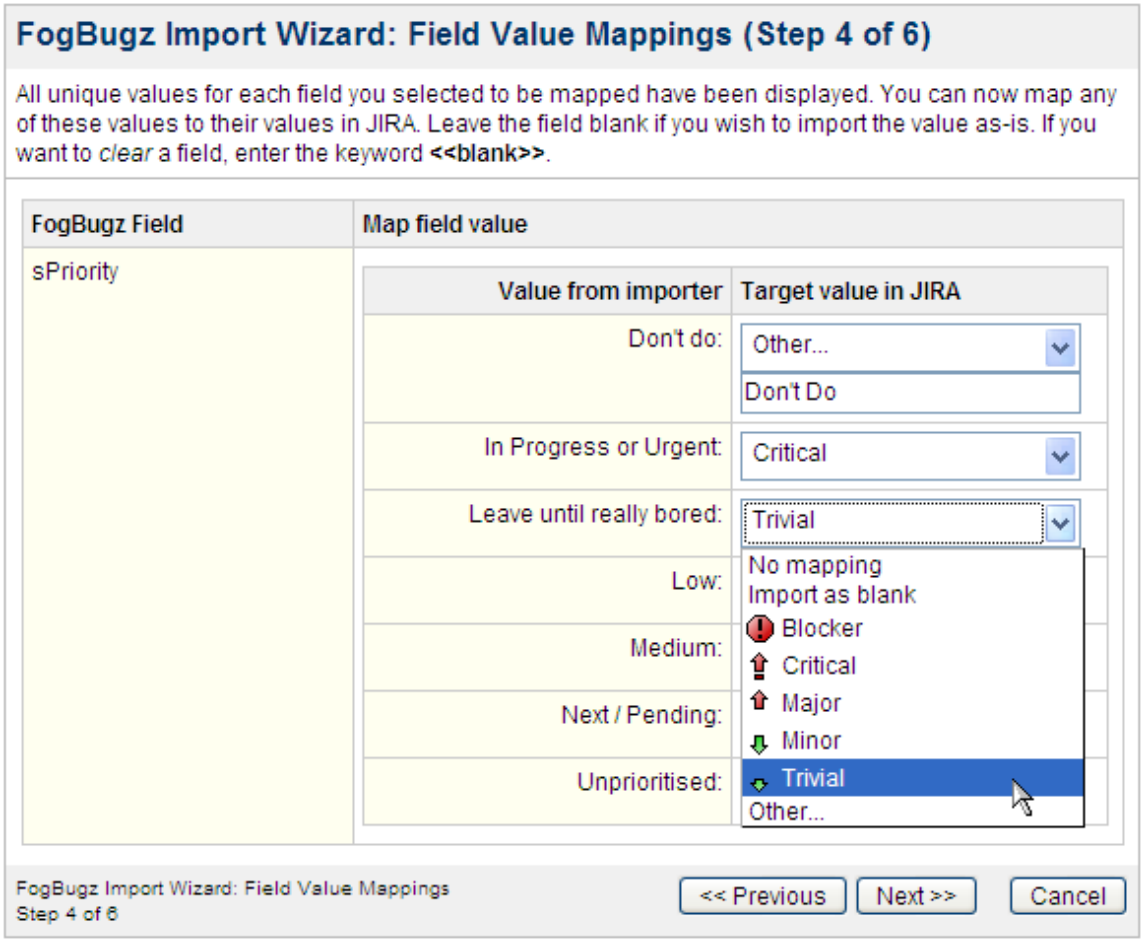
Value Mappings

Value mappings are how values from your FogBugz importer can be 'transformed' during import. The fields **sPriority, sFullName, sComputer** and **sCategory** can have their values mapped. The first screen allows you to choose which fields you would like to map values.



On the next screen, all unique values for each field you selected to be mapped have been displayed. You can now map any of these values to a new value. Leave the field blank if you wish to import the value as-is. If you want to clear a field, enter the keyword **<<blank>>.**

For **sPriority** and **sCategory** you will get a select list with the available values in JIRA. **sFullName** refers to the usernames that will be created in JIRA. If you don't specify any particular mappings, the user name will be created from the first letter of the first name and the last name, all in lowercase.



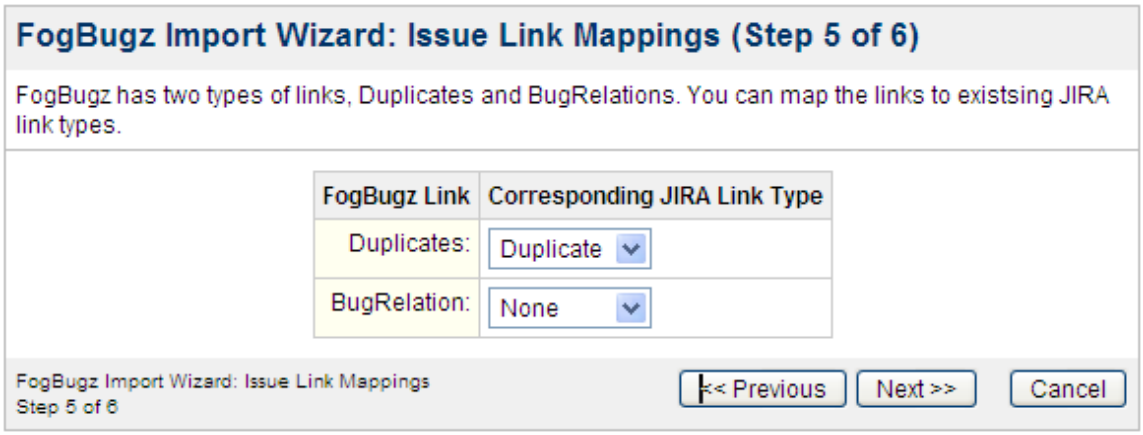
If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import.

If you have a user limited license (e.g. personal license) and have reached your user limit, you can import additional users but any new users added will not be able to log in to JIRA.

If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

Issue Links

FogBugz has two types of links, **Duplicates** and **BugRelations.** On this screen, you can map the links to existing JIRA link types. Leave as **none** to not import the links.



Saving the configuration file

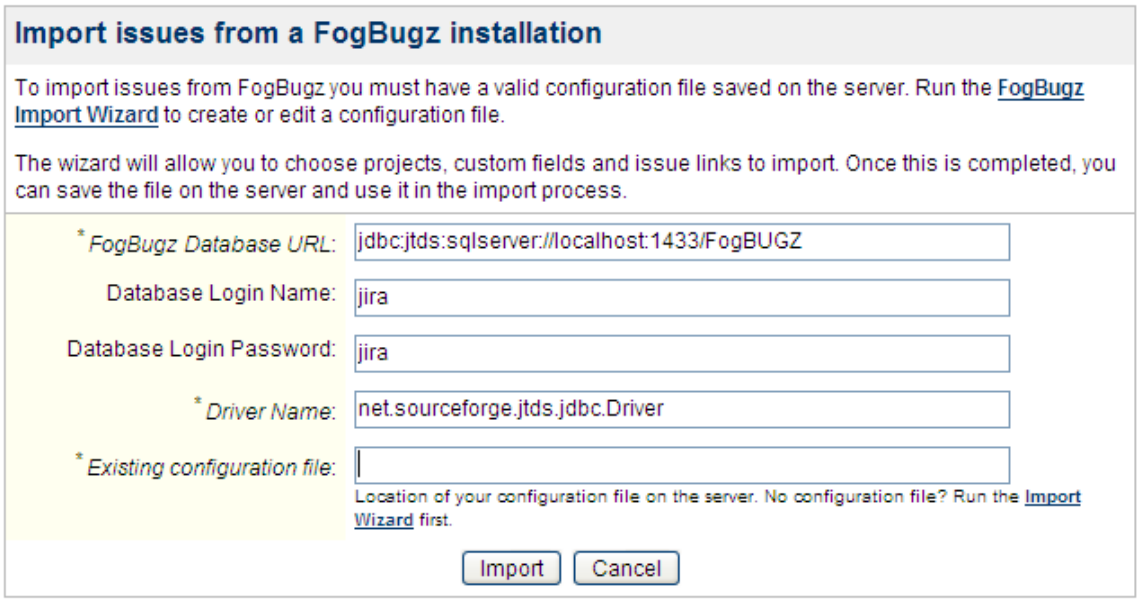
The final step of the Wizard is to save the file on your server. Please ensure you enter a valid path. You can also choose to continue on with the import without saving the file.

You can also see a preview of the mapping file that will be saved.

Step 2. Importing the FogBugz file

Once you have your configuration file, you can now import the FogBugz file into JIRA.

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title **'Import & Export',** click **'External System Import.** '
4. The **'Import Data'** page will be displayed. Select **'FogBugz'.**
5. The **'Import issues from a FogBugz installation'** page will be displayed:



6. Type the location of your FogBugz file and your configuration file, and click the **'Import'** button.

The **'Settings'** page gives you precise control over what will be imported on each import run.

Once the import has begun you'll be able to follow live progress of the import, with the screen refreshing at around every 10 seconds. You can change this value by updating the field at the bottom of the page. The importer also give you stats about what objects have been imported and time elapsed so you can have an idea as approximately how long the import will take. You can also choose to Abort the import, which will cease importing after the current issue is done.

Moving or Archiving Individual Projects

Over time, your organisation's requirements may change. This can lead to needing to:

- Archive a completed or obsolete project.
- Split a large JIRA instance into several JIRA instances, with particular projects in each.
- Restore a single project from a backup file into a JIRA instance.
- Restore an entire JIRA instance, from a backup into a new empty JIRA instance.

Archiving a Project

It is sometimes necessary to archive an old project, while retaining the project's data for future auditing purposes. There are a number of ways to achieve this:

- Online archiving
  - 'Hiding' a project
  - Making a project 'Read-Only'
  - Accessing an archived online project
- Offline archiving
  - Archiving a project offline
  - Accessing an archived offline project
  - Restoring a deleted project

Online archiving

Archiving a project online means keeping all of the project's issue data in your live JIRA instance. The advantage of archiving a project online is that you can easily make the project accessible again if required.

There are two ways to archive a project online:

'Hiding' a project

A 'hidden' project will still be visible via the 'Administration' menu, but it will no longer appear in the 'Browse Projects' list, and noone will be able to search, view or modify any of the project's issues.

1. Create a new permission scheme. Leave all of the permissions empty.
2. Associate the new permission scheme with the project that you wish to hide (see Assigning a Permission Scheme to a Project).

Making a project 'Read-Only'

If you make a project read-only, the project will be visible via the 'Administration' menu, and will appear in the 'Browse Projects' list. The project's issues will be searchable and viewable, but noone will be able to modify them.

1. Create a new permission scheme. Grant the **'Browse Project'** to everyone who needs to be able to search or browse the project, or view its issues. Leave all of the other permissions empty.
2. Associate the new permission scheme with the project that you wish to hide (see Assigning a Permission Scheme to a Project).

Accessing an archived online project

If you archived a project online, by hiding it or making it read-only, then all of the project's data can be made accessible very easily. Simply associate the project with a permission scheme where the appropriate permissions (e.g. **'Edit Issue', 'Assign Issue', 'Resolve Issue',** etc) are assigned to the appropriate people.

Offline archiving

Archiving a project offline means creating an XML backup, then deleting the project and all of its issue data from your live JIRA instance. The project will no longer be available via the 'Administration' menu or the 'Browse Projects' list, and its issues will no longer exist in your live JIRA system.

The disadvantage of offline archiving is that there is no easy way to restore a deleted project to your live JIRA instance.

If there is a possibility that you will need to restore the project into your live JIRA instance at some point in the future, then online archiving is recommended. Offline archiving should only be done if you are certain you will never need to restore this project to a live JIRA instance (i.e. you will only ever restore the data to a non-production instance).

Archiving a project offline

1. Create a global XML backup of your entire live JIRA instance.
2. Import the XML backup into a test JIRA instance. **Make sure that the test JIRA instance uses a separate database from your live JIRA instance, as the import will overwrite all data in the database.**
3. In your test JIRA instance, verify that you can view the issues of the project that you are archiving.
4. In your live JIRA instance, select **Projects** from the **Administration** menu, then click the **Delete** link to delete the project and all of its issues.

Accessing an archived offline project

1. Import the XML backup into a test JIRA instance. **Make sure that the test JIRA instance uses a separate database from your live JIRA instance, as the import will overwrite all data in the database.**

Restoring a deleted project

If you wish to restore a project from a backup file, please refer to the instructions in the Restoring a Project from Backup documentation.

Splitting a JIRA instance

Occasionally an organisation may need to split its existing JIRA instance into two separate instances. For example, there might be a requirement to have some particular projects in one JIRA instance, and other projects in a second instance.

> **Note**
> This requires two separate server licenses.

To split a JIRA instance,

1. Backup your database, using your database backup procedures, and verify the backup.
2. Backup your attachments directory and verify the backup.
3. Install JIRA on your new server. NOTE:
    - The JIRA version number on your new server must be the same as (or higher than) the version number on your existing server.
    - The JIRA edition on your new server must be the same as (or higher than) the edition on your existing server. E.g. If your existing server is running JIRA Professional edition, the new server must be running either JIRA Professional or JIRA Enterprise edition (not JIRA Standard edition).
    - Do not use the same JIRA home directory for the two JIRA instances. Specify a new JIRA home directory for the JIRA on your new server.
    - Do not connect the two JIRA instances to the same external database instance.
4. Create an XML file from your existing JIRA server, as described in Backing up data.
5. Import the XML file into your new server, as described in Restoring data.
6. Copy the attachments directory from your existing server to your new server, and configure your new server to use its own directory (for details please see Enabling File Attachments).
7. At this point you should have two JIRA instances with the same users, projects, issues and attachments. Log into both instances and perform some random searches to verify that the data is identical in both instances.
8. Delete the non-required projects from each JIRA instance.

Integrating with a Source Control System

JIRA can be easily integrated with many popular source control systems:

- Integrating JIRA with CVS and ViewCVS
- Integrating JIRA with Subversion
- Integrating JIRA with Perforce
- Integrating JIRA with ClearCase
- Integrating JIRA with FishEye

Integrating JIRA with CVS and ViewCVS

JIRA's CVS integration shows the related CVS commit information for an issue. When a CVS commit message mentions an issue, JIRA picks this up and displays the commit log in a tab in the mentioned issue.

JIRA's CVS integration features include:

- Ability to interact with a CVS server log directly via local access, pserver or external (ssh) protocols, or to parse a CVS log file generated by an external process.
- Access to the version control information in JIRA can be easily controlled using flexible permissions. If you are running a public instance of JIRA, and do not want the rest of the world to see the version control information, JIRA can be configured to restrict access to that information to the chosen users.
- ViewCVS or FishEye are supported out-of-the-box; and Subversion is available as a plugin (drop-in extensions to JIRA).

- If CVS integration is configured, the files and revisions in JIRA are linked to the relevant pages. E.g. the following screenshot shows a JIRA project:

  Because ViewCVS is configured, JIRA has turned the displayed commit information into ViewCVS links.
  - Clicking the name of the file will take the user to the ViewCVS file summary page.
  - Clicking the revision will take the user to the page that shows the contents of the file as it was at that revision.
  - Clicking the 'diff' summary will show the ViewCVS 'diff' page between the shown revision of the file and its previous revision.
- Each project in JIRA can be associated with a CVS module. In JIRA Standard, a project can be associated with only one module. In JIRA Professional and Enterprise editions, a project can have multiple modules.

There are 3 steps to configure CVS integration in JIRA:

1. **Create a CVS module**
2. **Associate project(s) with CVS module(s)**
3. **Grant permission to view CVS information**

How JIRA's CVS integration works

JIRA retrieves the CVS commit information for an issue by parsing the output of the 'cvs rlog' (or cvs log) command of each associated CVS module and scanning it for the issue's key. If an issue key is found in the commit message, the commit message is displayed on the Version Control tab for the issue.

If you have allowed JIRA to automatically synchronise with the CVS repository, JIRA will periodically run the 'cvs rlog' command for the module and store the results in a file which path is specified by the module's *Log File Path* attribute. The file is then parsed for commit information.

> Even if you are using local repository access JIRA will obtain the CVS log for the module and then parse it. JIRA does not access the CVS repository directly.

If you have chosen to update the log manually, JIRA will only periodically parse the CVS log specified by the module's *Log File Path* attribute.

As JIRA parses the module's CVS log and keeps relevant commits in memory, the required memory for JIRA is relative to the size of the CVS module.

Please note:

- Currently, JIRA is able to retrieve CVS log data via local access, pserver protocol or ssh (ext method). If your CVS is not reachable by these methods you can JIRADOC:disable automatic log retrieval (see below).
- If you would like JIRA to automatically keep synchronised with your CVS repository, the communication between JIRA and the CVS server might be fairly bandwidth intensive as JIRA will periodically retrieve the CVS module's log data from the CVS repository. If this is causing problems, consider JIRADOC:adjusting the frequency (see below) or JIRADOC:disabling CVS log retrieval.
- JIRA loads and parses the output of the 'cvs log' command for each CVS module and keeps 'relevant' commits in memory. Therefore JIRA's memory requirements depend on the number of relevant commits found in the CVS module. Relevant commits are CVS commits which have at least one potential JIRA key in their commit messages.
- Only commit messages which contain a possible JIRA issue key are linked to an issue.
- JIRA's 'System encoding' is used when parsing the CVS logs, so it needs to match that of the CVS log. The system encoding can be seen at Admin -> System -> System Info. See also how to set the system encoding.

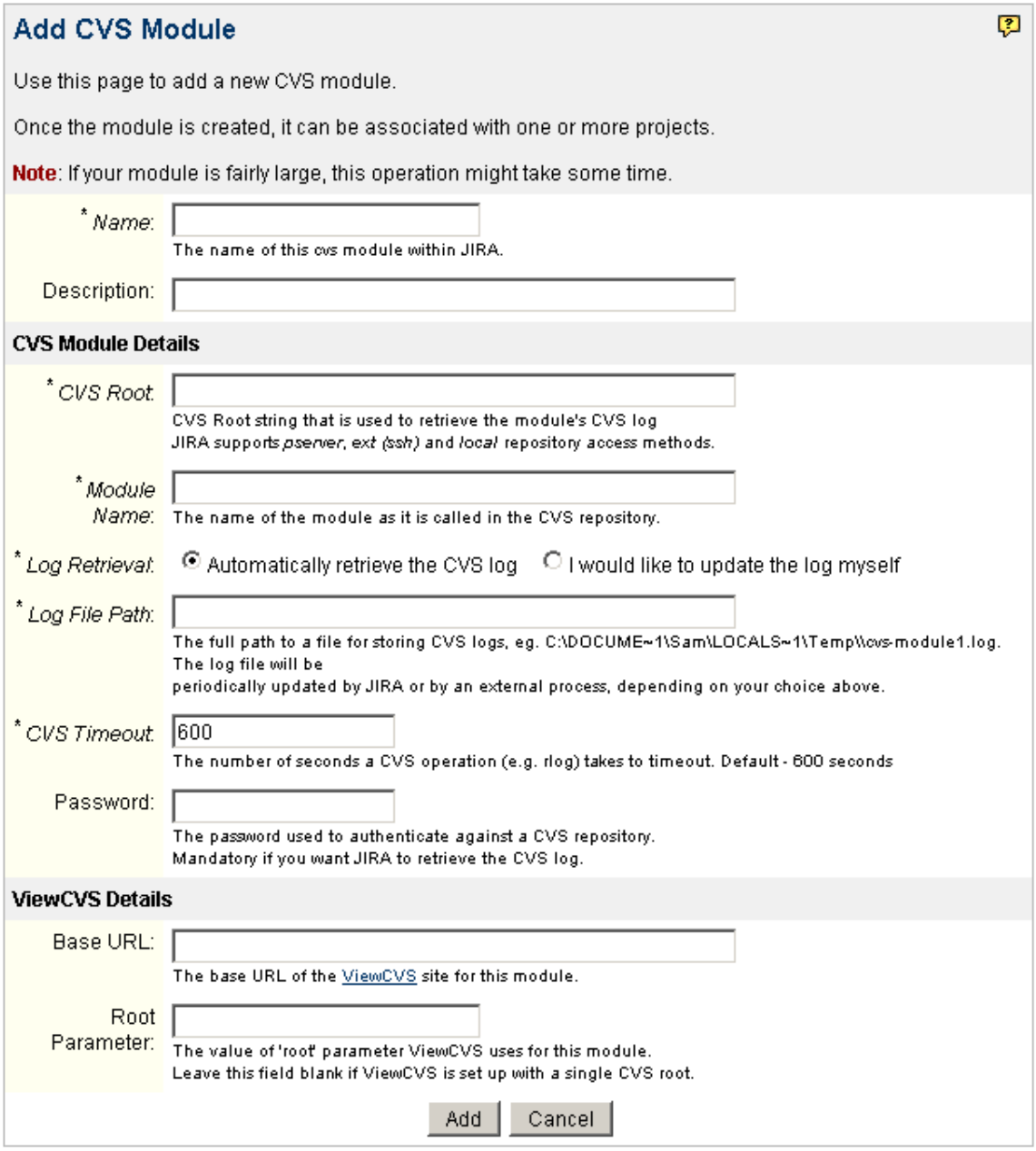
Step 1. Create a CVS Module in JIRA

A CVS 'module' refers to a top-level directory in a CVS repository. To create a CVS module:

1. Create or decide which existing directory will be used to store CVS module's log data (The file with the output of the 'cvs log' command). JIRA must have read and write access to the directory. The write access is required even if you choose to update the CVS log manually

as JIRA needs to use this directory to create a lock file in order to synchronise access to the CVS module's log.

2. Log in as a user with the **'JIRA System Administrators'** global permission.
3. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
4. On the panel on the left, expand the sub-menu titled **'Global Settings'** if it is not open already. Click on the link labelled **'CVS Modules'.** This should bring up the **'CVS Modules'** page.
5. Click on the **'Add new CVS module'** link on this page.
6. This will bring up the **'Add CVS Module'** page.



Fill in as follows:

a. For **'Name'** put a short descriptive name, possibly just the name of the CVS module as it appears in your CVS repository.
b. (Optional) For **'Description'** put a short phrase that describes this CVS module.
c. Specify **'CVS Root'** that will be used to retrieve the CVS module's log or was used to retrieve the log. The CVS Root is needed while parsing the log data so it is required even if you choose to retrieve CVS log manually. Please provide 'full' CVS Root details. For example:

- /some/local/path (for local repository access)
- :pserver:username@hostname:port/some/path (for pserver access)
- :ext:username@hostname:/some/path (for ssh access)If JIRA finds trouble understanding your local CVS Root (e.g. on Windows systems) please prefix the path with *:local:*. For example, :local:d:\some\path.

a. For **'Module Name'** specify the name of the module as it is called in the CVS repository. This will usually be the top-level directory (eg. `myproject`), but can also include subdirectories (`myproject/subproject/src/java`) - basically anything that can be parsed to a `cvs checkout` command.This information is required to retrieve the CVS log as well as to parse it, so you will need to provide the module's CVS name even if you choose to retrieve the CVS log manually.
b. For **'Log Retrieval'** choose whether you would like JIRA to automatically synchonise with the CVS repository. If you choose

'Automatically retrieve the CVS log', JIRA will JIRADOC:periodically retrieve the CVS log for the module automatically and then parse it for commit information. If you choose **'I would like to update the log myself',** JIRA will not retrieve the log, but will JIRADOC:periodically just parse it. If you choose this option you will need to JIRADOC:update the CVS log by other means (e.g. manually or using a scheduled JIRADOC:script) to keep the CVS information in JIRA current.

c. For **'Log File Path'** specify the full path to the **file** that will contain the CVS log data. This file should be located in a directory mentioned in step 1. If you would like JIRA to periodically JIRADOC:update the contents of the log this file does not need to exist at the moment, as JIRA will automatically create it. If you choose to manually update the file please ensure that the log file already exists at the specified path and is readable by JIRA.
d. For **'CVS Timeout',** specify how many seconds it takes the CVS operation (e.g. rlog) to timeout
e. The **'Password'** needs to be provided only if you let JIRA automatically retrieve the module's CVS log. Please specify the password that is needed to retrieve the log using the method specified in the CVS Root. If no password is required, leave the field empty.
f. (Optional) For **'Base URL'** in the **'ViewCVS Details'** section of the page, enter the fully qualified URL (i.e. include 'http://' or 'https://' at the beginning) to the ViewCVS site of the CVS module. The URL needs to point to the root of the module on the ViewCVS site.

> If you are integrating with FishEye you do not need to perform any special steps. FishEye can resolve all the URLs that ViewCVS expects. You just need to enter the fully qualified URL to your FishEye installation and the specific repository you wish to view. This is the same URL you would get if you were to browse to the project within FishEye.

g. (Optional) For **'Root Parameter'** in the **'ViewCVS Details'** section of the page, enter the name of the Project Root that is used in ViewCVS to navigate the CVS module. This parameter is required only if ViewCVS has been set up to work with multiple CVS modules, and this module is not the default module on the ViewCVS server. The value that should be placed in this field is the same as the value of the 'root' URL parameter that appears on every ViewCVS URL (e.g. when viewing a file). If the URL that appears in your browser when viewing a file from this CVS module on ViewCVS does not have the 'root' parameter, leave this field blank.

7. Click the **'Add'** button.
8. This should bring you back to the **'CVS Modules'** page, where you should see the new CVS module listed. You can edit and delete this module here.

> If JIRA has trouble understanding your local CVS Root (e.g. on Windows systems) please prefix the path with :local:. For example, :local:d:\some\path

Step 2. Associate Project(s) with CVS Modules

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand navigation panel, click the **'Projects'** link.
4. This will bring up the **'Projects'** page. It lists all the existing projects. Select a project that you would like to associate with the CVS module.
5. The project's summary page sill be displayed. Next to **'CVS Modules'** click the **'Celect Module'** link. This will display the **'Select Version Control Modules'** page, where you can associate the project with a CVS module (or with multiple CVS modules, in JIRA Professional and Enterprise editions).
6. Select the appropriate module(s), and click the **'Select'** button.

Step 3. Configuring Permissions

The 'View Version Control' permission needs to be given to users/groups/roles that should be allowed to see CVS commit information. Note: by default this permission is given to the 'jira-developers' group. Please read the Project Permissions section, and follow the instructions given there to assign the 'View Version Control' permission.

Disabling Automatic CVS Log Retrieval

To disable automatic CVS log retrieval for a CVS module please choose the **'I would like to update the log myself'** option for the module's **'Log Retrieval'** attribute.

If you have disabled automatic CVS log retrieval for the CVS module, JIRA will only *parse* the CVS log periodically. Therefore, for the new commit information to appear in JIRA, the log needs to be updated by other means. This can either be done manually, or a scheduled CVS update script can be used.

Before updating the module's CVS log, please check for the existence of a lock file with name **'cvslog.write.lock'** in the *same* directory as the CVS log file. If the lock file exists, please wait until it is removed before updating the log.

When updating the CVS log for a module, please create a lock file with the name **cvslog.write.lock** in the *same* directory as the CVS log file to ensure that JIRA does not start parsing the log while it is still being updated. Please do not forget to remove the lock file after the update has finished.

Adjusting the Frequency of Module Updates

To minimise the network traffic between JIRA and the CVS server, JIRA updates and re-parses the commit information of the associated CVS modules only once during the specified period of time. By default, this period of time is 1 hour, but it can be adjusted if required.

When the first CVS module is created in JIRA, a background service is automatically started. The service is called **'VCS Update Service'.** To change the frequency of the module updates, follow these steps:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Open the **'System'** tab of the left-hand menu, if it is not already open.
4. Select **'Services'** from the **'System'** tab. A page showing all the configured services will appear. If at least one CVS module has been configured, the **'VCS Update Service'** should be present in the list.
5. Click the **'Edit'** link in the right-most column of the **'VCS Update Service'.** This will display a page where you can set the delay for the service.
6. Change the value as required. Remember that the delay is specified in minutes.
7. Click the **'Update'** button to make the changes take effect.

Please keep in mind:

- The CVS modules are updated one after another every specified period of time. That is, it is not possible to specify a different update delay for each configured CVS module.
- If you are using automatic log retrieval for your CVS modules and you set the delay to a very low value, the bandwidth consumption between JIRA and the CVS server might be very high.
- If the delay is set to a very large value, the 'new' cvs commit messages will not appear in JIRA for some time.

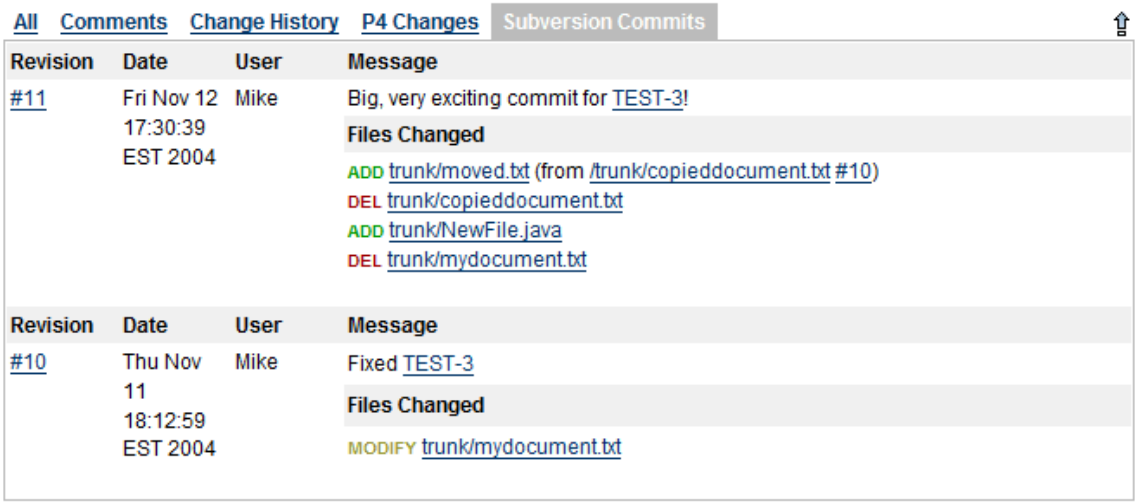
CVS Aliases

JIRA does not currently support CVS aliases. If you have a CVS alias that references more than one module, please create each CVS module in JIRA and then associate each module with the relevant JIRA project(s) (assuming you are using JIRA Professional or Enterprise edition, as Standard edition only supports one CVS module per project).

The feature request for adding CVS alias module support to JIRA is JRA-4586. Please vote for the issue to increase its popularity. Please refer to this document, which describes the way Atlassian implements new features and improvements.

Integrating JIRA with Subversion

JIRA's Subversion integration lets one see Subversion commit information relevant to each issue. Subversion integration is implemented as a plugin (drop-in extension) to JIRA.



Commits will appear in this tab if the commit log mentions the issue key ('TEST-3' above).

For more information, see the Subversion plugin page online.

Integrating JIRA with Perforce

> ⚠ The Perforce Plugin for JIRA is now **deprecated,** as it has been superseded by the FishEye plugin for JIRA. Please ensure that you read the Perforce plugin page for further information, including details on support and our recommended FishEye solution.

JIRA's Perforce integration lets you see Perforce commit information relevant to each issue:

School

no create function

Return to search

Created: Today 09:01 AM Updated: Today 03:13 PM

Affects Version/s: None

Fix Version/s: None

Perforce Job:	Perforce job exists - view details.

All Comments Change History P4 Job

Perforce Job: SCH-19

This shows all of the change lists associated with this issue's Perforce job (SCH-19).

Change	Date	User	Description
9	2004/09/30 15:06:09 *pending*	Mike	my new changelist **Files Changed** EDIT //depot/myfolder/new-subfolder/still not exciting.txt #1
Change	**Date**	**User**	**Description**
8	2004/09/30 14:21:26	Mike	lots of modifications like adds and deletes **Files Changed** EDIT //depot/mydocument.txt #4 DEL //depot/myfolder/AnotherDoc.java #6 ADD //depot/myfolder/new-subfolder/still not exciting.txt #1 ADD //depot/myfolder/not exciting.txt #1
Change	**Date**	**User**	**Description**
7	2004/09/30 14:17:36	Mike	not exciting **Files Changed** EDIT //depot/myfolder/AnotherDoc.java #5

This includes two-way integration with Perforce Jobs, for example creating a job when an issue is created:

Create Issue

Step 2 of 2: Enter the details of the issue...

Project: School

Issue Type: Bug

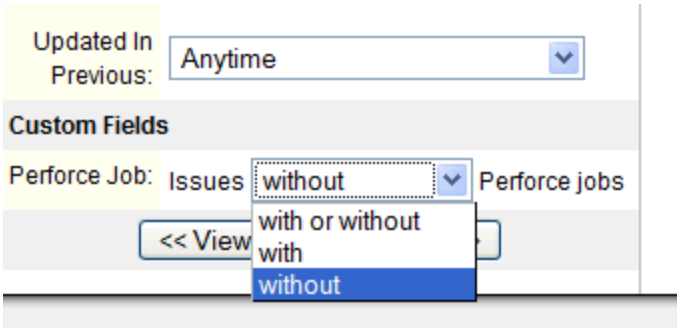
* *Summary*:

Perforce Job: Create Perforce job

Priority: Major

Affects Version/s: Unknown

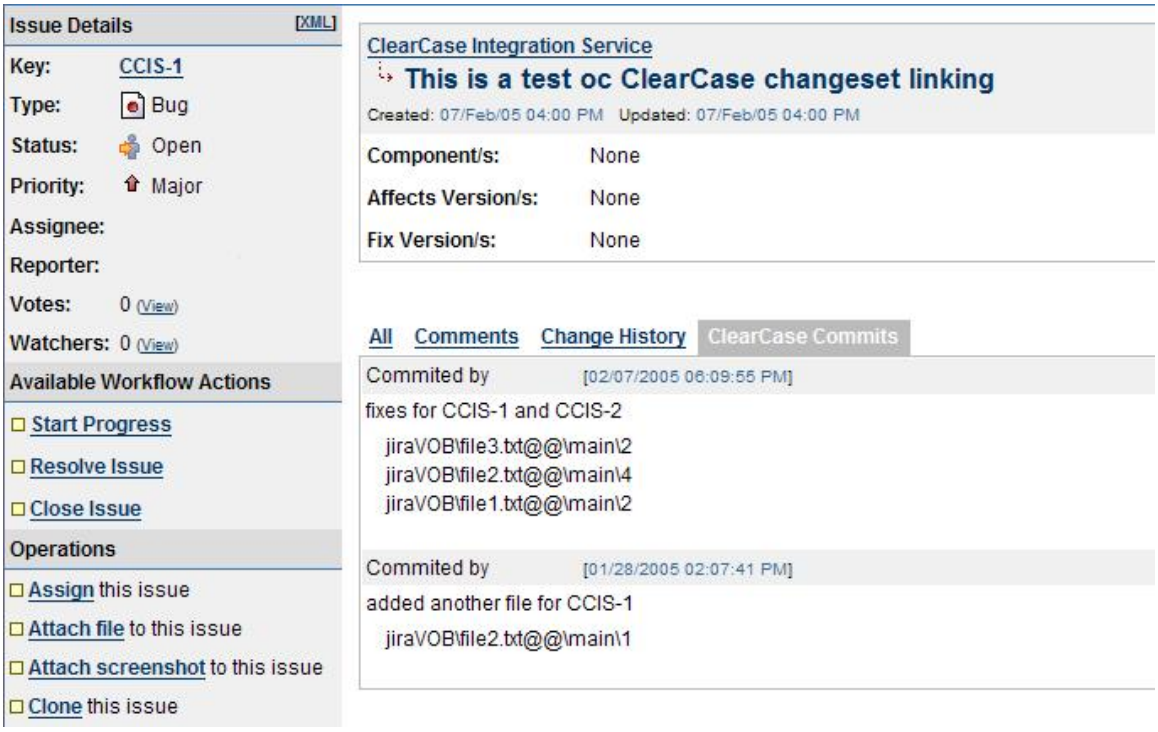
and searching for JIRA issues with associated perforce jobs:



Perforce integration is implemented as a plugin (drop-in extension) to JIRA, which is licensed separately to JIRA.

Integrating JIRA with ClearCase

Although not developed or supported by Atlassian, there is a JIRA ClearCase plugin available which you may find useful. It shows ClearCase checkins associated with JIRA issues:



This can be combined with the Commit Acceptance Plugin to get further towards the level of integration ClearQuest offers.

Integrating JIRA with FishEye

JIRA's FishEye integration allows you to browse your source-control repository from inside JIRA, provided you are using Atlassian FishEye with your source-control repository. FishEye integration is implemented as a plugin (drop-in extension) to JIRA.

FishEye integration allows you to:

- View an Issue's FishEye Changesets
- Browse a Project's FishEye Changesets

For more information, see the FishEye plugin page.

Configuring Global Settings

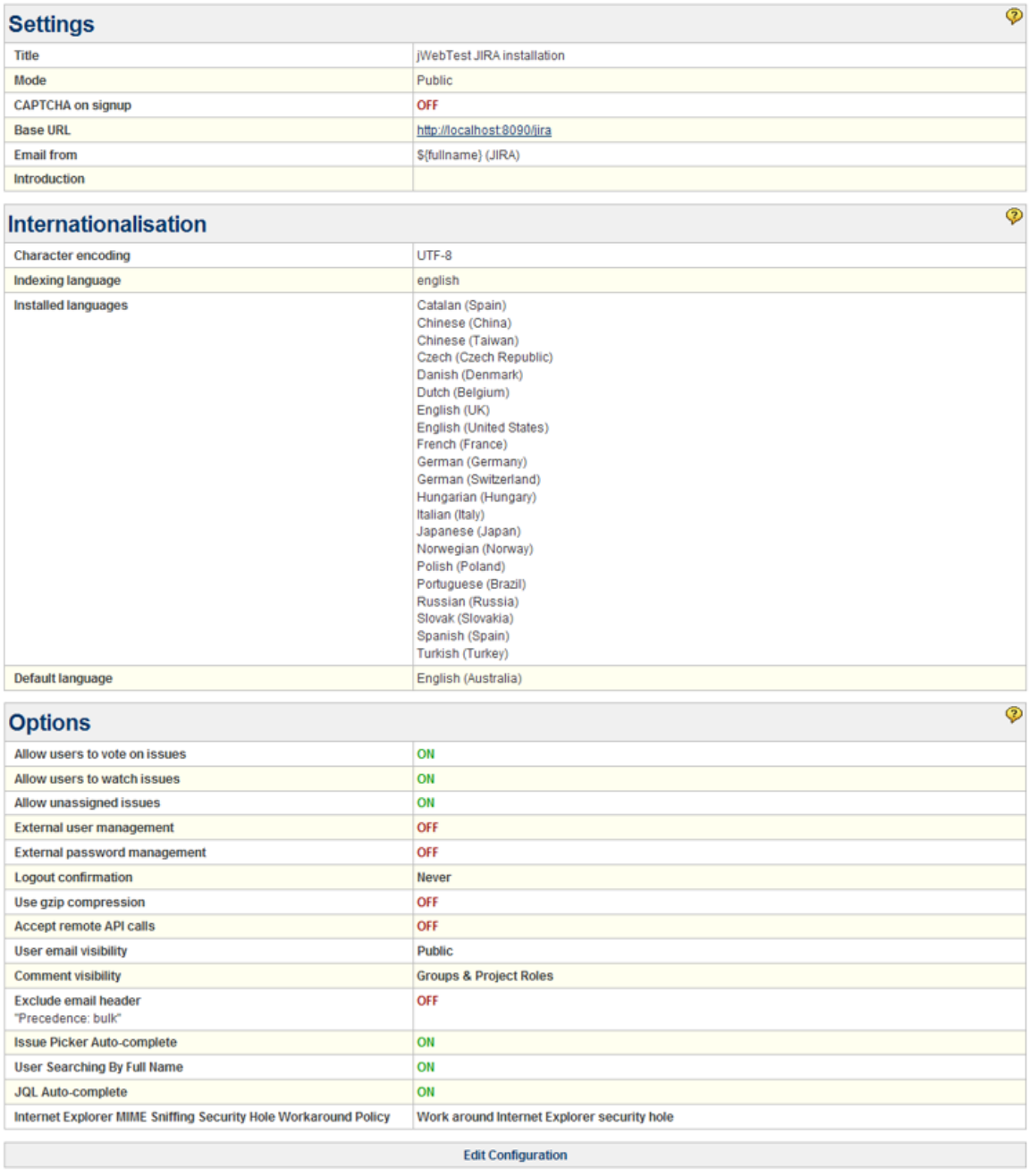
- Configuring JIRA Options
- Setting properties and options on startup
- Advanced JIRA configuration with jira-application.properties
- Enabling File Attachments
    - Enabling Thumbnails for Attachments

- Enabling Sub-tasks
- Enabling Issue Linking
- Enabling Trackback
- Enabling Time Tracking
- Configuring OAuth Consumers

Configuring JIRA Options

JIRA has a number of configuration options that allow the system to be customised for use within your organisation. To edit these options:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'Global Settings', click on the link labelled 'General Configuration'.
4. The following screen will be displayed. Click 'Edit Configuration' to edit the three sections as described below:
    - Settings
    - Internationalisation
    - Options



Settings

Title	jWebTest JIRA installation
Mode	Public
CAPTCHA on signup	OFF
Base URL	http://localhost:8090/jira
Email from	${fullname} (JIRA)
Introduction	

Internationalisation

Character encoding	UTF-8
Indexing language	english
Installed languages	Catalan (Spain) Chinese (China) Chinese (Taiwan) Czech (Czech Republic) Danish (Denmark) Dutch (Belgium) English (UK) English (United States) French (France) German (Germany) German (Switzerland) Hungarian (Hungary) Italian (Italy) Japanese (Japan) Norwegian (Norway) Polish (Poland) Portuguese (Brazil) Russian (Russia) Slovak (Slovakia) Spanish (Spain) Turkish (Turkey)
Default language	English (Australia)

Options

Allow users to vote on issues	ON
Allow users to watch issues	ON
Allow unassigned issues	ON
External user management	OFF
External password management	OFF
Logout confirmation	Never
Use gzip compression	OFF
Accept remote API calls	OFF
User email visibility	Public
Comment visibility	Groups & Project Roles
Exclude email header "Precedence: bulk"	OFF
Issue Picker Auto-complete	ON
User Searching By Full Name	ON
JQL Auto-complete	ON
Internet Explorer MIME Sniffing Security Hole Workaround Policy	Work around Internet Explorer security hole

Edit Configuration

Settings

Setting	Description
Title	This is the title that will be displayed on the JIRA login page and the dashboard. It helps identify your installation and its purpose. (Note: also see logo, which is displayed on every JIRA page.)
Mode	JIRA can operate in two modes: • **Public** — Anyone can sign themselves up and create issues (within the bounds of your JIRA system's permissions). • **Private** — Useful for internal issue-tracking systems where you do not want public users to login. Self-signup is disabled; only Administrators can create new users. *Default:* ***Public***
CAPTCHA on signup	If you are running JIRA in Public mode (see above), it is strongly recommended that you enable CAPTCHA. This will show a CAPTCHA image on signup to prevent spambots from signing up. *Default:* ***ON***
Base URL	This is the base URL of this JIRA installation.It is used in outgoing email notifications as the prefix for links to JIRA issues.
Email from	Specifies the **From:** header format in notification emails. Default is of the form "John Doe (JIRA) <jira@company.com>". Available variables are $'{fullname'}, $'{email'} and $'{email.hostname'}. Note that the actual address (e.g. 'jira@company.com') cannot be specified here - it is determined by the mail server or individual project configuration.
Introduction	A short introduction message displayed on the dashboard. (Note: also see the announcement banner, which is displayed on *every* JIRA page.)You can include HTML, but ensure all tags are correctly closed.

Internationalisation

Setting	Description
Character encoding	The character encoding for input and viewing of information within JIRA. For most western languages, the default ("UTF-8") should be suitable.If you change this setting, ensure that you also change your database's encoding. View a list of supported encodings. However, please use the IANA preferred MIME name (such as 'iso-8859-1' instead of 'ISO8859_1') to ensure that XML backups have the correct encoding string.
Indexing language	JIRA uses Lucene, a high-performance text search engine library, in full-text searches over the issues stored in JIRA.Indexing and searching for issues is affected by the language that the issues are entered in. Please choose the language that issues are entered in. If multiple languages are chosen, choose 'Other'. Note: You will need to re-index JIRA if you change this value.
Installed languages	This section lists all language packs available within the JIRA system. (Note: to install additional languages, see Internationalisation.)
Default language	This is the language used throughout the JIRA interface (as selected from the list displayed in **Installed Languages** above). Users can override the default language by using the **Language** setting in their user profile.

Options

Setting	Description
Allow users to vote on issues	Controls whether voting is enabled in JIRA. Voting allows users to indicate a preference for issues they would like to be completed or resolved. See also the 'View Voters and Watchers' permission. *Default:* ***ON***
Allow users to watch issues	Controls whether watching is enabled in JIRA. Users can 'watch' issues which they are interested in. Users watching an issue will be notified of all changes to it. See also the 'View Voters and Watchers' and 'Manage Watcher List' permissions. *Default:* ***ON***
Allow unassigned issues	When turned **ON,** JIRA will allow issues to be unassigned or assigned to 'no-one'.When turned **OFF,** issues must always be assigned to someone - by default, the assignee will be the **Project Lead** as defined for each project. *Default:* ***OFF***
External user management	When turned **ON,** JIRA will assume that you are managing users from outside JIRA (e.g. using Crowd). This means that you will no longer be able to create, edit or delete users/groups from within JIRA (or via email or import); but you can still assign users/groups to project roles, and create/edit/delete user properties. Additionally, JIRA will not display options for users to change their password, or edit their profile. **Note:** JIRA's LDAP integration currently consists only of external *password* management, so this option should be left **OFF** when using LDAP. *Default:* ***OFF***

External password management	When turned **ON,** JIRA will assume that you are managing passwords from outside JIRA. JIRA will not display options for users to change their password, or display the 'Forgot Password' link on the login screen. **Note:** With the default 'osuser' LDAP provider, this option should be turned **ON,** as accounts are not yet stored in LDAP and this option only hides the password features within JIRA. *Default:* ***OFF***
Logout confirmation	Controls whether to obtain user's confirmation when logging out: **NEVER COOKIE** - prompt for confirmation if the user was automatically logged in (via a cookie). **ALWAYS** *Default:* ***NEVER***
Use gzip compression	Controls whether to compress the web pages that JIRA sends to the browser. It is recommended that this be turned ON, unless you are using mod_proxy. *Default:* ***OFF***
Accept remote API calls	Controls whether to allow remote client access (via XML-RPC or SOAP) to this JIRA installation, for authenticated users. *Default:* ***OFF***
User email visibility	Controls how users' email addresses are displayed in the user profile page. **PUBLIC** - email addresses are visible to all. **HIDDEN** - email addresses are hidden from all users. **MASKED** - the email address is masked (e.g. 'user@example.com' is displayed as 'user at example dot com'). **LOGGED IN USERS ONLY** - only users logged in to JIRA can view the email addresses *Default:* ***PUBLIC***
Comment visibility	Determines what will be contained in the list that is presented to users when specifying comment visibility and worklog visibility. **Groups & Project Roles** - the list will contain groups and project roles. **Project Roles only** - the list will only contain project roles. *Default:* ***Project Roles only***
Exclude email header 'Precedence: bulk'	Controls whether to prevent the **Precedence: Bulk** header on JIRA notification emails. This option should only be enabled when notifications go to a mailing list which rejects 'bulk' emails. In normal circumstances, this header prevents auto-replies (and hence potential mail loops). *Default:* ***OFF***
Issue Picker Auto-complete	Provides auto-completion of issue keys in the 'Issue Picker' popup screen. Turn OFF if your users' browsers are incompatible with AJAX. *Default:* ***ON***
User Searching By Full Name	Enables auto-completion of user names in the 'User Picker' popup screen. Also enables JQL-based searching of Full Names. Turn OFF if you have a very large number of users, or if your users' browsers are incompatible with AJAX.Note: If **'User email visibility'** (see above) is set to HIDDEN, the users' email addresses will not be searched or shown in the auto-complete results. *Default:* ***ON*** *if you have less than 5,000 users (or* ***OFF*** *if you have more than 5,000 users).*
JQL Auto-complete	Provides auto-completion of search terms when users perform an advanced (JQL) search. Turn OFF if you prefer not to use this feature, or are experiencing a performance impact. *Default:* ***ON***
Internet Explorer MIME Sniffing Security Hole Workaround Policy	Attachment viewing security options for cross-site site scripting vulnerabilities present in Internet Explorer 7 and earlier. Changes the default browser action for attachments in JIRA. Options are: • **Insecure: inline display of attachments** - allows all attachments to be displayed inline. Only select this option if you fully understand the security risks. See JIRA Security Advisory 2008-08-26 for further details. • **Secure: forced download of all attachments for all browsers** - force the download of all attachments. This is the most secure option, but is less convenient for users. • **Work around Internet Explorer security hole** - forced download of high-risk attachments (IE-only Workaround) - for IE browsers, force the download of attachments that IE would mistakenly detect as an HTML file. Declared HTML attachments are also never displayed inline. Use this option to reduce the risk of attacks to IE users via attachments. *Default:* ***Work around Internet Explorer security hole***

See Also

There are a handful of other properties (usually storing defaults of little interest to most JIRA users) in the **WEB-INF/classes/jira-application.properties** file, which you may want to edit. For details, please see Advanced JIRA configuration with jira-application.properties.

Setting properties and options on startup

This page describes how to set Java properties and options on startup, in various application servers.Java applications can be configured on startup by setting command-line options called system properties. For instance, one can set the default encoding with **file.encoding** or the

timezone with **user.timezone**, or language with **user.language** (and many others).

The general format for setting properties on the command-line is *-Dname=value*. For example, here is how the Orion application server is started, where the *file.encoding* property is set to *utf-8*:

```
java -Dfile.encoding=utf-8 -jar orion.jar -userThreads
```

Other app servers have startup and shutdown scripts. The same *-Dname=value* options are set under the covers, but the means of setting properties varies.

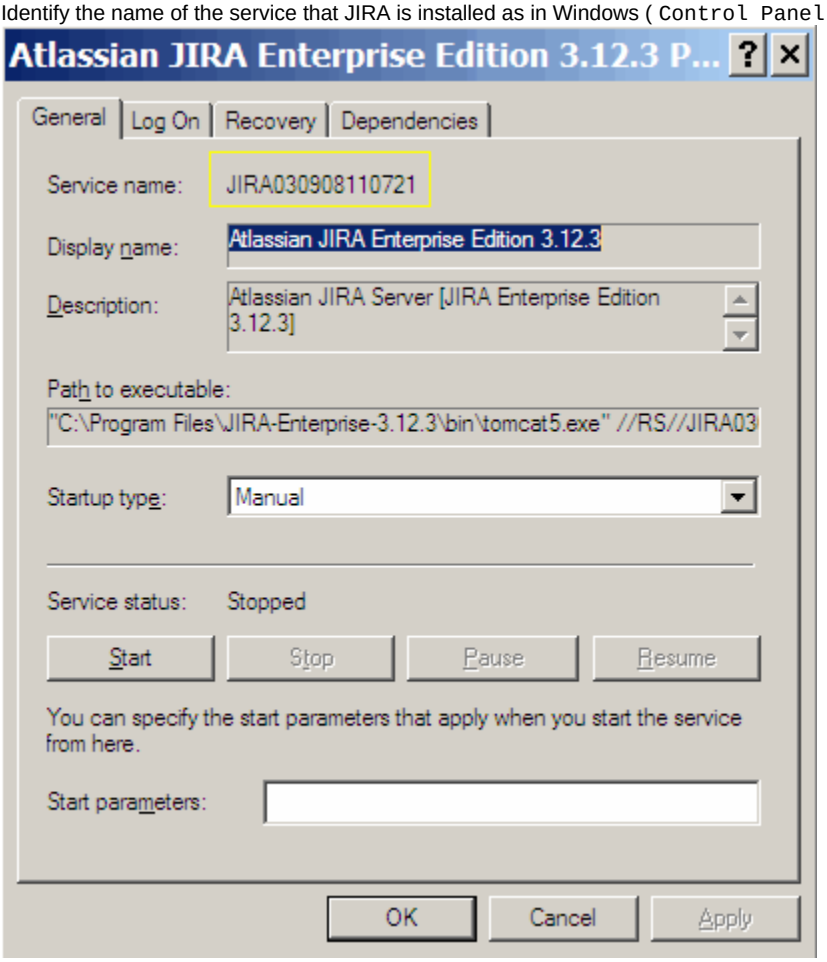
- Setting properties and options on startup
  - In Tomcat (eg. JIRA Standalone)
    - Tomcat installed as a Windows Service:
      - Modify via the Registry Editor
    - Tomcat on Windows, running startup.bat:
    - In Linux/*nix:
  - JBoss
  - Weblogic
  - Orion
  - Resin
  - Jetty
- JIRA Startup Properties

Setting properties and options on startup

In Tomcat (eg. JIRA Standalone)

Tomcat installed as a Windows Service:

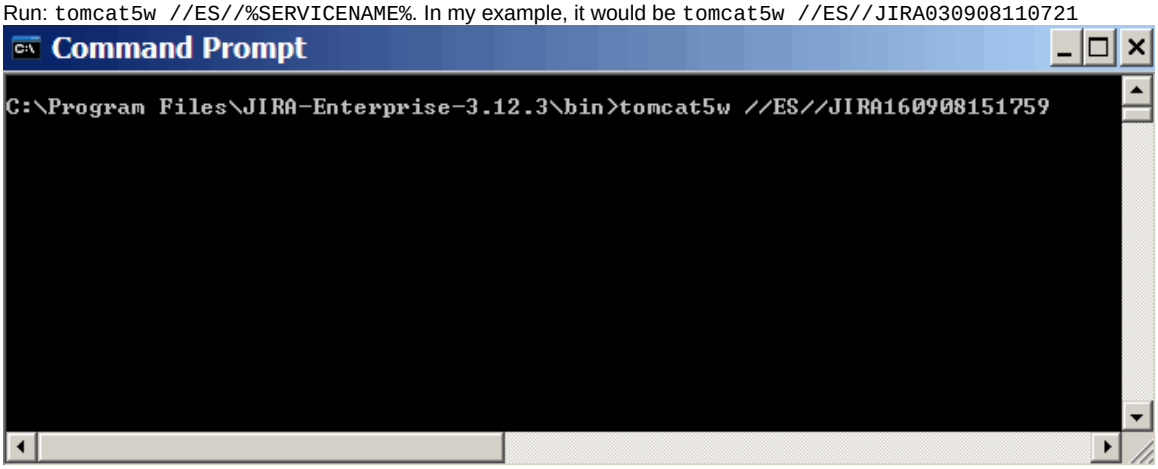
Identify the name of the service that JIRA is installed as in Windows (`Control Panel > Administrative Tools > Services`):



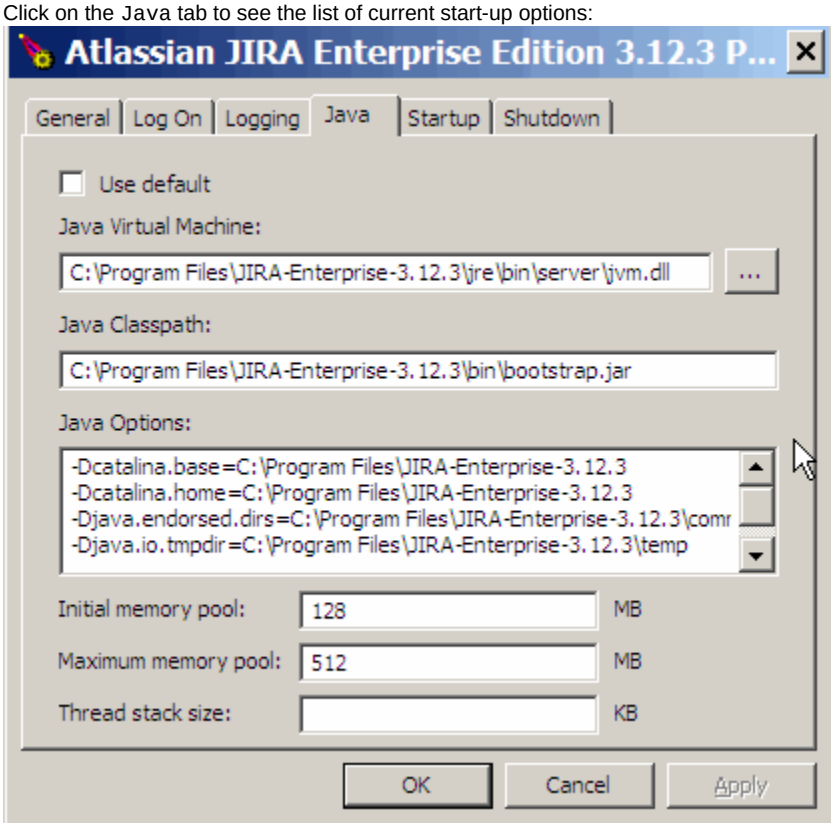
In the above example, the **SERVICENAME** is: `JIRA030908110721`

Open command window, and `cd` to the `bin` directory of your JIRA Standalone instance, or the `bin` directory of your Tomcat installation if your are running JIRA EAR/WAR.

Run: `tomcat5w //ES//%SERVICENAME%`. In my example, it would be `tomcat5w //ES//JIRA030908110721`



Click on the `Java` tab to see the list of current start-up options:



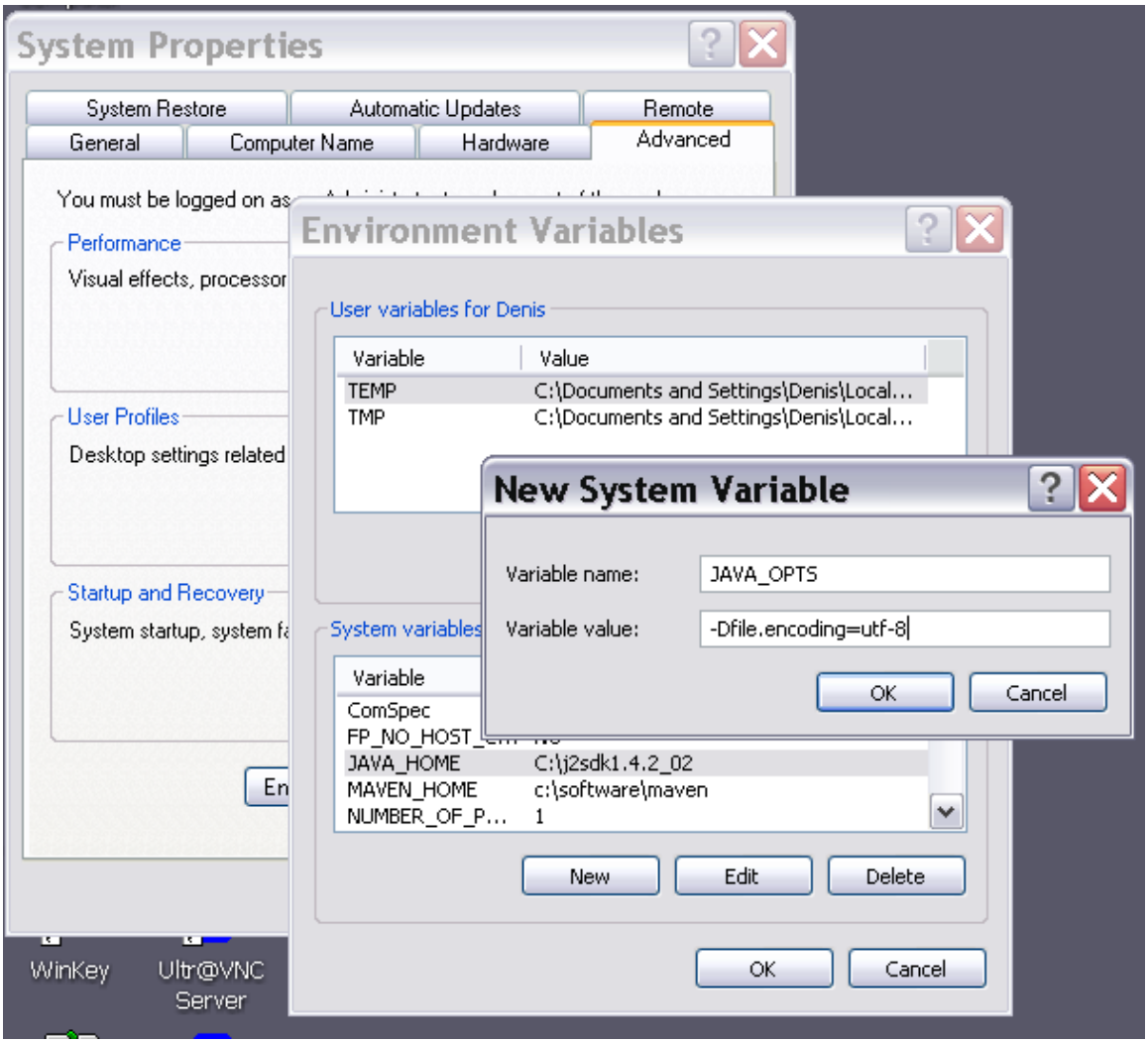
Append any new option on its own **new line** by adding to the end of the existing Java Options.

Modify via the Registry Editor

You can also check the properties by viewing the option list in the registry. Go to Start -> Run, and run "regedt32.exe". For JIRA, there should be an entry at HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Procrun 2.0 -> JIRA.

Tomcat on Windows, running startup.bat:

Right-click 'My Computer', go to Properties -> Advanced -> Environment Variables and set **JAVA_OPTS** to the desired value:



In Linux/*nix:

Set:

```
export JAVA_OPTS=-Dname1=value1 -Dname2=value2 ...
```

on the command-line before starting Tomcat, in your ~/.bash_profile or at the beginning of the the startup script (bin/setenv.sh for Linux or bin\setenv.bat for Windows).

Please note, if setenv.sh(bat) does not exist in your Tomcat, you can create it manually.

JBoss

The JBoss `run.sh` and `run.bat` also use the **JAVA_OPTS** variable, so see the above instructions for Tomcat. To set **JAVA_OPTS** in the JBoss script, edit `bin/run.conf` (Unix) or `bin/run.bat` (Windows).

Weblogic

Edit *$WEBLOGIC*/domains/*domain*/startWebLogic.sh (or .bat), and add properties to the end of the **JAVA_OPTIONS** value.

Orion

As shown above, simply specify *-Dname=value* after the 'java' command, before the '-jar' part.

```
java -Dfile.encoding=utf-8 -jar orion.jar -userThreads
```

Resin

Edit `bin/wrapper.pl`, and set the property in the `$JAVA_ARGS` variable.

Jetty

As with Orion, properties can be passed to Jetty directly on the command-line:

```
java -Dfile.encoding=utf-8 -jar start.jar
```

JIRA Startup Properties

JIRA also has a few internal startup options that can be set, some of these are used for debugging but a few may come in handy. Note that some of these are configurable via the Administration sections of JIRA.

```
java -Dmail.debug=false -Djira.jelly.on=true -Datlassian.mail.fetchdisabled=true -Djira.trackback.senddisabled=true
```

-Dmail.debug=(true|false) — will enable/disable SMTP mail debugging (but note this will only work in 3.7.2+).
-Djira.jelly.on=(true|false) — will enable/disable Jelly scripting
-Datlassian.mail.fetchdisabled=(true|false) — will enable/disable mail fetching services for IMAP and POP
-Datlassian.mail.senddisabled=(true|false) — will enable/disable JIRA sending mail
-Djira.trackback.senddisabled=(true|false) — will enable/disable trackback linking
-Djira.i18n.texthighlight=(true|false) — will show text elements of JIRA that are internationalised

Advanced JIRA configuration with jira-application.properties

In JIRA, most commonly accessed configuration items are editable from Administration -> Global Settings -> General Configuration. The default values here are set in the file `WEB-INF/classes/jira-application.properties`. In addition, jira-application.properties contains other properties which are not editable from the web interface. Usually these are of little interest to most users, but sometimes you may wish to edit them.

A sample jira-application.properties file is available here.

Making changes to jira-application.properties

The process of changing this file depends on whether you are running JIRA Standalone (JIRA deployed as an 'open' webapp) or JIRA deployed as a packed '.war' file deployed in an app server.

JIRA Standalone

The file is located in `atlassian-jira/WEB-INF/classes/jira-application.properties`. Edit the file here, and then restart JIRA by running `bin\shutdown.bat` and `bin\startup.bat` (or .sh equivalents).

JIRA as a webapp

If you have downloaded the .WAR/Webapp distribution of JIRA:

1. Copy `webapp/WEB-INF/classes/jira-application.properties` to `edit-webapp/WEB-INF/classes/jira-application.properties`
2. Make your changes in `edit-webapp/WEB-INF/classes/jira-application.properties`
3. In the root directory, run build.bat or build.sh to regenerate the webapp (.war file)
4. Redeploy the new .war file in your app server.

See also

Setting properties and options on startup — for changes like setting available memory, disabling email, enabling Jelly, etc.

Enabling File Attachments

When you enable file attachments, you allow users to attach files and screenshots to JIRA issues. This requires space on the server to save the attachments into. Note that attachments are not stored in JIRA's database and so will need to be backed up separately.

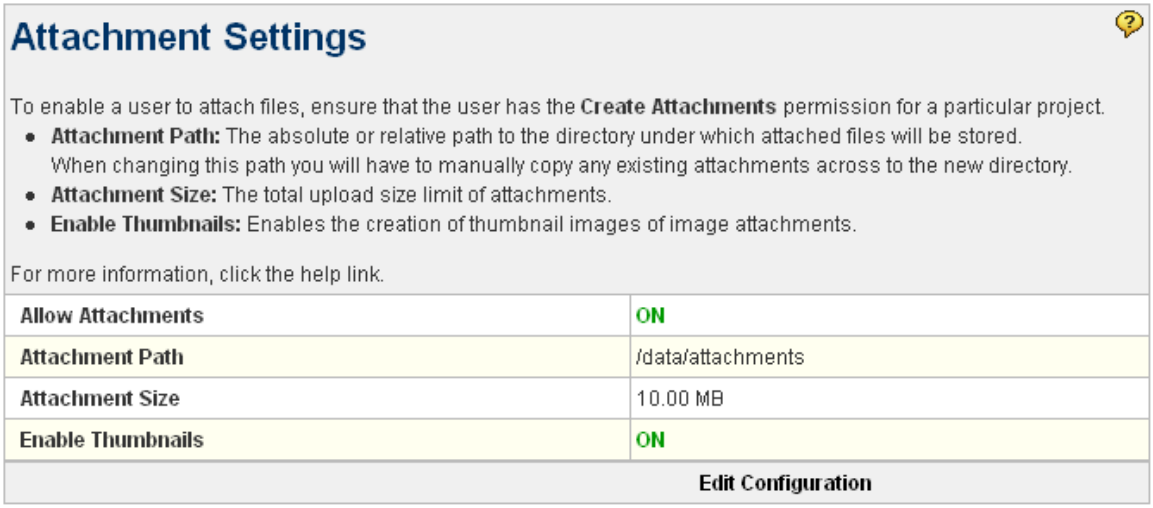
Configuring file attachments takes two steps:

1. Enabling attachments.
2. Granting the **'Create Attachments'** permission to appropriate users.

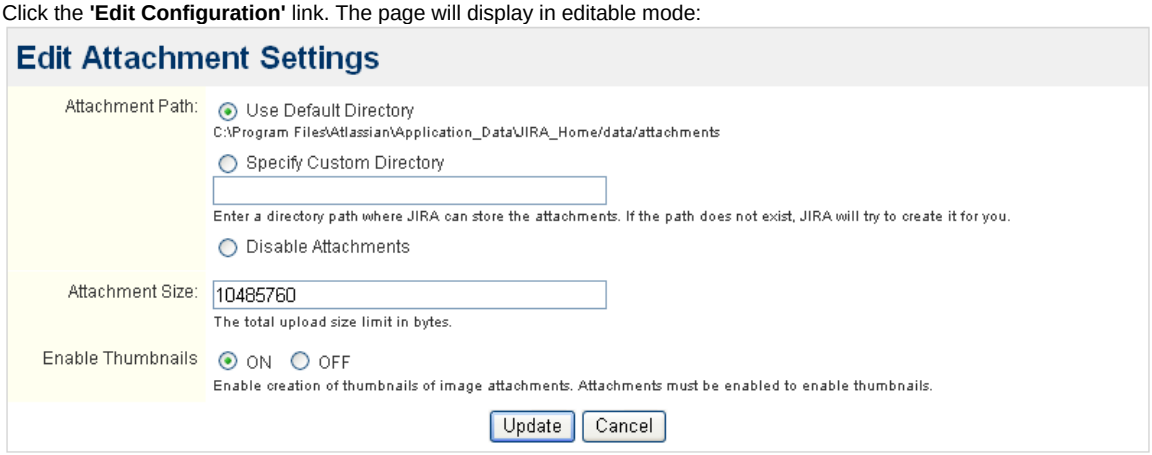
Additionally, if you wish to allow users to attach a file *when creating a new issue,* you need to ensure that the **'Attachment'** field is not hidden within the field configuration(s) associated with the specific issue type(s).

Step 1: Enabling attachments

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Global Settings'** sub-menu in the left-hand navigation column, click the **'Attachments'** link. This will display the **'Attachment Settings'** page, which states whether attachments are on or off:



4. Click the **'Edit Configuration'** link. The page will display in editable mode:



5. In the **'Allow Attachments'** field, select **'ON'.**
6. In the **'Attachment Path'** field, type the absolute or relative path to the directory where attachments will be stored. By default, your attachments are located in your JIRA home directory under the `data/attachments` sub-directory.

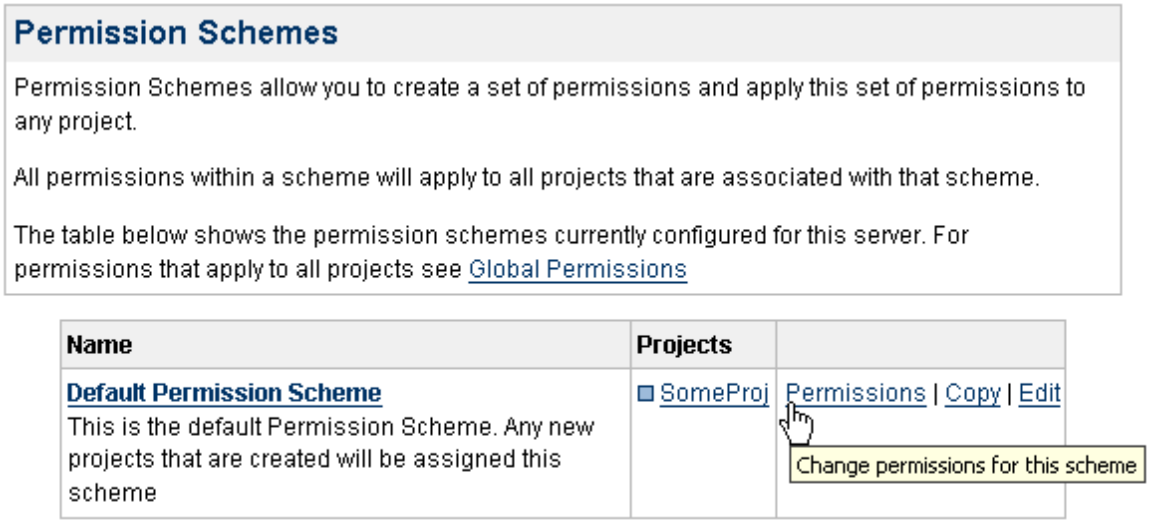
> **Changing your attachment path**
> If you are running a large instance of JIRA, you may wish to consider moving your attachments from the default directory under JIRA home to another machine. The directory which stores your attachments may need to hold a considerable amount of data. Hence, it may be more efficient for you to host the attachments on a separate network drive.
>
> If you change your attachments directory, please ensure that it is given appropriate security, as described in Security Overview.

7. In the **'Attachment Size'** field, specify the maximum attachment size. The default is 10485760 bytes (10 MB).
8. *(Optional)* In the **'Enable Thumbnails'** field, select **'ON'** if you wish to enable image attachments to be displayed as thumbnails. For details please see *' Image attachment thumbnails '.*
9. Click the **'Update'** button.

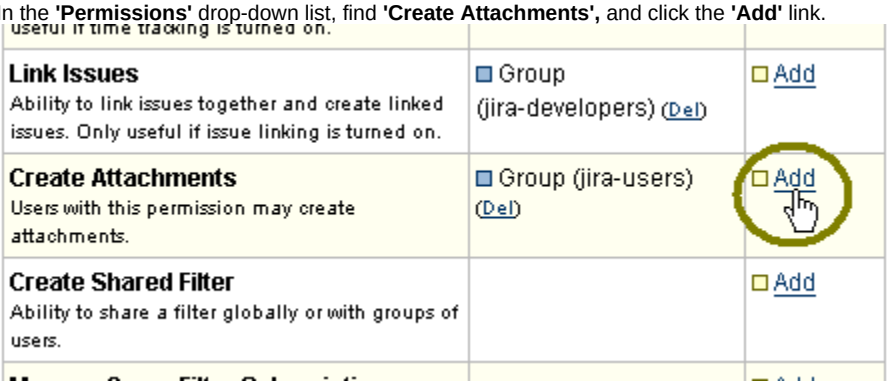
Step 2: Granting the 'Create Attachments' permission to users

You now need to grant the **'Create Attachments'** permission to appropriate users in the permission scheme(s) of project(s) for which you wish to allow attachments.

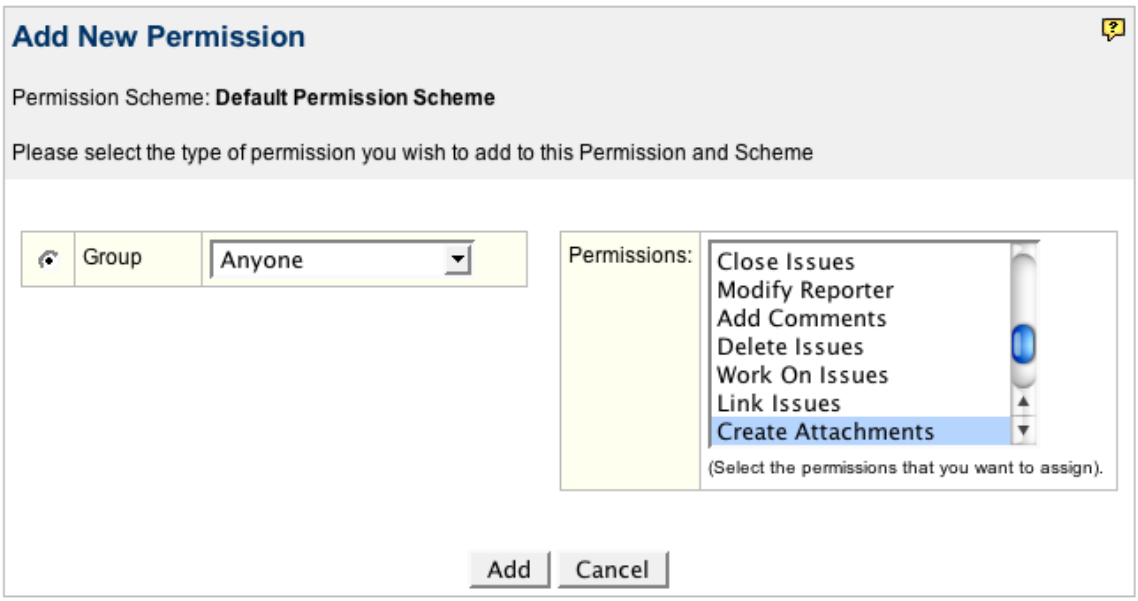
1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Schemes'** sub-menu in the left-hand navigation column, click the **'Permission Schemes'** link. This will display a list of all permision schemes in your JIRA system, and the projects which use each scheme.
4. For each relevant permission scheme,
    a. Click the **'Permissions'** link to edit the scheme.



    b. In the **'Permissions'** drop-down list, find **'Create Attachments',** and click the **'Add'** link.



    c. In the **'User-Group'** drop-down list, select the relevant group. Then click the **'Add'** button.



Specifying the maximum attachments per issue

JIRA allows multiple files to be attached to an issue in one operation. From the **'Attach File'** page, the user can toggle between multiple and single attachment screens by selecting the **'Attach multiple'** files link. The attachment form will retain the multiple/single attachment preference for that specific user for the duration of the user's session or until manually changed.

The number of attachment 'boxes' to be displayed on the multiple attachment screen is set to 3 by default. To change this, edit `jira.attachment.number` in the `jira-application.properties` file, then restart JIRA. For details, please see ' Advanced JIRA configuration with jira-application.properties '.

Enabling Thumbnails for Attachments

'Thumbnails' allow an image to be previewed in miniature, without having to download the original full-size image. JIRA can automatically create thumbnails for file attachments of the following types:

- GIF
- JPEG
- PNG

Once thumbnail functionality is enabled, thumbnails are displayed to users in the **'Image Attachments'** section when viewing an issue. (All other types of file attachments are listed in the **File Attachments** section.) You can also configure the Issue Navigator column layout to display the thumbnails in an `Images` column.

All thumbnail images are stored in JPEG format in the `attachments` directory (see *' Configuring File Attachments '),* together with the original attachments. The thumbnail images are denoted by ' *`thumb`* ' in the filename.

Thumbnail image generation requires the following:

1. The system must have X11 support. This web page details the minimum set of libraries needed to use JDK 1.4.2 under RedHat Linux 9.0.
2. The following java system property must be set: `-Djava.awt.headless=true` .

Enabling thumbnails

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Global Settings'** sub-menu in the left-hand navigation column, click the **'Attachments'** link. This will display the **'Attachment Settings'** page, which states whether attachments are currently on or off.
    - **Note:** attachments must be enabled in order to enable thumbnails. Attachments can only be enabled by people who have the **'JIRA System Administrators'** global permission.
4. Click the **'Edit Configuration'** link.
5. In the **'Enable Thumbnails'** field, select **'ON'.**

6. Click the **'Update'** button.

Configuring thumbnail size

By default, thumbnails are 200 pixels wide and 200 pixels high. To change the dimensions of thumbnail images:

1. Stop JIRA.
2. Edit the following values found in the file `jira-application.properties` :
    - `jira.thumbnail.maxwidth`
    - `jira.thumbnail.maxheight`
3. Delete all existing thumbnail images within the `attachments` directory (denoted by '`*_thumb_*`' in the filename).
4. Restart JIRA. All thumbnails will be automatically recreated using the new dimensions.

Enabling Sub-tasks

Sub-task issues are generally used to split up a parent issue into a number of tasks which can be assigned and tracked separately.

Sub-tasks have all the same fields as standard issues, although note that their 'issue type' must be one of the *sub-task issue types* (see below) rather than one of the standard issue types.

Once you have enabled sub-tasks and defined at least one sub-task issue type, users will be able to:

- create sub-tasks.
- convert issues to sub-tasks (and vice versa).

Note
Sub-tasks are only supported in the Professional and Enterprise editions of JIRA.

On this page:

- Enabling sub-tasks
- Defining sub-task issue types
- Configuring sub-tasks and workflow
- Configuring sub-task fields
- Disabling sub-tasks

Enabling sub-tasks

Sub-tasks are disabled by default. To enable sub-tasks:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Locate the 'Global Settings' sub-menu on the left hand side of the screen, and choose *Sub-Tasks* from the list.
4. The 'Sub-Tasks' administration screen will be displayed. Click the 'Enable' link.
5. The page will reload and inform you that the sub-tasks are now enabled. A default JIRADOC:sub-task issue type has also been automatically created. You can edit it by selecting the *Edit* link in the *Operations* column.

Defining sub-task issue types

Sub-tasks must be assigned one of the *sub-task issue types,* which are different to standard issue types. Please note that you must define at least one sub-task issue type before users can create sub-tasks.

Sub-task issue types can be customised on the 'Sub-Tasks' administration screen (which is described above). The 'Sub-Tasks' administration screen allows you to create, delete, edit, translate and choose icons for your sub-task issue types. For details, please see ' Defining Issue Types '.

Configuring sub-tasks and workflow

It is possible to restrict the progression of an issue through workflow depending on the state of the issue's sub-tasks. For example, you might need to restrict an issue from being resolved until all of its sub-tasks are resolved. To achieve this, you would create a custom workflow and use the *Sub-Task Blocking Condition* on the workflow transitions that are to be restricted by the sub-tasks' state.

Configuring sub-task fields

You can choose which subtask fields are displayed in the 'Sub-Tasks' section of an issue (see Working with sub-tasks '), i.e.:

- the parent's 'View Issue' screen.

- the sub-task 'Quick Create' form.

This is done via the jira-application.properties file.

Disabling sub-tasks

Sub-tasks are disabled from the sub-task administration screen. To disable sub-tasks please follow the following steps:

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Open the *Global Settings* sub-menu on the left hand side if it is not open already, and choose *Sub-Tasks* from the list.
4. After the *Sub-Tasks* administration screen loads please click the "Disable" link. The page should reload and inform you that the sub-tasks are now disabled.

> **Note**
> Sub-tasks cannot be disabled if one or more sub-tasks exist in the system. You will need to remove the existing sub-tasks (or convert them to standard issues) before disabling this feature.

Enabling Issue Linking

Issue Linking allows you to create an association between issues. For instance, an issue may *duplicate* another, or its resolution may *depend* on another's.

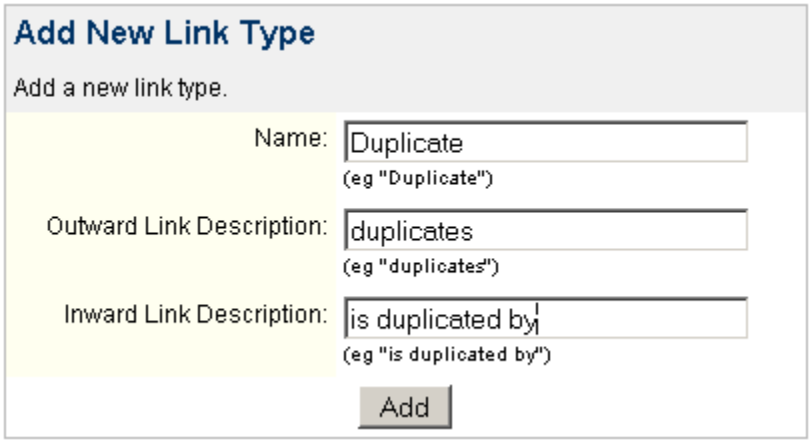
There are two steps involved in configuring 'Issue Linking'. The first step is configuring JIRA to allow issue links. This is a global setting. The second step is to the create the issue link types required. These are also global.

Step 1: Turning on Issue Linking

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title '**Global Settings**', click on the link labelled 'Issue Linking'.
4. You will be shown a status page stating whether linking is enabled. If it is not, click the '**Activate**' button.
5. You will be shown the status page again, this time stating that Issue Linking is enabled.

Step 2: Creating Issue Link Types

1. On the Issue Linking page there will now be a form titled 'Add New Link Type'.

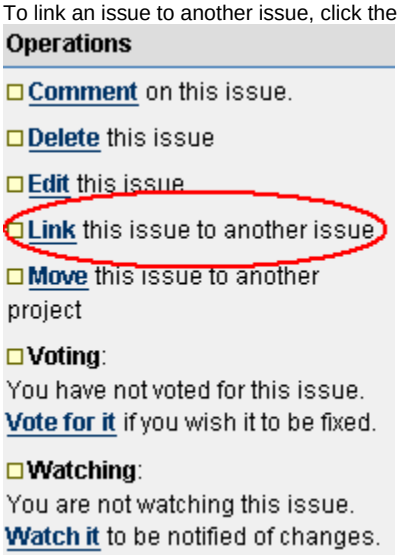


2. To create a new link type, say Duplicate, proceed as follows:
   - enter "Duplicate" in the '**Name**' text field
   - enter "duplicates" in the '**Outward Link Description**' text field
   - enter "is duplicated by" in the '**Inward Link Description**' text field
3. Click the '**Add**' button.
4. This returns to the link type management page, with a new section listing the "Duplicate" issue linking type. Here you can edit or delete the relationship, as required.

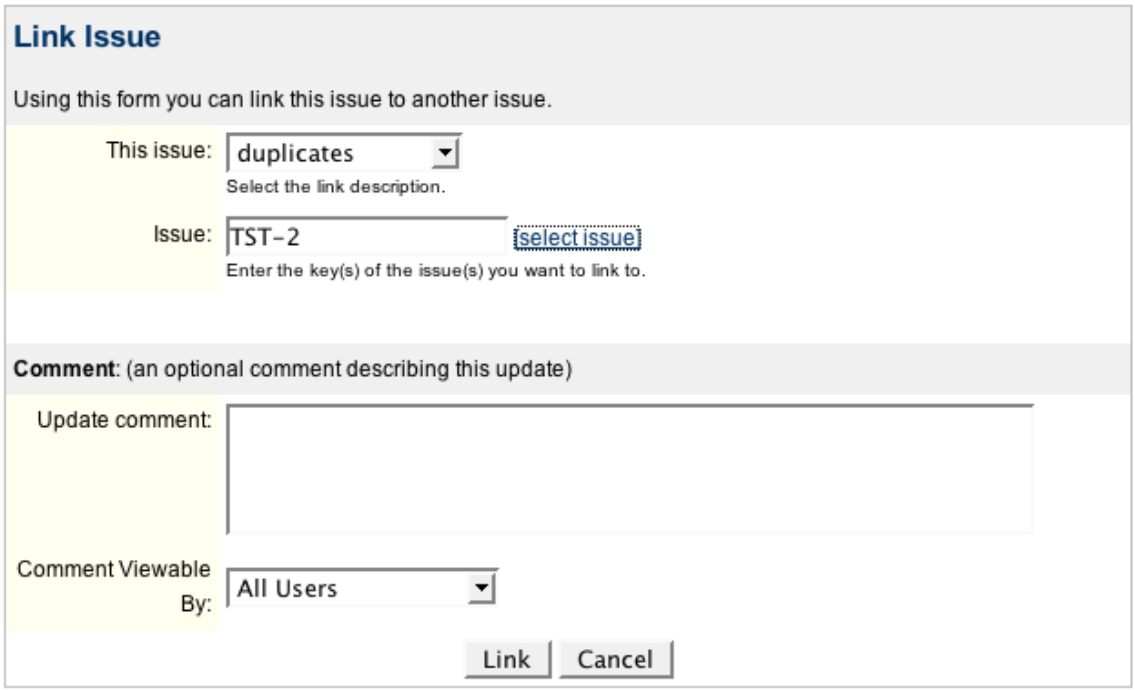
Name	Outward Desc.	Inward Desc.	
Duplicate	duplicates	is duplicated by	Edit | Del

Step 3: Linking Issues

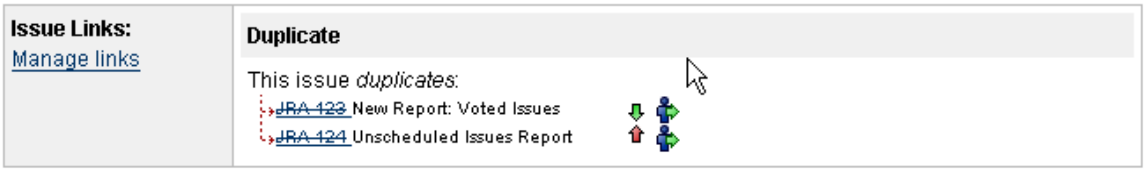
1. To link an issue to another issue, click the link '**Link this Issue to Another Issue**' on the 'View' Issue page.



2. This will bring up the 'Link Issue' form.



3. Select the link type and enter the key of the issue that you want to link to. It is also possible to link to multiple issues. You can optionally add a comment.
   Click on the '**Link**' button.
4. You will see the issue page again, with a new section listing the issues that are linked to this issue

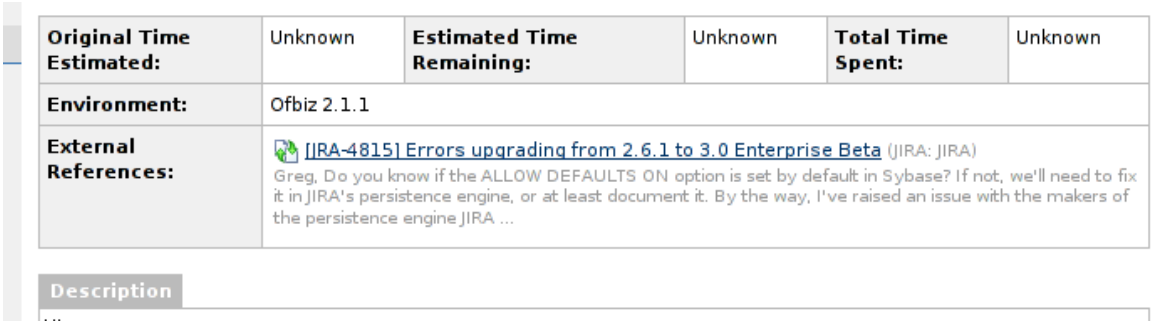


Enabling Trackback

Trackback linking is a means by which a page can *tell another page* that it has been linked to. (To learn more about how trackback works, please have a look at the Trackback specification).

For instance, say that a user writes a URL in a JIRA comment:

!trackback-source.png|title="Comment containing URL to trackback-enabled site"!If the URL is to a trackback-enabled web application like a weblog, Confluence page or another JIRA site, the linked-to page will be told that it was linked to, and can automatically create a link back to the linker:



Original Time Estimated:	Unknown	Estimated Time Remaining:	Unknown	Total Time Spent:	Unknown
Environment:	Ofbiz 2.1.1				
External References:	[JRA-4815] Errors upgrading from 2.6.1 to 3.0 Enterprise Beta (JIRA: JIRA) Greg, Do you know if the ALLOW DEFAULTS ON option is set by default in Sybase? If not, we'll need to fix it in JIRA's persistence engine, or at least document it. By the way, I've raised an issue with the makers of the persistence engine JIRA ...				

Description

...

Note
When linking *to* a JIRA instance, make sure its **base URL is correct.** (To check a JIRA instance's 'Base URL', go to **'General Configuration'** under **'Global Settings'** in the **'Administration'** menu.)

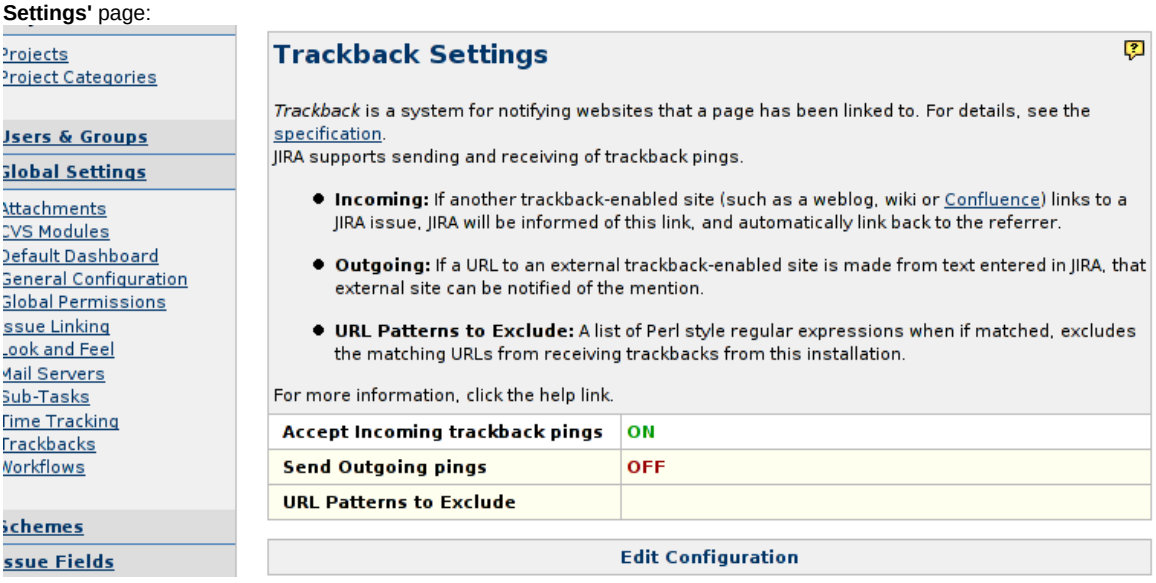
Configuring trackbacks

In JIRA, you can configure whether to:

- display links to external pages that link to your pages (accept incoming trackback pings)
- notify external pages that they have been linked to (send outgoing pings)
  The default configuration is to display incoming links, but not notify external pages.

To configure trackbacks:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Global Settings'** sub-menu in the left-hand navigation column, click the **'Trackbacks'** link. This will display the **'Trackback Settings'** page:



4. Click the **'Edit Configuration'** link.
5. In the **'Accept Incoming Trackback Pings'** field, select **'ON'** to enable trackbacks.
6. In the **'Send Outgoing Trackback Pings'** field, either:
   - select **'On for all issues'** to always notify external sites that they have been linked to.
   - select **'On for public issues only'** to only notify external sites that they have been linked to if the issue is publicly visible.
   - select **'Off'** to never notify external sites that they have been linked to.

7. *(Optional)* In the **'URL Patterns to Exclude'** field, specify the URLs of any sites which you always want to exclude from being notified that they have been linked to. Use regular expressions (one per line), e.g.:
   `.server.domain.com..server2.domain.com..domain2.com.`
8. Click the **'Update'** button.

Temporarily disabling trackbacks

Trackback pings can be disabled (e.g. during a data import) by setting the **jira.trackback.senddisabled=true** flag on startup.

Enabling Time Tracking

To enable time tracking in JIRA, you must first activate it and then assign permissions for users to log work on issues.

Step 1: Activating Time Tracking

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title "Global Settings", click on the link labelled **Time Tracking**'. By default, time tracking is OFF:

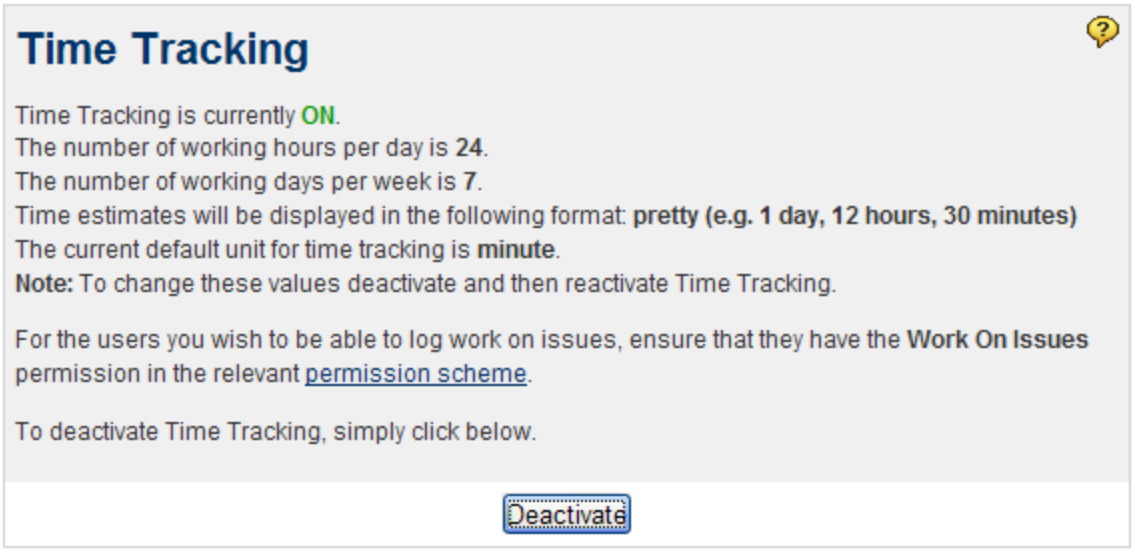
4. Select a suitable value for **Hours Per Day** (e.g. 8)
5. Select a suitable value for **Days Per Week** (e.g. 5)
6. Select your preferred **Time Format.** This will determine the format of the 'Time Spent' field when an issue is displayed.
7. Select your preferred **Default Unit** (minutes/hours/days/weeks). This will be applied whenever your users log work on an issue without specifying a unit.
8. Click **Activate** to turn time tracking ON.

> To change the **Hours Per Day** and **Days Per Week** once Time Tracking is activated, you will need to deactivate and then reactivate Time Tracking with the new values.

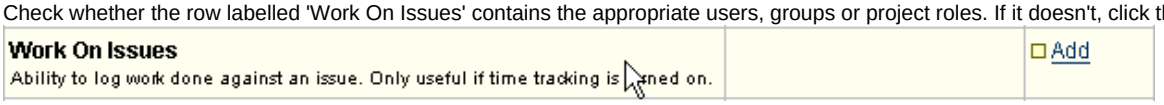
Step 2: Allowing users to log work

To be able to log work on issues, users, groups or project roles must first be assigned permissions to the appropriate project(s) as follows:

1. Once you have activated time tracking (see above) you will see the following screen:



2. Click the '**Permission Schemes**' link.
3. Select the permission scheme associated with the project(s) for which you wish to allow work to be logged.
4. Check whether the row labelled 'Work On Issues' contains the appropriate users, groups or project roles. If it doesn't, click the **Add** link:



5. Then select the user, group or project role you wish to be able to log work on issues. Then click the '**Add**' button.

Configuring OAuth Consumers

An OAuth 'consumer' is an application that accesses ('consumes') data from another application. When you add an OAuth consumer in JIRA, you are allowing the consumer application to access JIRA's data. For example, if you want your users to be able to add JIRA gadgets to their iGoogle homepages, then you will need to add iGoogle as an OAuth consumer.

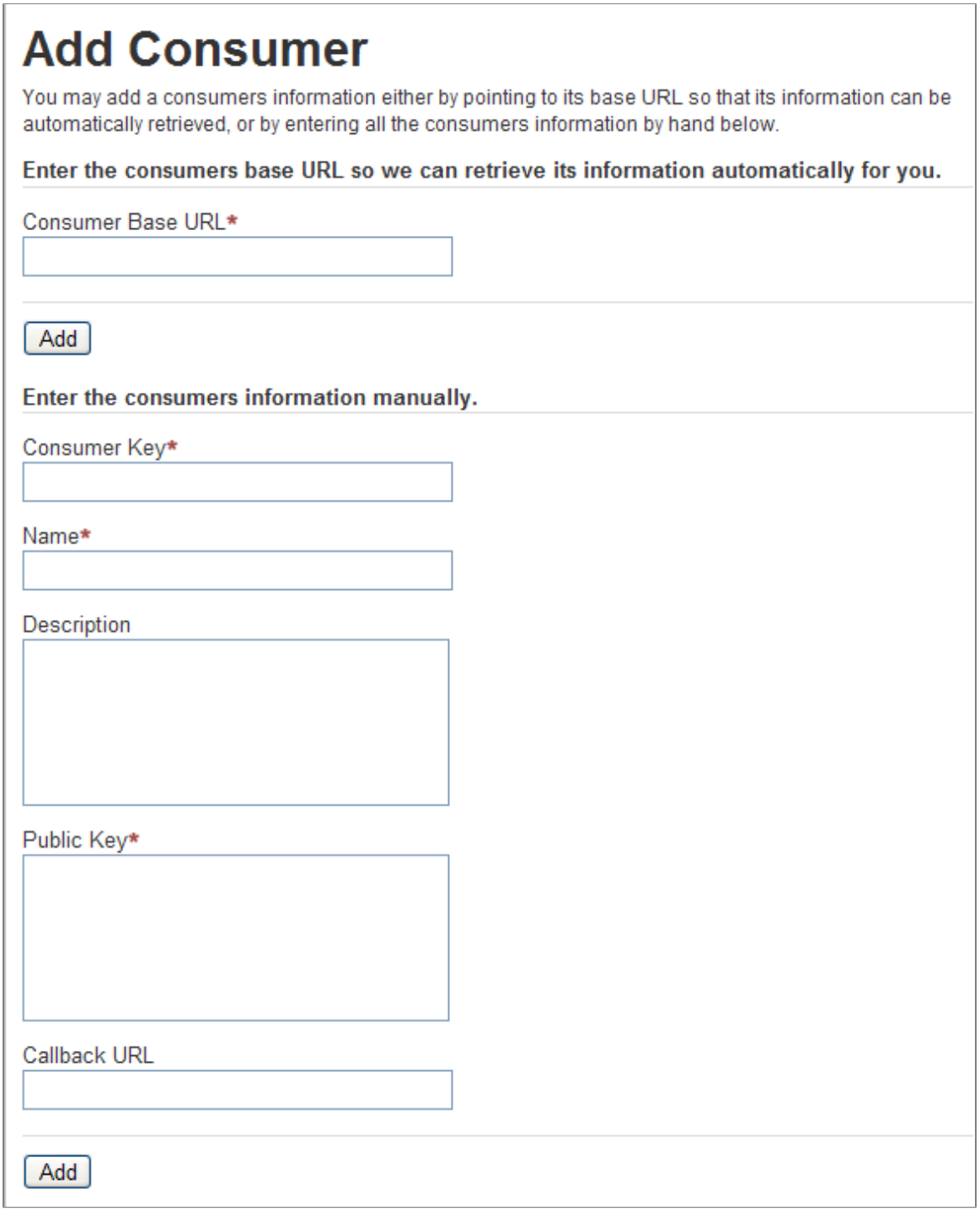
OAuth consumers are a potential security risk. Do not add an OAuth consumer unless you trust all code in the consumer application to behave itself at all times.

Adding an OAuth consumer

Before you begin: Note that adding an OAuth consumer requires the transmission of sensitive data. To prevent 'man-in-the-middle' attacks, it is recommended that you use SSL while adding a OAuth consumer.

To add an OAuth consumer,

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. In the left-hand panel, under the title **'System',** click the **'OAuth Consumers'** link. The **'OAuth Consumers'** page will be displayed, showing a list of configured OAuth Consumers (if any exist).
4. Click the '**Add Consumer**' link at the bottom of the list. The '**Add Consumer**' page will be displayed:



5. You can either:
    a. If the consumer is another Atlassian application (e.g. Bamboo), type the consumer's root URL in the '**Consumer Base URL**' field (e.g. "http://bamboo.mycompany.com"), so that JIRA can automatically look up the consumer's details for you;
    OR:
    b. Enter the consumer's details yourself:
        i. '**Consumer Key**' — Type the consumer's unique identifier (e.g. for iGoogle, type "www.google.com").
        ii. '**Name**' — Type a short name that is meaningful to you and your end-users (e.g. "iGoogle").
        iii. '**Description**' *(optional)* — Type a longer description if you wish.
        iv. '**Public Key**' — Paste the consumer's RSA certificate, e.g. you can copy the iGoogle one from here: http://code.google.com/apis/gadgets/docs/oauth.html#rsa
        v. '**Callback URL**' *(optional)* — Type the URL of the page that JIRA will redirect to after an end-user has approved the Oauth request, to let the consumer application (e.g. iGoogle) know about the result of the approval. E.g. for iGoogle, type "http://oauth.gmodules.com/gadgets/oauthcallback".
6. Click the '**Add**' button.

See also

Adding an Atlassian Gadget to iGoogle and Other Web Sites

Server Administration

- Search Indexing
- Backing Up Data
- Restoring Data
- Using the Database Integrity Checker
- Optimising Performance
- Precompiling JSP pages
- Logging and Profiling
- Increasing JIRA Memory
- Database Indexing
- Generating a Thread Dump
- Using robots.txt to hide from Search Engines

Search Indexing

In order to provide fast searching, JIRA creates an index of the text entered into issue fields. This index is stored on the file system, and updated whenever issue text is added or modified. It is sometimes necessary to regenerate this index manually; for instance if issues have been manually entered into the database, or the index has been lost or corrupted.

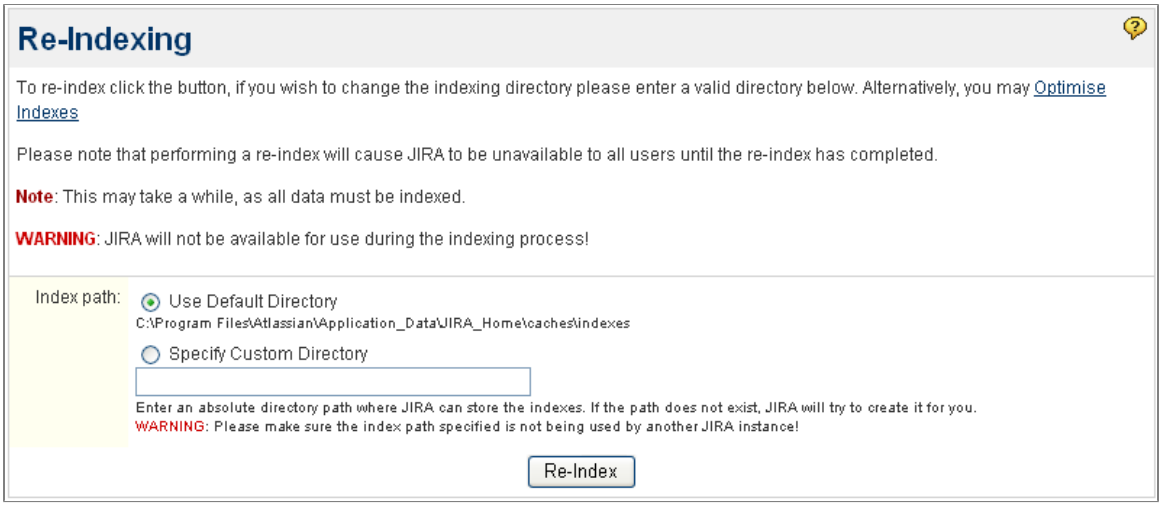
Indexing Administration

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu on the left, click the **'Indexing'** link.
4. This page allows you to:
    - **re-index your data** — whenever you re-index data, JIRA will clear any existing indexes and re-index all the current data from scratch. This may take a few minutes, depending on how many issues you have, and users will be unable to access JIRA during this time.
        - Note that JIRA by default schedules a re-index to run every night at midnight, through the use of Services.
        - See also Re-Indexing after Major Configuration Changes.
    - **optimise your indexes**
    - **move your indexes to another directory** — by default, your indexes are located in your JIRA home directory under the `caches/indexes` sub-directory. Note that only people with the **'JIRA System Administrators'** global permission can move the index.

Performance Note
If you are running a high load JIRA instance, you may wish to consider moving your indexes from the default directory under JIRA home to another machine. The directory which holds your indexes is heavily accessed while JIRA is running. Hence, your JIRA instance may run more efficiently by hosting the indexes on the fastest disk available and reducing the number of other applications using the same disk to reduce access contention.

If you move your index directory, please ensure that it is given appropriate security, as described in Security Overview.

Screenshot: Indexing JIRA



Re-Indexing after Major Configuration Changes

Once issues have been created, modifying the configuration of your JIRA instance can result in the search index becoming out-of-sync with JIRA's configuration. Configuration details such as the following can affect the search index:

- Field Configuration Schemes
- Custom Fields
- Plugins
- Time Tracking

If you make changes to any of these areas of configuration, you might see the following message in your Administration view:

```
USERFULLNAME made configuration changes to 'SECTION' at TIME. It is recommended that you perform a re-index.
```

The above message means that configuration changes have been made to JIRA, but have not yet been reflected in the search index. Until JIRA's search index has been rebuilt, it is possible that some search queries from JIRA will return incorrect results. For example:

- If a plugin containing a custom field is enabled after being disabled, search queries which specify that the custom field should be empty will return *no issues* instead of *all issues*.
- If a Field Configuration is modified by altering the visibility of a particular field, search queries which specify that field may also return erroneous results (depending on which field is being modified and what query is being executed).
- etc

The way to resolve the discrepancy is to rebuild JIRA's search index . This can take anywhere from seconds to hours, depending on the number of issues and comments in your JIRA instance. While re-indexing is taking place, your instance will be unavailable to all users. For these reasons, it is recommended that you:

- Make all your necessary configuration changes in one go before starting the re-index process; and
- Start the re-index process in a time period of low activity for your instance.

Backing Up Data

This page describes how to back up your JIRA data, and establish processes for maintaining continual backups. Backing up your JIRA data is the first step in upgrading your server to a new JIRA revision, or splitting your JIRA instance across multiple servers. See also *Restoring JIRA data* and *Restoring a Project from Backup.*

Creating a complete backup of JIRA consists of two stages:

1. Backing up the data in the database
2. If attachments are enabled, backing up the attachments directory

1. Backing up database contents

There are two possibilities: native database-specific tools, or JIRA's XML backup utility.

For production use, it is **strongly recommended to use native database-specific tools.** The XML backup is not guaranteed to be consistent, as the database is not locked during the backup process.

Using native database tools

All serious databases come with tools to back up and restore databases (the 'MS' inRDBMS). We strongly recommend these tools in preference to the XML backup option described below, as they:

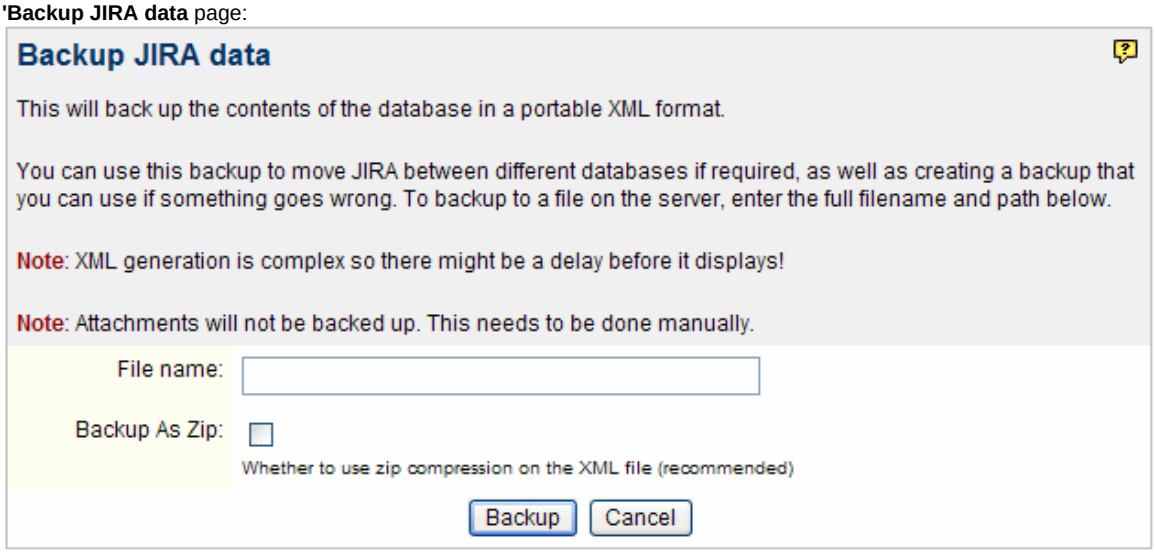
- ensure integrity of the database by taking the backup at a single point in time
- are much faster and less resource-intensive than JIRA's XML backup.
- integrate with existing backup strategies (e.g. allowing one backup run for all database-using apps).
- may allow for incremental (as opposed to 'full') backups, saving disk space.
- avoid character encoding and format issues relating to JIRA's use of XML as a backup format.

See the documentation for your database on how to set up periodic backups. This typically involves a cron job or Windows scheduled task invoking a command-line tool like mysqldump or pg_dump,

Using JIRA's XML backup utility

To perform a once-off backup, e.g. before an upgrade, follow the steps below. (Note that you can also configure scheduled XML backups, as described in *Automating JIRA Backups.*)

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Import & Export'** sub-menu in the left-hand navigation column, click the **'Backup Data to XML'** link. This will display the **'Backup JIRA data** page:



4. In **'File name'** field, type the full path, including filename, of the location to which you want JIRA to write the backup file. **Note: Ensure that JIRA has rights to write to this location.**
5. *(Optional but recommended)* Select the **'Backup as Zip'** checkbox.
6. Click the **'Backup'** button, and wait while your JIRA data is backed up.
7. When the backup is complete, a message will be displayed, confirming that JIRA has written the data to the file you specified.

2. Backing up attachments

If you have attachments enabled you also need to create a backup of the attachments directory, as the attachments do not get stored in the database. By default, file attachments are stored in the JIRA home directory under the `data/attachments` sub-directory.

To back up attachments, you need to create a snapshot of the attachment directory (including all files and subdirectories). Note that the directory structure under the attachments directory **must** be preserved in the snapshot.

Creating this snapshot is an operating system-specific task, e.g.:

- On MS Windows, a batch script copying the directory can be written and scheduled periodically (Programs > Accessories > System Tools > Scheduled Tasks). There are also various utilities available to simplify this (eg. http://www.picozip.com).
- On Linux/Solaris, it is best to write a small shell script, placed in `/etc/cron.daily` , backing up files to a directory like `/var/backup/jira` . It is best to copy an existing script in `/etc/cron.daily` to ensure local conventions (file locations, lockfiles, permissions) are adhered to.

Automating JIRA Backups

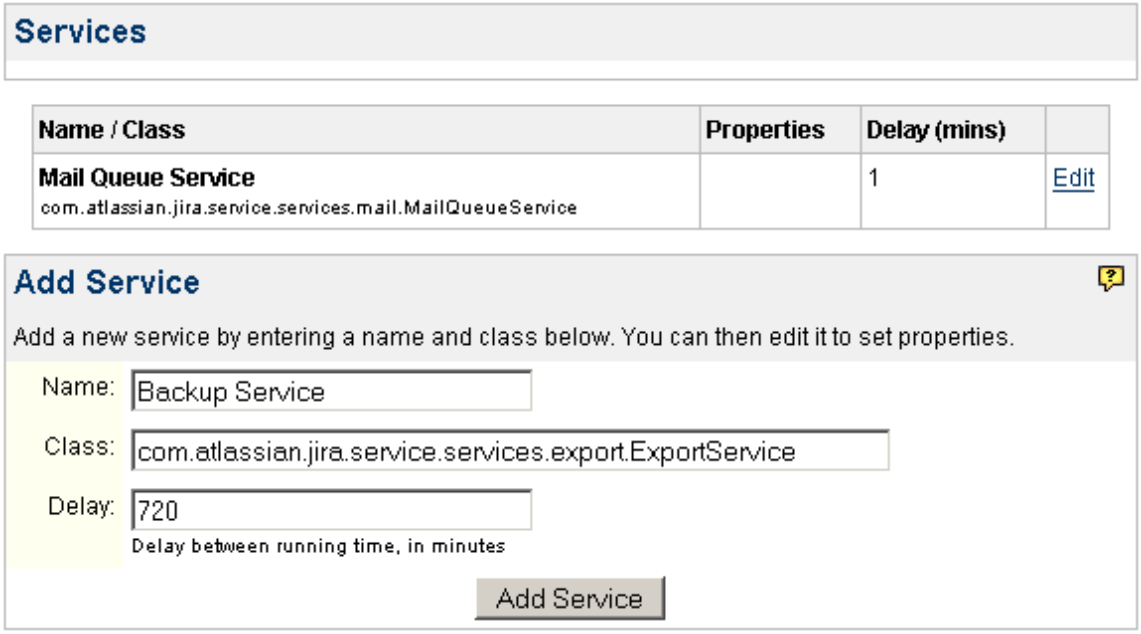
JIRA can be configured to automatically create an XML backup of JIRA data on a routine basis.

The XML backup includes all data in the database. However, it **does not include** attachments, which are on the filesystem.

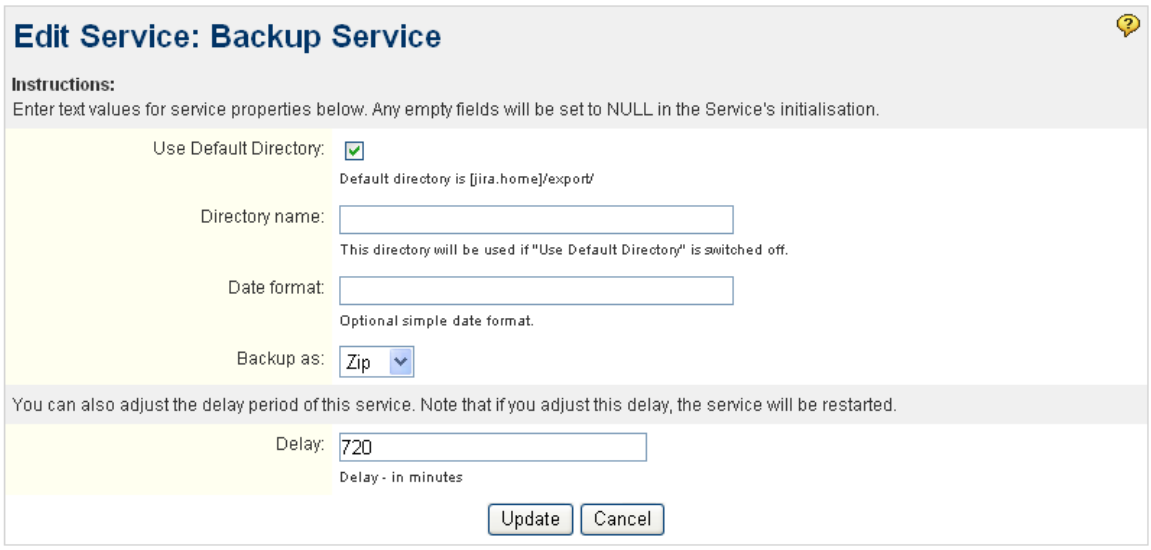
For production use, it is **strongly recommended** to use native database-specific tools instead of the XML backup service. The XML backup is not guaranteed to be consistent, as the database is not locked during the backup process.

When JIRA is installed, the first step in the Setup Wizard prompts you for a backup path, and if entered, JIRA will automatically generate XML backups (as ZIP files) every 12 hours. If you did not specify this path, follow the steps below to enable automated backups. (Note that you can also perform XML backups manually - see *Backing up data.*)

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu in the left-hand navigation column, click the **'Services'** link.
4. This will display the **'Services'** page. It lists the current services running on this system. By default there should be one service running: **'Mail Queue Service'.** You cannot delete this service.
5. Fill in the **'Add Service'** form as follows:



   - For **'Name',** enter a descriptive name, e.g. **Backup Service**
   - For **Class,** enter **com.atlassian.jira.service.services.export.ExportService**
   - For **'Delay',** enter the number of minutes between backups. A good default for this would be 720 minutes (12 hours).Click the **'Add Service'** buttont.
6. This will display the **'Edit Service'** screen. Fill in the following fields:



- If you wish to write your backup files to the the default directory, leave the '**Use Default Directory**' checkbox checked. By default, your backup files are written to your JIRA home directory under the `exports` sub-directory.
- If you wish to specify a different sub-directory, type the full path of the directory to which JIRA will write backup files in the **'Directory Name'** textbox. Ensure that the '**Use Default Directory**' checkbox is unchecked and that JIRA has rights to write to this location.

> **Changing your backups path**
> If you are running a large instance of JIRA, you may wish to consider moving your backups directory from the default directory under JIRA home to another machine. The directory which stores your backups may need to hold a considerable amount of data. Hence, it may be more efficient for you to host the attachments on a separate network drive.
>
> If you change your backups directory, please ensure that it is given appropriate security, as described in Security Overview.

- For **'Date format',** specify the format which JIRA will use to name the individual backup files. This format can be anything that SimpleDateFormat can parse. A good default is 'yyyy-MMM-dd-HHmm', which would generate files named like this: '2007-Mar-05-1322'.
- For **'Backup as:',** either:
  - Select **'XML'** to have JIRA store your data as an XML file; or
  - Select **Zip'** to have JIRA to compress your backup and store it as a ZIP file.

7. Your backup service is now configured. XML backups will be performed according to the schedule you specified in the **Delay** field (above).
   - For every successful backup, an XML (or ZIP) file will be saved in the backup directory that you specified in the **Directory Name** field (above).
   - If a scheduled backup fails for any reason, the XML (or ZIP) file will be saved into the 'corrupted' directory, which is directly under your nominated backup directory. (NB. JIRA will created the 'corrupted' directory if required - you don't need to create it.) Additionally, a file explaining the reason for the failure will be written to the 'corrupted' directory. This file will have the same name as the backup file, but with the extension '.failure.txt'.

Restoring Data

This page describes how to restore JIRA data from a backup. This is the second step in either upgrading your server to a new JIRA revision, or splitting your JIRA instance across multiple servers.

If you wish restore a single project from your backup into an existing JIRA instance, refer to these instructions on restoring a project from backup instead.

> When restoring data, **all data in the existing JIRA database is deleted,** including all user accounts. Before you begin, make sure you have the password to a login in the backup file that has the 'JIRA System Administrator' global permission.

Restoring JIRA from backup is a three stage process:

1. *(Optional)* Disable email sending/receiving
2. Restore data from XML to the database
3. *(Optional)* Restore the attachments to the attachments directory (if attachments were backed up)

1. Disabling email sending/receiving

If you are restoring production data into a *test* JIRA instance for experimentation purposes, you probably want to disable JIRA's email interaction features before you begin:

- **Disable email notifications** — if JIRA is configured to send emails about changes to issues, and you want to make test modifications to the copy, you should start JIRA with the `-Datlassian.mail.senddisabled=true` flag.
- **Disable POP/IMAP email polling** — if JIRA is configured to poll a mailbox (to create issues from mails), you will likely want to disable this polling on your test installation. This is done by setting the `-Datlassian.mail.fetchdisabled=true` flag.

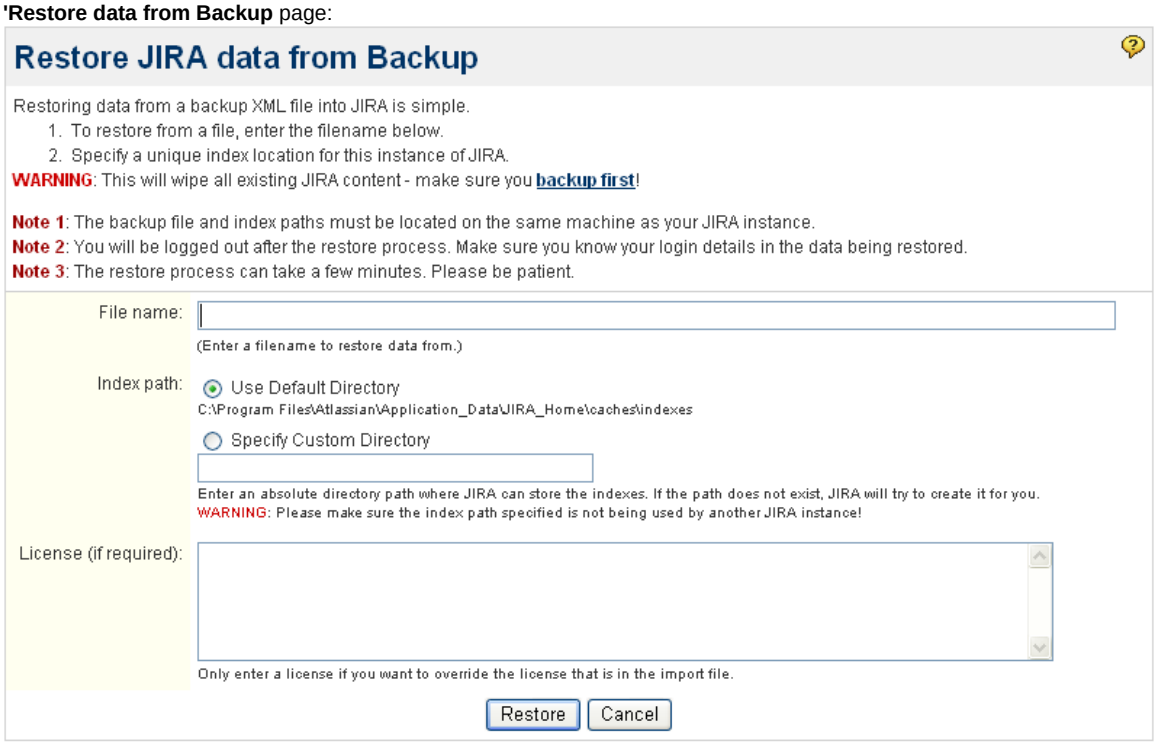
Exactly how to set these flags is dependent on your particular application server, but for JIRA Standalone (i.e. Tomcat), it is done by setting the JAVA_OPTS environment variable before starting JIRA:

```
set JAVA_OPTS="-Datlassian.mail.senddisabled=true -Datlassian.mail.fetchdisabled=true"
cd bin
startup.bat
```

2. Restoring XML data

Note: these instructions assume that you are restoring an XML backup. If you used native database tools to create your backup, the restore process will also be tool-specific so these instructions do not apply to you.

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Import & Export'** sub-menu in the left-hand navigation column, click the **'Restore Data from XML'** link. This will display the **'Restore data from Backup** page:



4. In the **'File name'** field, fill in the full path to the ZIP or XML backup file generated by JIRA.
5. Specify the directory that will hold the indexes for this JIRA instance, by either selecting '**Use Default Directory**' (i.e. `caches/indexes` sub-directory in the JIRA home directory) or specifying your own location in '**Specify Custom Directory**'. Ensure this directory contains *only* the indexes, as its contents may be deleted by the restore process.
6. Click the **'Restore'** button, and wait while your JIRA data is restored.
7. JIRA will come back, informing you that you have been logged out. This is done because all the users from the previous JIRA instance have been deleted and replaced with users from the JIRA export file.
8. Log in, and if necessary, correct the search index path. Note: if you are running more than one JIRA instance on the same server, ensure that each instance uses a different index path.

3. Restoring attachments

If you created a backup of the attachments directory, you will need to restore the backup into a directory where JIRA can access it. The process of restoring the attachments backup depends on the way it was created. Usually you can use the same tool to restore the backup as the one that was used to create it (see *Backing up attachments).* **Note: When you restore your attachments, ensure that the file permissions are correct.**

If you are restoring the attachments into a different location (i.e. a different directory path) from where they were previously located (e.g. this will be the case when moving servers), please follow the instructions provided in *Configuring attachments* to change the location of the attachments directory so that JIRA can find the restored attachments.

Restoring a Project from Backup

JIRA's Project Import tool allows you to restore a single project from a backup file into your JIRA instance. This feature is particularly useful if you do not wish to overwrite the existing projects or configuration of your JIRA instance by importing the entire backup. Your backup file must have been created using JIRA's backup tool. You cannot import a project from a backup using your native database tools.

Please note, if you wish to restore a project from a backup file into a **new empty JIRA instance,** we highly recommend that you **do not use the Project Import tool.** Restoring the entire backup file into the new instance and then deleting unwanted projects is much simpler in this scenario, as you will retain the configuration settings from your backup. Instructions on moving a project to a new instance are available on the splitting a JIRA instance page. Projects can be deleted via the 'Projects' page in JIRA, which is accessed from the '*Administration' menu.

On this page:

- Before you begin
  - Project Import Restrictions
    - JIRA versions do not match
    - JIRA editions do not match
    - Custom fields plugin versions do not match
- Restoring your project
  - Preparing your target JIRA instance
    - 1. Setting up the project
    - 2. Setting up users and groups
    - 3. Setting up custom fields
    - 4. Setting up workflows, system fields, groups and roles
    - 5. Setting up links
  - Project Import
    - 1. Specify the backup file
    - 2. Select a project to restore
    - 3. Review data mapping validations
    - 4. Verify the restored project
  - Need Help?

Before you begin

Restoring a project from a backup is not a trivial task. You may be required to change the configuration of your target JIRA instance to accommodate the project import. Additionally, the Project Import data mapping can be resource intensive on your hardware and may take a long time to complete, if you are importing a large project. Note, the Project Import tool will lock out your instance of JIRA during the actual data import (not during the validations), so please ensure that your instance does not need to be accessible during this time.

We strongly recommend that you perform a full backup of your target JIRA instance before attempting to restore a project into it.

Project Import Restrictions

The Project Import tool will only import a project between identical instances of JIRA. That is;

- The **version** of JIRA in which your backup was created must be identical to the version of your target JIRA instance, e.g. if your backup file was created in JIRA 4.0, then your target instance of JIRA must be version 4.0.
- The **edition** of JIRA in which your backup was created must be identical to the edition of your target JIRA instance, e.g. if your backup file was created in JIRA Standard Edition , then your target instance of JIRA must be JIRA Standard Edition.
- If your instance of JIRA had a **custom fields plugin** (e.g. JIRA Toolkit) installed when the backup file was created and the custom field was used in your project, then your target instance of JIRA must have the same version of the plugin installed for the Project Import tool to automatically work.

If any of these restrictions apply and you still wish to restore your project from backup, you will need to create a compatible backup file before importing your project by following the appropriate instructions below.

JIRA versions do not match

- If your backup file was created in an earlier version of JIRA than your target instance of JIRA:
    1. Set up a test JIRA instance, which is the same version as your target instance of JIRA. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance.
    2. Import the backup file into a test JIRA instance.
    3. Create a new backup file from your test JIRA instance.
- If your backup file is from a later version of JIRA than your target instance of JIRA:
    1. Upgrade the version of your target instance of JIRA to match the version of JIRA in which the backup was created.

JIRA editions do not match

- If your backup file is from a lower edition of JIRA than your target instance of JIRA:
    1. Set up a test JIRA instance, which is the same edition as your target instance of JIRA. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance.
    2. Import the backup file into a test JIRA instance.
    3. Create a new backup file from your test JIRA instance.
- If your backup file is from a higher edition of JIRA than your target instance of JIRA, you may wish to consider upgrading the edition of your target JIRA instance. Otherwise:
    1. Import the backup file into a test JIRA instance. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance, as the import will overwrite all data in the database.
    2. In your test JIRA instance, downgrade the edition of JIRA to match the edition of your target instance of JIRA. The downgrading JIRA page contains useful information that can help you with this process.
    3. Create a new backup file from your test JIRA instance.

Custom fields plugin versions do not match

- If the custom fields plugin from your backup is an earlier version than the custom fields plugin in your target instance of JIRA:
    1. Import the backup file into a test JIRA instance. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance, as the import will overwrite all data in the database.
    2. In your test JIRA instance, upgrade your version of your custom fields plugin to match the version of the plugin in your target instance of JIRA.
    3. Create a new backup file from your test JIRA instance.
- If the custom fields plugin from your backup is a later version than the custom fields plugin in your target instance of JIRA:
    1. Upgrade the custom fields plugin version of your target instance of JIRA to match the version of JIRA in which the backup was created.

Restoring your project

The Project Import tool will attempt to map the data in your backup file into your target JIRA instance. If the project you are restoring does not exist in your target JIRA instance, it will create and populate the project with data from your backup. If the project already exists and is empty, it will attempt to populate the data from your backup into the project.

> **Why should I create an empty project in my target JIRA instance?**
> It is important to note that the primary task of the Project Import tool is to restore the data from your backup project into your target JIRA instance. While the Project Import tool can create a project if one does not exist in your target JIRA instance, it does not recreate any configuration settings that affect the data (e.g. screen schemes). If you wish to retain any configuration settings from your original project, we recommend that you create an empty project in your target instance with the necessary configuration settings before importing the data from your backup project.

You may wish to carry out the following setup tasks to ensure that your target JIRA instance is prepared to receive a project import beforehand. This can improve the time taken to validate the data mappings to your target JIRA instance.

If you are confident that your JIRA instance is set up appropriately, you can skip straight to the Project Import tool instructions. If there are any problems mapping the data from your backup file to your target JIRA instance, the Project Import tool will present validation errors for you to address.

Preparing your target JIRA instance

The Project Import tool does not automatically add missing project entities (e.g. user groups, issue priorities, custom field types) or fix incorrect associations (e.g. issue types in workflow schemes), so some manual work is required to set up your target JIRA instance so that your project can be restored. If the Project Import wizard cannot find a valid target location for any of the backup project data, it will not be able to restore the project. The instructions below describe the setup activities that address the most common data mapping problems that occur when restoring a project from a backup.

We recommend that you perform as much of the configuration of your target JIRA instance as possible, prior to starting the project import. However, if you do not have the information available to complete these setup activities beforehand, the Project Import wizard will inform you of any problems that need your attention. Alternatively, you can import the backup file into a test JIRA instance to check the configuration.

1. Setting up the project

If you have a project in your target JIRA instance that you wish to restore data into, you will need to ensure that the project is empty, i.e.

- no issues — read the Quick Search page to find out how to find all issues in a project
- no components — read the Component Management page to find out how to view a summary of a project's components
- no versions — read the Version Management page to find out how to view a summary of a project's versions

2. Setting up users and groups

The following types of users are considered mandatory for a project to be imported:

- reporter, assignee, component lead or project lead.

The following users are considered to be optional for a project to be imported:

- comment author/editor, work log author/editor, a user in a custom field (user picker), voter, watcher, change group author (i.e. someone who has changed an issue), attachment author, user in a project role.

The Project Import will attempt to create missing users if they are associated with the project. However, if the Project Import tool cannot create missing mandatory users in your target JIRA instance, then you will not be permitted to import the project. This may occur in the following situations:

- you have External User Management enabled in your target JIRA instance — you will need to disable External User Management or create the missing users manually in your external user repository before commencing the import.
- Atlassian's Crowd was connected to your JIRA instance when the backup was made (hence, the backup file will only contain minimal reference data for each user which is insufficient to create users in JIRA) — you will need to create any missing mandatory users manually. Alternatively, you may wish to connect the current JIRA to the same external user management system as the original, if that is possible. This check does not apply if you connected an LDAP (not using Crowd) to your JIRA instance when the backup was made, as connecting an LDAP to JIRA requires the creation of the users in JIRA (hence, full user data will be in the backup file).

Please note that if you do not have enough information about the users in your backup file, the Project Import wizard will provide a link to a table of the missing users on a new page as well as a link to an XML file containing the missing users (on the new page). The table of users will display a maximum of 100 users, but the XML file will always be available.

3. Setting up custom fields

As described previously, the versions of your custom field plugins must match between your backup and your target instance of JIRA for your project to be imported. You need to ensure that you have set up your custom fields correctly in your target JIRA instance, as follows:

- **Custom Field Type** — If you do not have a particular custom field type (e.g. cascading select) installed on your target JIRA, then all custom field data in your backup project that uses that custom field type will not be restored. However, your project can still be restored. For example, say you have a custom field, 'Title', which is a 'Cascading Select' field type and was used in your backup project (i.e. there is saved data for this field). If you do not have the 'Cascading Select' custom field type installed on your target JIRA, then all data for custom field 'Title' (and all other cascading select custom fields) will not be restored.
- **Custom Field Configuration** — If you do have a particular custom field type (e.g. multi select) installed on your target JIRA, then you must configure all of the custom fields (of that custom type) in your target JIRA to match the equivalent custom fields in your backup project. Additionally, if your custom field has selectable options, then any options used (i.e. there is saved data for these options) in your backup project must exist as options for the custom field in your target JIRA.
  For example, say you have a custom multi select field named, 'Preferred Contact Method', in your backup project with options, 'Phone', 'Email', 'Fax'. Only the 'Phone' and 'Email' were actually used in your backup project. In this scenario, you need to set up your target JIRA instance as follows:
  - There must be a field named, 'Preferred Contact Method', in your target JIRA instance.
  - 'Preferred Contact Method' must be a multi select custom field type.
  - 'Preferred Contact Method' must have the options, 'Phone' and 'Email' at a minimum, since they were used in your backup project. Please note, 'Preferred Contact Method' in your target JIRA could also have additional options like 'Fax', 'Post', 'Mobile', etc, if you choose.
    If you have not configured your existing custom field correctly, you will not be permitted to import your backup project until you correct the configuration errors in your target JIRA.
    You may wish to refer to the custom fields documentation for more information on the custom field types and custom field configuration.
- **Compatibility with the Project Import tool** — Custom fields also need to be compatible with the Project Import tool for the custom field data to be imported. Custom fields created prior to JIRA v4.0 cannot be imported by the Project Import tool. The custom field developer will need to make additional code changes to allow the Project Import tool to restore the custom field data. If any of the custom fields used in your backup file are not compatible with the Project Import tool, the Project Import wizard will warn you and the related custom field data will not be imported. All the target JIRA system custom fields and the custom fields included in JIRA plugins supported by Atlassian (e.g. JIRA Toolkit, Charting Plugin, Labels Plugin, Perforce Plugin) are compatible with the Project Import tool.

4. Setting up workflows, system fields, groups and roles

In addition to custom fields, you need to correctly configure the project workflow, issue attributes (e.g. issue types) and groups/roles in your target JIRA instance for your project to be restored successfully. Please ensure that you have reviewed the constraints on each of the following:

Workflows and Workflow Schemes *(Professional and Enterprise editions only):*

- The project import process does not import workflows or workflow schemes. If you wish to retain a customised workflow from your backup, you will need to create a new workflow in your target JIRA instance and manually edit the new workflow (e.g. create steps and transitions) to reflect your old workflow (note, the default JIRA workflow is not editable). You will then have to add this workflow to a workflow scheme to activate it.
  Read more about creating and editing workflows in the JIRA Workflow and Activating Workflows documents. Please note that you may be required to create and edit a new workflow and workflow scheme to satisfy constraints on workflow entities from your backup, as described in the sections below, even if you do not wish to recreate the exact same workflow.

> **Do not** use the JIRA functionality for exporting and importing workflow XML definitions, to copy your backup workflow to your target JIRA instance. The workflow import/export tools do not include workflow screens in the process. Hence, you will be required to manually edit the workflow definitions post-import to match up new screens to the workflow, which is more work that it is worth.

Issue Types:

- If an issue type has been used in your backup project (i.e. there are issues of this issue type), you must set up the same issue type in your target JIRA project. If you are using JIRA Professional or JIRA Enterprise, you may wish to set up Issue Types for the project instead of globally.
- Workflow schemes — If you have associated an issue type with a particular workflow scheme in your backup project, you must ensure that the same association exists in your target JIRA. See the above section on **'Workflow and Workflow Schemes'** for further information on how to set up a workflow in your target JIRA instance.
- Custom field configuration schemes — custom field configuration schemes can be used to apply a custom field configuration to specific issue types. If you have configured a custom field differently for different issue types in your backup project, you may wish to set up a custom field configuration scheme to apply the same custom field configuration to the same issue types in your target JIRA instance. This will help ensure that you do not have a custom field for an issue type that is configured incorrectly (e.g. missing an option, if it has multiple selectable options), as described in the 'Setting up custom fields' section above.

Statuses:

- If an issue status has been used in your backup project (i.e. there are issues with the status), you must set up the same status in your target JIRA project.
- Workflow schemes — If you have linked a status into a particular workflow scheme in your backup project, you must ensure that the same association exists in your target JIRA. See the above section on 'Workflow and Workflow Schemes' for further information on how to set up a workflow in your target JIRA instance.

Security Levels (Enterprise Edition only):

- If an issue security level has been used in your backup project (i.e. there are issues with this security level), it must be set up in your target instance of JIRA. If you did not create an existing empty project, we recommend that you do so and set up the appropriate security levels for the project (via an issue security scheme).
- Issue Security schemes — Not applicable. It does not matter which users, groups or project roles are assigned to which security levels, as long as the appropriate security levels exist (please see the constraints on security levels in the 'Setting up entities and types' section).

Priority:

- If an issue priority has been used in your backup project (i.e. there are issues with this priority), it must be set up in your target instance of JIRA.

Resolution:

- If an issue resolution has been used in your backup project (i.e. there are issues with this resolution), it must be set up in your target instance of JIRA.

Issue Link Type:

- If an issue link type has been used in your backup project (i.e. there are issues associated by this link type), it must be set up in your target instance of JIRA.

Project Role:

- If a project role has been used in your backup project (i.e. there are users/groups assigned to this project role), it must be set up in your target instance of JIRA.
  (Note: The Project Import tool will copy across the project role membership from your backup project to your target JIRA instance, if you choose. See the Project Import section for further details).

Group:

- If a user group has been used in your backup project (i.e. there are users in this group), it must be set up in your target instance of JIRA.

A note about schemes
The project import process does not directly affect schemes, although entities and types associated with schemes may be affected as described above. Please note that the following schemes are not affected at all by the project import:

- Permission schemes — Not applicable. Permissions schemes do not need to match between the backup and target instance of JIRA.
- Notification schemes — Not applicable. Notification schemes do not need to match between the backup and target instance of JIRA.
- Screen schemes — Not applicable. Screen schemes do not need to match between the backup and target instance of JIRA.
- Issue type screen schemes (Enterprise Edition only) — Not applicable. Issue type screen schemes do not need to match between the backup and target instance of JIRA.
- Field Configuration schemes (Enterprise Edition only) — Not applicable. Please note that if a field was configured as optional in your backup project and is configured as a required field in your target JIRA instance, then the project will still be imported even if the field is empty. However, this field will be enforced as mandatory the next time a user edits an issue containing the field.

5. Setting up links

The Project Import tool will automatically create all issue links between issues within your backed up project. It will also try to create links between the backup project and another project, as long as the other project already exists in your target JIRA instance with the relevant issue keys. If the source/target of a link cannot be found (i.e. the entire project or the particular issue may be missing), the link will not be created although the project will still be imported.

Note that the Project Import tool will create issue links between projects in either direction (source to target, or target to source). This means that if you import two projects from the same backup file, the second project import will create all of the links between the two projects that were missing from the first project import.

Once you have completed as many of the setup tasks as you are able to, run the Project Import tool.

Project Import

Restoring your project is a four step process:

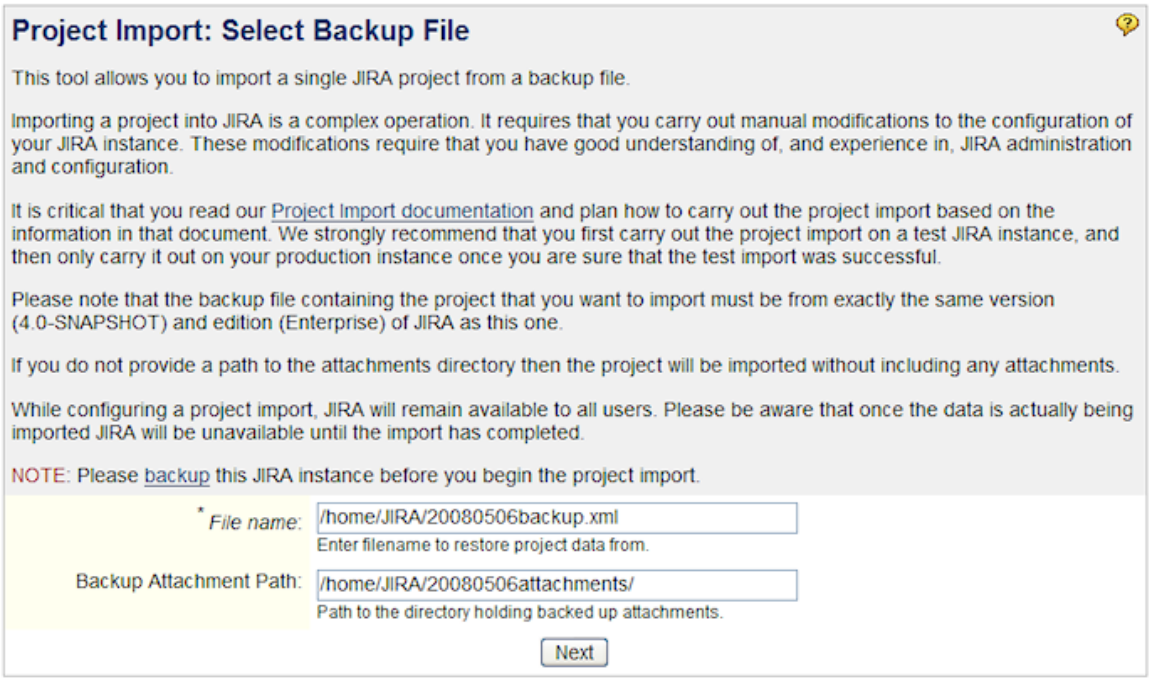
1. Specify the backup file
2. Select a project
3. Review data mapping validations
4. Verify the restored project

If you start the Project Import tool, we strongly recommend that you complete all steps of the wizard before performing any other activities in JIRA. Please be aware that it can take some time to validate the data mappings and then import the project.

You will most likely need to navigate away from the Project Import wizard to correct your JIRA configuration, as advised by validation errors in the wizard. If you have to navigate to other pages in JIRA to correct your JIRA configuration or for other activities, you should:

- ***(recommended)*** open a separate session of JIRA in a new browser window/tab. When you return to the Project Import wizard in the original browser window/tab, you can use the **'Refresh validations'** button on the validation screen to re-validate the data mappings; or,
- wait until the progress bar completes for the step you are currently in, before navigating elsewhere in JIRA. The state of the Project Import wizard will be saved until you log out of JIRA, your user session expires or you commence a different project import. You can resume your project import by returning to the Project Import page (via the main Administration menu) and selecting the 'resume' link on the first page of the wizard.

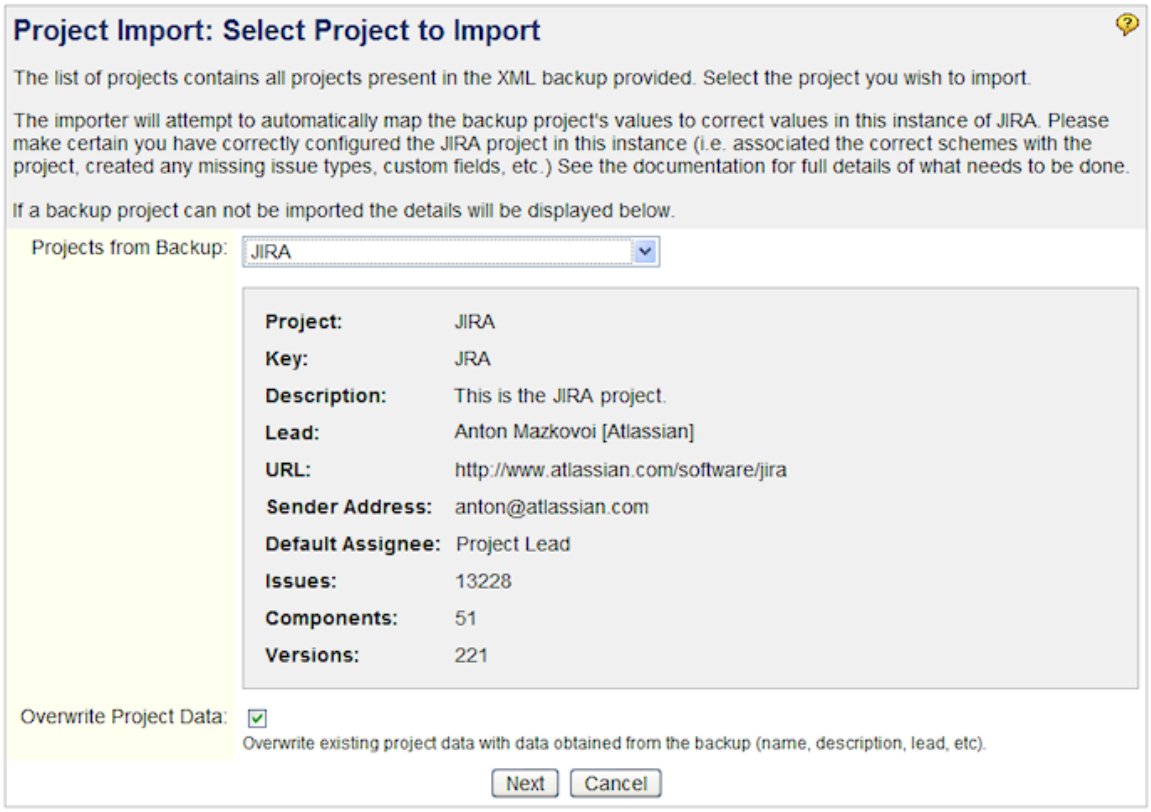
1. Specify the backup file



To start the Project Import tool,

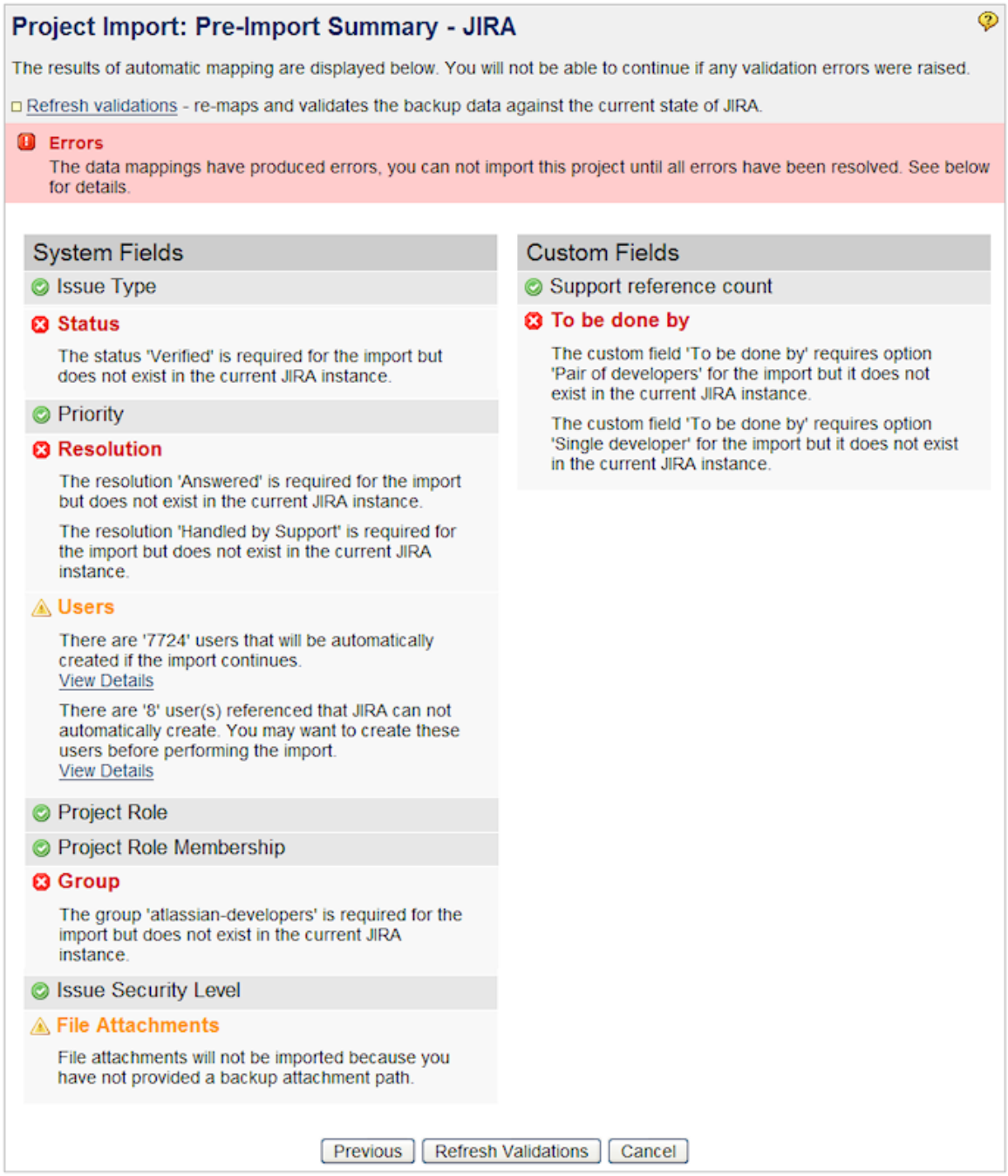
1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Click the **'Project Import'** link in the left hand menu. The first step of the Project Import wizard will display, ' Project Import: Select Backup File'.
4. Specify the path and name of your backup file in the 'File name' field. Your backup file must be an XML or ZIP file (as exported by JIRA).
5. Specify the path where you have backed up the attachments (add anchor to backup attachments section) for your project in the **'Backup Attachment Path'** field. Do not specify the attachment path for your target instance of JIRA as the backup attachment path, as the Project Import tool will overwrite attachments in that directory. Please also ensure that you have enabled file attachments in your target JIRA instance. You will not be allowed to proceed with the import if you have specified a backup attachment path and do not enable file attachments in your target JIRA instance.
   Note: You can choose to not specify a backup attachment path. If so, you will be able to restore your project from backup, however it will have no attachments associated with it. Please note, you cannot restore your attachments separately if you do not restore them as part of the project import, as the database entries for the attachments will be missing.

2. Select a project to restore



1. Select a project to restore from the **'Projects from Backup'** dropdown. This dropdown will list all of the projects contained in your backup file.
2. If you have a valid project to restore from your backup, and your target JIRA instance has an existing empty project, then the **'Overwrite Project Details'** option will display. Select the **'Overwrite Project Details'** option if you want to overwrite the project details of the existing empty project with the project details from your backup. The project details are the Name, URL, Project Lead, Default Assignee and Description of the project, as well as any project role members set up on your project. If there is no existing empty project in your target instance of JIRA, this option will be checked and disabled as the Project Import will create the project with project details from your backup file.

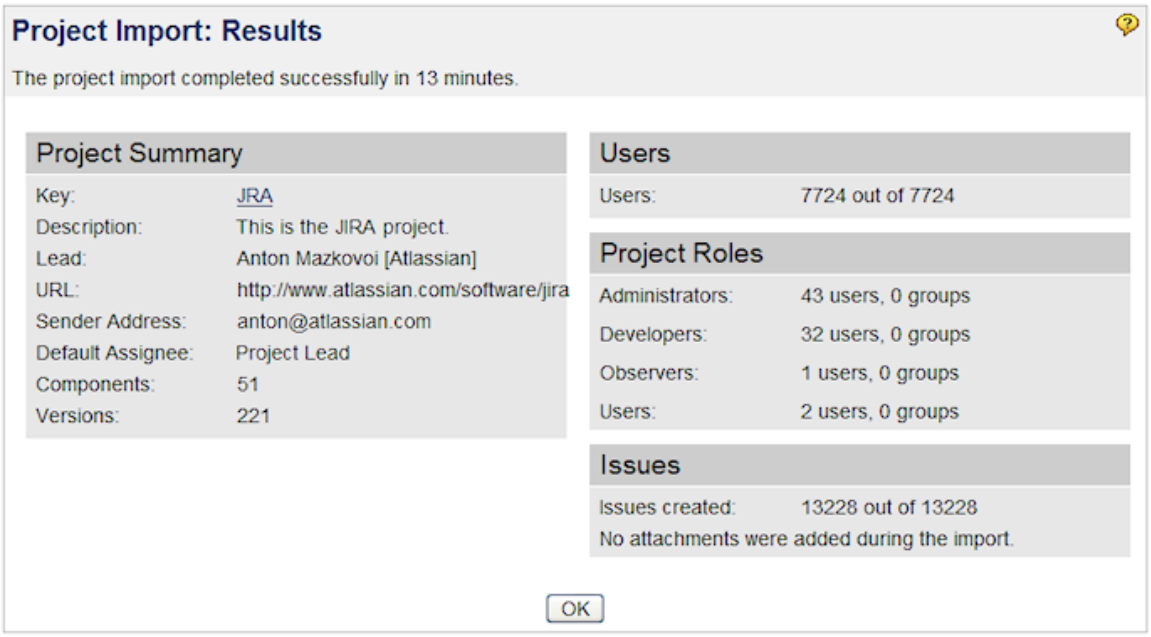
3. Review data mapping validations



1. The Project Import wizard will attempt to validate the data mappings required to import your project from the backup file. You can review the validations at this step of the wizard and modify your target JIRA instance as required.
    - A tick symbol ( ) means that there are no problems with mapping these entities.
    - An exclamation mark symbol ( ) means that there are problems with the data mapping that you should review before importing the project, but the project can still be imported. For example, a missing optional user that cannot be created automatically by the Project Import tool.
    - A cross symbol ( ) means that there are problems with the data mapping that must be fixed before you can import the project. For example, an Issue Type that is used in the backed up project is missing in your target JIRA instance.
2. The 'Preparing your target JIRA instance' section on this page lists the common data mapping errors.
3. Once you have resolved the data validation errors as required, click **'Import'** to commence the import of data from your backup file.

The Project Import tool will lock out your instance of JIRA during the actual data import (not during the validations), so please ensure that your instance does not need to be accessible during this time.

4. Verify the restored project



1. Once the Project Tool has finished running, click **'OK'** to navigate to the restored project. You should verify that the issues, components and versions have been restored correctly. You should also check that any custom field data and links have been restored correctly.
2. Check that your attachments were correctly restored from your attachments backup directory.

> The Project Import tool will add an entry to every imported issue's Change History, showing when the issue was imported. Note that old entries in the Change History, from before the import, are retained for historical purposes only. Old entries may contain inconsistent data, since the configuration of the old and new JIRA systems may be different.

What if something went wrong?

- If your project import **did not complete,** you can refer to the JIRA log file. The Project Import tool will log details of the operation to this file, including any unexpected errors and exceptions. e.g. database locked out, disk full... etc.
- If your project import completed but **did not restore your project as expected,** you may wish to attempt to fix the problem manually in your target JIRA instance. You may also wish to try deleting the project in your target JIRA instance and re-importing it from backup, paying special note to any warning validations (e.g. users that will not be added automatically).

If you cannot resolve the problem yourself, you can contact us for assistance. Please see the **'Need help'** section below for details.

Need Help?

Need further help? You can raise a support request in the JIRA project at https://support.atlassian.com for assistance from our support team. Please attach to the support case:

- the backup file you are trying to import projects from, and
- the following information from your target JIRA instance:
  - your log file
  - an XML backup of your target JIRA instance
  - a copy and paste of the **entire contents** of the **System Info** page (accessed via the **Administration** tab), so that we know the details of your JIRA configuration.

You can anonymise the XML backups, if your data contains sensitive information.

Using the Database Integrity Checker

Searching for common data inconsistencies, the **Database Integrity Checker** attempts to ensure that all JIRA data is in a consistent state.

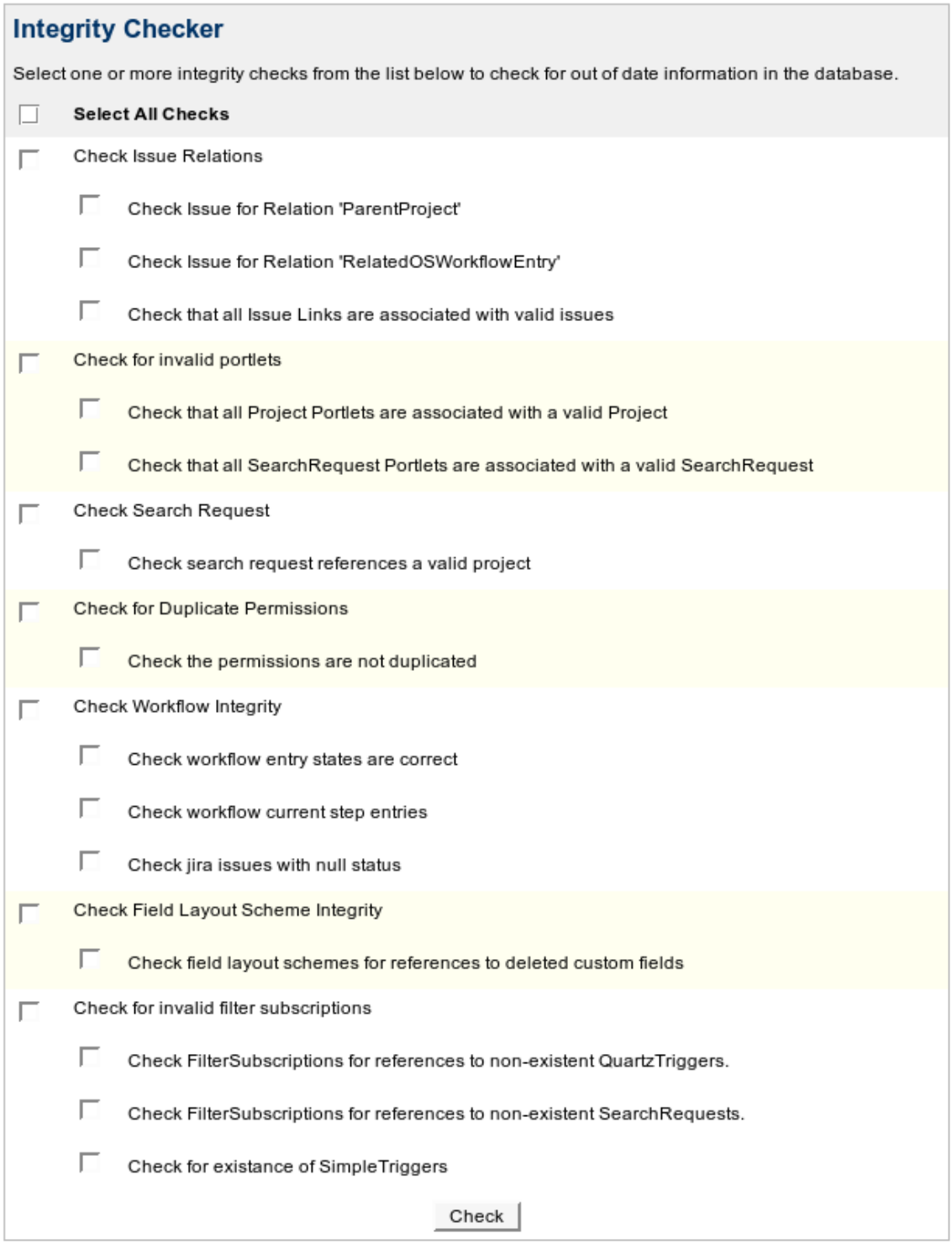
This is useful in a number of situations, e.g.

- Before migrating a project to a new workflow
- An external program is modifying JIRA's database
- Troubleshooting a server crash

If an error is encountered, most of the integrity checks provide a 'repair' option which attempts to reset the data to a stable state.

Using the Integrity Checker

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu in the left-hand navigation solumne, click the **'Integrity Checker'** link.
4. The **'Integrity Checker'** screen will be displayed:



5. The integrity checker has a number of 'integrity checks' that look for common inconsistencies in the data. Select one or more checks you would like to run, then click the **'Check'** button.
6. After the selected checks run, the preview screen will be shown.

The screen provides details about the existing data inconsistencies. If any inconsistencies were found, the **'Fix'** button will also appear on the page. The messages in red describe inconsistencies that the check will correct if it is chosen and the **'Fix'** button is clicked. Messages that appear in yellow are warnings that the check will not correct; JIRA will auto-recover from these inconsistencies when an action is taken on an issue.
Select any inconsistencies that you would like to correct, then click the **'Fix'** button.

We **strongly** recommend taking a backup of your data before correcting any data inconsistencies.

7. If any inconsistencies were found and you chose to correct them, you will be presented with a summary screen describing all the corrective actions that have taken place.

Optimising Performance

While implementing some of the following steps to help improve the performance of your JIRA install, it will also help to gather some data on just where your performance bottlenecks might be. See Is your JIRA Running Slowly for a step-by-step guide on gathering the type of information you will need to help diagnose and resolve performance problems.

On this page:

- Profiling
- Environment options
  - Virus checking
  - Network shares
  - SSL or HTTPS
- Database options
  - JDBC drivers
  - Databases
  - Network latency
  - Index your database
- JDK options
  - Choose the latest JDK version
  - Use the Server JVM
  - Allocate enough Memory
- JIRA options
  - Enable GZip Compression
  - Remove HSQLDB parameters in JIRA Standalone
  - External User Management
- Application Server options
  - Database Connection pooling.
- Server specifications

Profiling

To quantify performance problems, you can turn JIRA profiling on. You will then get a profile for each URL requested in the logs:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

One performance problem with JIRA can be the application server that you are running. The database connection pooling, JSP compilation times and resource allocation are different between application servers. Known slow servers include JBoss 3.x, Tomcat 4.0 and Tomcat 4.1.24. The fastest servers for JIRA at the moment are more recent versions of Tomcat, as well as Resin and Orion.

Databases can also have a large impact on performance, particularly if the database is accessed across a network, or has not been indexed properly.

If you can't change your application server, there are performance improvements available, both by tuning your server, database and through setting certain JIRA options.

Environment options

Virus checking

If you are experiencing slowness with JIRA, try running JIRA with virus checking disabled. As JIRA creates many temporary files, **virus checking software can slow JIRA dramatically.** McAfee's NetShield 4.5 in particular claims to let you exclude folders from scanning, but doesn't actually - upgrade to 7.0.0 to fix this. Symantec must be **uninstalled** - painful experience has proven that even stopping the services does not prevent it slowing JIRA down.

Network shares

JIRA needs fast access to the local filesystem. If you are hosting JIRA, or its index directory, on a network share (SMB, NFS etc), this can cause a large loss in performance. Run JIRA with fast local disk access.

SSL or HTTPS

Although some organisations have a requirement to run JIRA over SSL or HTTPS, please note that this this can affect performance.

Database options

JDBC drivers

Different JDBC drivers have different performance characteristics. Ensure that you are using the latest patched version of the JDBC drivers for your database.

Databases

JIRA Standalone (and many application servers) ship with an in-memory database like hsqldb. Using another database (e.g. MySQL, PostgreSQL or Oracle) will usually result in higher performance.

Network latency

The latency between the database server and the server hosting JIRA can be a source of performance problems. If the database is hosted on a different machine to JIRA, please check the ping times between the servers.

Index your database

JIRA 3.0+ automatically creates database indexes when the database is first created. However, if you have been doing in-place JIRA upgrades from earlier versions (not dropping/recreating the database tables), your database will not be indexed. Doing a full XML backup and restoring into an empty database will fix this. Additional indexes may be created by hand, but this is usually not necessary.

JDK options

Choose the latest JDK version

The latest JDKs contain performance optimisations that will improve the performance of JIRA. JIRA uses a lot of reflection, which was greatly improved in the 1.4 release.

Use the Server JVM

Sun ships two versions of the JDK, a client version and a server version. They have different characteristics such as memory management and inline optimisations. You may need to explicitly start your application server like "java -server -jar <app-server-jar>.jar". With JDK 1.5 it is best to leave this unset.

Allocate enough Memory

By default, many application servers are not started with enough memory for JIRA to run at an optimum speed. A lack of memory increases garbage collection time, as garbage collection has to be run more frequently.

To see if lack of memory is causing slowness, please add these parameters to JIRA's startup command:

```
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -Xloggc:${LOG}/gc.log
```

Where `${LOG}/` is a filesystem path to your log directory. Garbage collection times will then be logged in `gc.log`.

You may need to start your application server like "java -server -Xms100m -Xmx300m <app-server-jar>.jar". See Increasing JIRA Memory for more details.

JIRA options

Enable GZip Compression

JIRA can compress page contents between the server and your browser, resulting in improved performance especially over slow connections. Check that GZip compression is enabled in Administration -> Global Settings -> General Configuration (unless you are using `mod_proxy` .

Remove HSQLDB parameters in JIRA Standalone

If you are using JIRA Standalone *modified to use an external database,* make sure you **delete** the highlighted section in your `conf/server.xml` , which otherwise results in poor performance:

```
      <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
        username="jirauser"
        password="jirapassword"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost/jiradb?autoReconnect=true"
minEvictableIdleTimeMillis="4000"
        timeBetweenEvictionRunsMillis="5000"
        />
```

External User Management

If External User Management is turned on (**not** the default), JIRA will not cache users & groups, potentially resulting in slow access.

Application Server options

Database Connection pooling.

Obtaining a connection to a database is an expensive operation, and most application servers maintain a pool of open connections to reduce this overhead. It is worth checking that you have a sensible number of connections pooled, sensible expiry times etc. This is configured in your app server.

If you are using the standalone version of JIRA or Apache Tomcat you can modify the DBCP connection pool in Tomcat's server.xml (or possibly jira.xml, depending on how you have setup JIRA). The values you will be most interested in modifying are the following:

```
              <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
              username="sa"
              password=""
              driverClassName="org.hsqldb.jdbcDriver"
              url="jdbc:hsqldb:${catalina.home}/database/jiradb"
              minEvictableIdleTimeMillis="4000"
              timeBetweenEvictionRunsMillis="5000"
              maxActive="20"
                  minIdle="4"
                  maxIdle="8"
              />
```

For information on what these values mean please view the Apache DBCP documentation

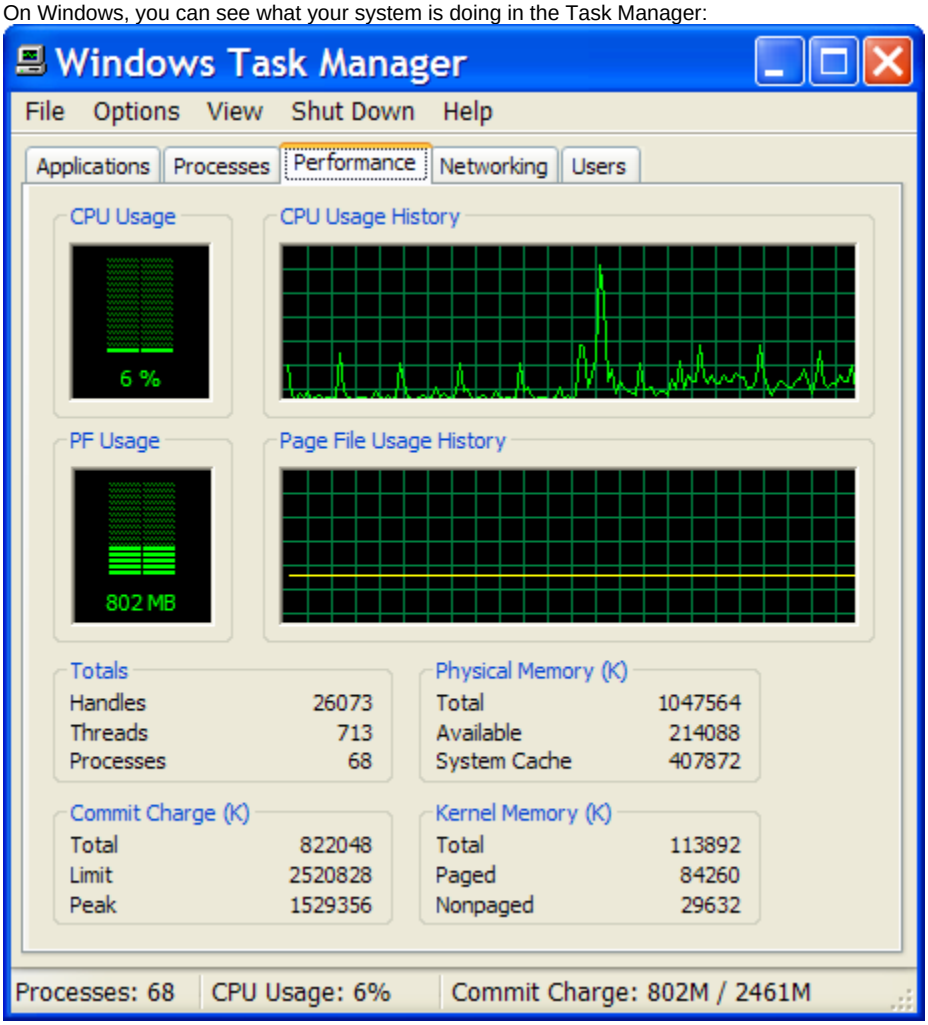
Other application server tuning may be of benefit. Consult your application server documentation for more information.

Server specifications

JIRA performance is very dependent on CPU and available memory. Lack of physical memory, or overly high maximum heap size setting (the

-Xmx flag) can seriously degrade JIRA performance, as memory accesses result in constant swapping of data between memory and disk (virtual memory).

On Windows, you can see what your system is doing in the Task Manager:



On Linux/Solaris, `vmstat 1` will print virtual memory and CPU statistics:

```
$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 9  0 520512  27132  15216 318944    3    2    65    40    0     3 10  3 85  2
12  0 520512  27004  15216 319080    0    0   104     0 2041 10992 88  8  3  0
20  0 520512  26764  15228 319068    0    0     0   436 2196 12869 85 13  2  0
11  0 520512  26700  15228 319068    0    0     0     0 1959  4041 88  9  4  0
20  0 520512  25724  15228 319068    0    0     0     4 2137  3307 84 14  2  0
17  0 520512  25724  15228 319068    0    0     4     0 2017 10488 89  8  3  0
 9  0 520512  25468  15228 319068    0    0     0     0 1886  7532 86 11  3  0
...
```

(this is a very busy server with ~97% CPU usage, but fortunately no swapping)

This system info can be captured over a long time with `vmstat -n 1 > vmstat.log`

On Linux, CPU and memory info can be obtained with `cat /proc/cpuinfo` and `cat /proc/meminfo` respectively.

If you need to raise a support request, please include this info (`vmstat 1`, `/proc/cpuinfo`, `/proc/meminfo`).

Precompiling JSP pages

If you decided to go the extra mile and extend JIRA's build process to precompile JSP pages, keep in mind that the "include" directory in the JIRA web application needs to be excluded from precompilation. The reason for this is that the JSP files in the "include" directory are not proper JSP files, but are includes that are only meant to be compiled as part of larger JSP pages.

For example, to exclude the JSP pages in the "include" directory when using Maven use the <exclude> sub-element of the <ant:jspc> task, as shown:

```
<ant:path id="jspc.classpath">
  <ant:pathelement location="${tomcat.home}/common/lib/jasper-runtime.jar"/>
  <ant:pathelement location="${tomcat.home}/common/lib/jasper-compiler.jar"/>
  <ant:pathelement location="${tomcat.home}/common/lib/servlet.jar"/>
  <ant:path refid="maven-classpath"/>
  <ant:path refid="maven.dependency.classpath"/>
  <ant:pathelement path="${maven.build.dest}"/>
  <ant:pathelement path="${java.home}/lib/tools.jar"/>
</ant:path>
<ant:jspc
  package="${pom.package}.jsp"
  destDir="${jspOutDir}"
  srcdir="${warSource}"
  uriroot="${warSource}"
  uribase="/${pom.artifactId}"
  verbose="2"
  classpathref="jspc.classpath">
  <ant:include name="**/*.jsp"/>
  <ant:exclude name="**/includes/**/*.jsp"/>
</ant:jspc>
```

Logging and Profiling

On this page:

- Logging
  - To temporarily change the logging level:
  - To permanently change the logging level:
- Profiling
  - To temporarily enable profiling:
  - To permanently enable profiling:

Logging

JIRA uses a powerful logging module called log4j.

The default logging levels can be changed either;

- temporarily — your change to the logging level will not persist after you next restart JIRA, or
- permanently — your change to the logging level will persist, even after you restart JIRA.

For example, when troubleshooting, you might temporarily change the logging level from 'WARNING' to 'INFO' so as to get a more detailed error message or a stack trace.

To temporarily change the logging level:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu in the left-hand navigation column, click the **'Logging & Profiling'** link.
4. The **'Logging & Profiling'** page will display. This lists all the defined log4j categories and their current logging levels. To edit the logging level of a category, click the **'Edit'** link next to the category in the list.
5. Choose the new logging level for the category, then click **'Update'.**

To permanently change the logging level:

1. Edit the `log4j.properties` file, which is found in the `WEB-INF/classes/` directory under the JIRA web application directory.

   The `log4j.properties` file that ships with JIRA has the default logging levels specified. For more information about log4j (e.g. how to define new logging categories), and about the format of the `log4j.properties` file, please refer to the documentation on the log4j site.

2. Restart JIRA.

If your application server itself configures logging (e.g. JBoss), you may need to remove the `log4j.properties` file. On some servers (e.g. JBoss 3.0), you may also need to remove the entire `log4j.jar` file to get logging to work.

Profiling

If you are experiencing performance issues with JIRA, it is often helpful to see where the slow-downs occur. To do this you can enable profiling as described below, and then analyse the performance traces that JIRA will produce for every request. An example of a profiling trace is shown below:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

Profiling can be enabled either;

- temporarily — profiling will be enabled until you next restart JIRA, or
- permanently — profiling will remain enabled, even after you restart JIRA.

To temporarily enable profiling:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu in the left-hand navigation column, click the **'Logging & Profiling'** link.
4. The **'Logging & Profiling'** page will display. Scroll to the bottom of the screen. The **'Profiling'** section will inform you whether profiling is currently turned **'ON'** or **'OFF',** and will also show or hide the **'Disable profiling'** and **'Enable profiling'** links respectively.
   - To turn Profiling **'ON',** click the **'Enable profiling'** link. JIRA will start generating profiling traces in its log.
   - To turn Profiling **'OFF',** click the **'Disable profiling'** link.

To permanently enable profiling:

1. Edit `atlassian-jira/WEB-INF/web.xml` (if you are using JIRA Standalone) or `webapp/WEB-INF/web.xml` in your JIRA installation directory (if you are using JIRA EAR/WAR).
2. Find the following entry:

```
<filter>
            <filter-name>profiling</filter-name>
            <filter-class>com.atlassian.jira.web.filters.JIRAProfilingFilter</filter-class>
            <init-param>
                <!-- specify the which HTTP parameter to use to turn the filter on or off -->
                <!-- if not specified - defaults to "profile.filter" -->
                <param-name>activate.param</param-name>
                <param-value>jira_profile</param-value>
            </init-param>
            <init-param>
                <!-- specify the whether to start the filter automatically -->
                <!-- if not specified - defaults to "true" -->
                <param-name>autostart</param-name>
                <param-value>false</param-value>
            </init-param>
        </filter>
```

3. Modify the `autostart` parameter to be **true** instead of **false.** That is:

```
<init-param>
                <!-- specify the whether to start the filter automatically -->
                <!-- if not specified - defaults to "true" -->
                <param-name>autostart</param-name>
                <param-value>true</param-value>
            </init-param>
```

4. Save the file. If you are running JIRA Standalone this is all you have to do. Profiling will be enabled when you restart JIRA.
5. If you are running JIRA EAR/WAR, re-build and re-deploy the JIRA web application using the `build` script and the instructions for your application server.

Increasing JIRA Memory

Java applications like JIRA and Confluence run in a "Java virtual machine" (JVM), instead of directly within an operating system. When started, the Java virtual machine is allocated a certain amount of memory, which it makes available to apps like JIRA. By default, Java virtual machines are allocated 64Mb of memory, no matter how many gigabytes of memory your server may actually have available. 64Mb is inadequate for medium to large JIRA installations, and so this needs to be increased. Seeing OutOfMemoryErrors in the logs is symptomatic of this.

On this page:

- Checking available memory
- Increasing available memory
    - How much memory do you have?
    - Tomcat Windows Service
    - Tomcat on Windows (started manually)
    - Tomcat on Unix
    - Other application servers
- Permanent Generation Memory
    - Tomcat on Windows (started manually)
    - Tomcat on Unix

If you are seeing OutOfMemoryErrors, it is a good idea to also review the Causes of OutOfMemoryErrors page.

Checking available memory

How much memory is JIRA currently using? This can be determined by a JIRA administrator. In JIRA, go to Administration » System » System Info, and look at the memory graph:

System Info	
System Date	Wednesday, 30 Apr 2008
System Time	01:09:55
Current Working Directory	/opt/j2ee/domains/atlassian.com/extranet/jira/apache-tomcat-5.5.23
Java Version	1.6.0_02
Java Vendor	Sun Microsystems Inc.
JVM Version	1.0
JVM Vendor	Sun Microsystems Inc.
JVM Implementation Version	1.6.0_02-b05
Java Runtime	Java(TM) SE Runtime Environment
Java VM	Java HotSpot(TM) Server VM
User Name	j2ee
User Timezone	America/Chicago
User Locale	English (United States)
System Encoding	ANSI_X3.4-1968
Operating System	Linux 2.6.9-67.0.4.ELsmp
OS Architecture	i386
Application Server Container	Apache Tomcat/5.5.23
Database type	postgres72
Database JNDI address	java:comp/env/jdbc/JiraDS
Database URL	jdbc:postgresql://atlassian-db02:5432/atl-extranet-jira
Database version	8.2.6
Database driver	PostgreSQL Native Driver PostgreSQL 8.1 JDBC3 with SSL (build 408)
External user management	ON
Crowd integration	ON

Java VM Memory Statistics	
Total Memory	259 MB
Free Memory	72 MB
Used Memory	187 MB
Memory Graph	27 % Free (Force garbage collection)

This server has been allocated a maximum of 650Mb and a minimum of 256m. You can see the minimum displayed here; if you're trying to see whether your settings are being picked up, this is where to look. Of this, JIRA has reserved 543Mb, or which 310Mb is actually in use (this JIRA instance has about 6000 issues).

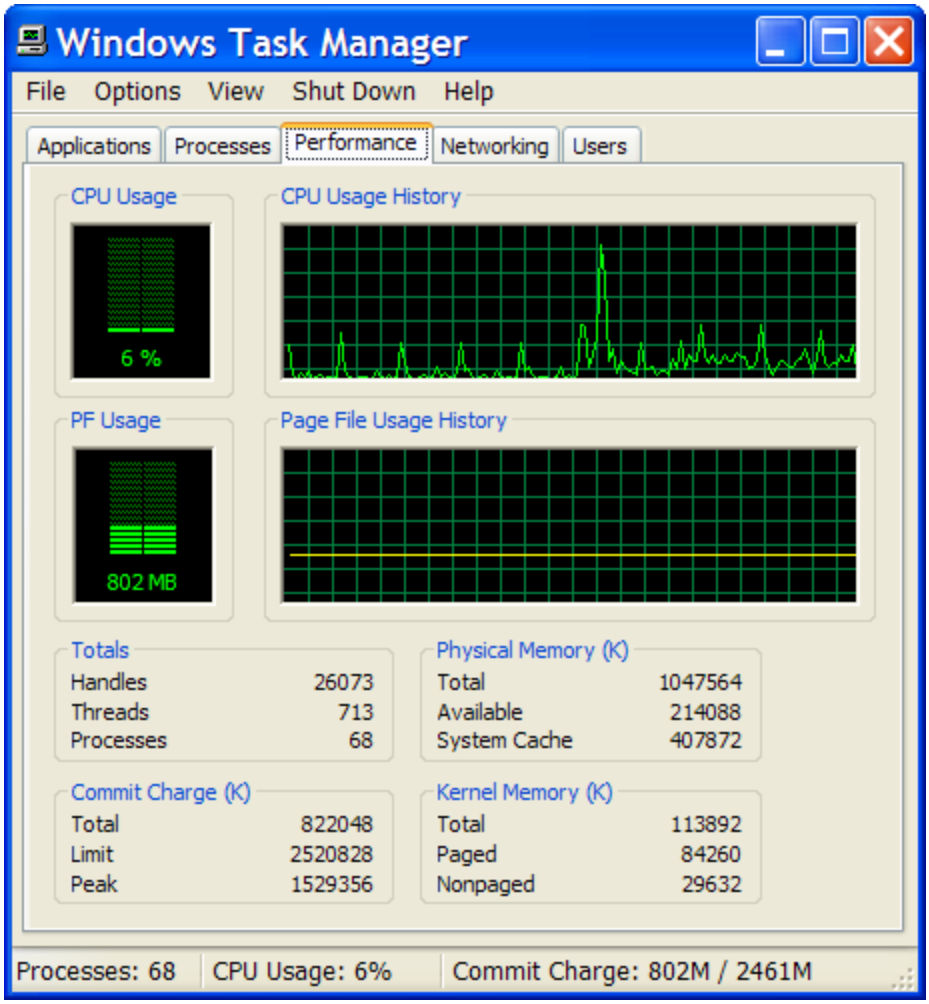
If this JIRA instance were running out of memory, it would have reserved the maximum available (650Mb), and would be using an amount close to this.

Increasing available memory

How much memory do you need? As a rule of thumb, if you have less than 5000 issues, JIRA should run fine in 256Mb. Granting JIRA too much memory will hurt performance, so it's best to start with 256Mb and make modest increases as necessary. As another data point, 40,000 issues fits comfortably into 750Mb.

How much memory do you have?

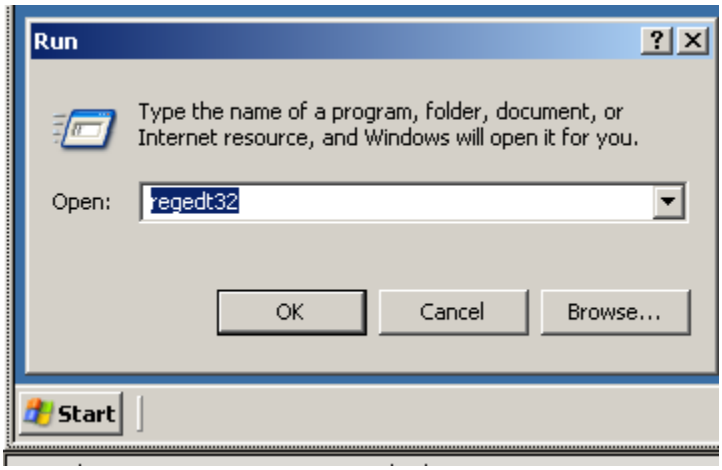
On Windows, press ctrl-alt-delete, and click on the Performance tab:



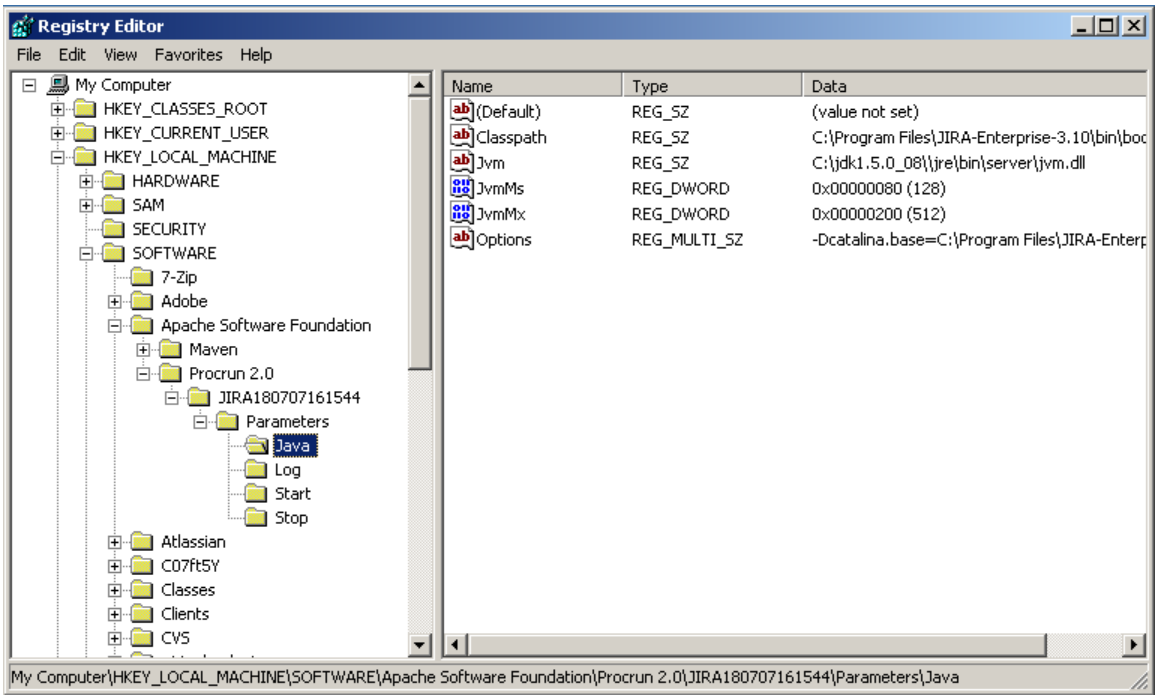
The amount marked **Available** is the amount in kilobytes you have free to allocate to JIRA (On linux, `cat /proc/meminfo` shows the memory usage). On this server we should allocate at most 214Mb. Any more, and we risk OutOfMemoryErrors due to lack of physical memory, and are certain to go into swap, which greatly decreases performance.

Tomcat Windows Service

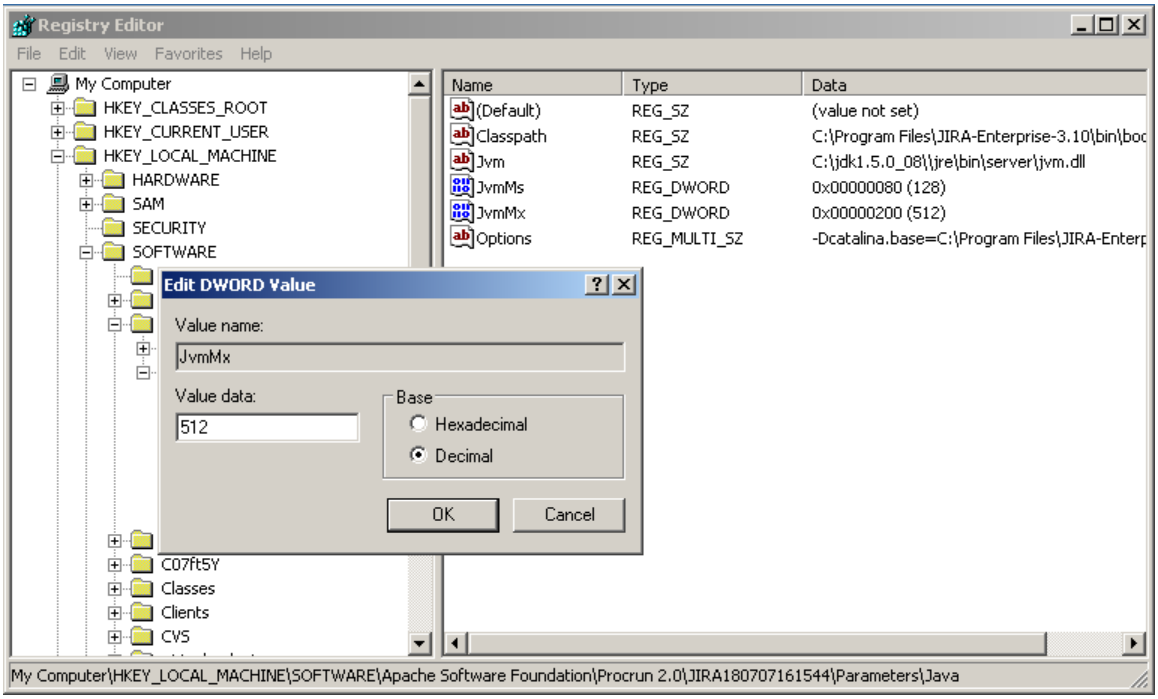
If you are running Tomcat (eg. that bundled with JIRA Standalone) on a Windows server, then JIRA should be installed as a service. To set the maximum memory limit of the service, click **Start**, then *Run..." and type `regedt32`:



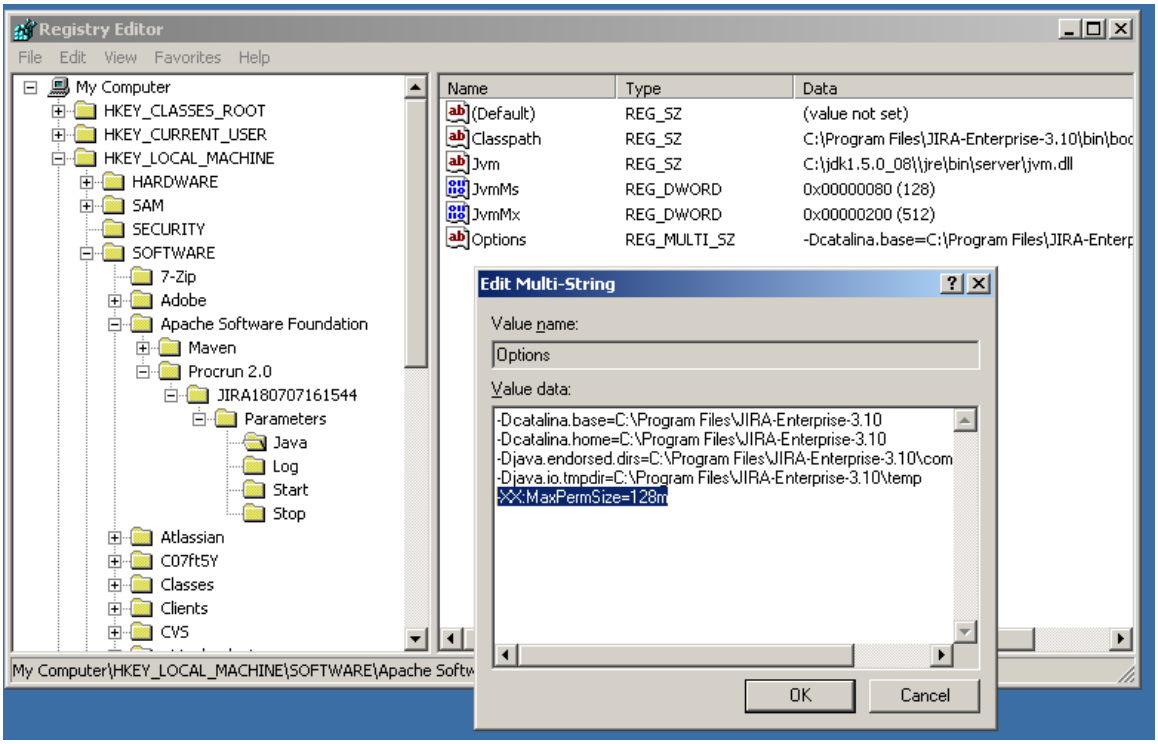
In the registry editor, click to HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Procrun 2.0 -> JIRA -> Parameters -> Java. Here you will see an entry for `JvmMx`, which is the "maximum memory" setting.



Double-click `JvmMx` to edit, change the Base to "Decimal" and adjust the value as necessary:



Additionally, if running Confluence (or other webapps) in the same Tomcat instance as JIRA, you should click `Options`, and add `-XX:MaxPermSize=128m`:



Click "OK", close the registry editor and restart the JIRA Windows service (under Start -> Settings -> Control Panel -> Administrative Tools -> Services).

Memory and options can also be adjusted using `tomcat5.exe` with commands like:

```
tomcat5 //US//JIRA --JvmMs 256 --JvmMx 256
```

and:

```
tomcat5 //US//JIRA ++JvmOptions="-XX:MaxPermSize=128m"
```

This is how the Tomcat Service docs suggest it be done; however you would need to figure out the JIRA service name first (it will be `JIRA180707161544` or something, not just `JIRA`. In Windows, you can find the JIRA service name under Start -> Settings -> Control Panel -> Administrative Tools -> Services. The name will usually begin with '**Atlassian JIRA**'). The regedt32 method described above is better, as it is less prone to failing with cryptic errors.

Note that the commands shown above apply only if you have installed the service manually. If you installed the service via the Windows installer, replace 'JIRA' with 'JIRA<timestamp value>'.

Tomcat on Windows (started manually)

If you run Tomcat (eg. from JIRA Standalone) on Windows, and are starting it manually by running `bin\startup.bat`, edit `bin\setenv.bat` and add the line:

setenv.bat

```
set JAVA_OPTS=-Xms256m -Xmx256m
```

and then restart. Adjust `256` to the maximum memory you want to allocate.

If `bin\setenv.bat` does not exist, create it.

Tomcat on Unix

Edit the `bin/setenv.sh` file and add:

setenv.sh

```
export JAVA_OPTS="-Xms256m -Xmx256m"
```

and then restart.

If `bin/setenv.sh` does not exist, create it.

Other application servers

Java applications all start by running a command

```
java <args>
```

where `<args>` varies. To increase the memory, add a `-Xmx` parameter. Eg. to use 256Mb, run

```
java -Xmx256m <args>
```

JIRA is run in an application server (Tomcat, Orion, Websphere, Weblogic, JBoss etc), and these app servers usually have scripts wrapping the actual `java` call. The `-Xmx` parameter is usually passed into these scripts through an environment variable. For app servers installed as a Windows Service, there is typically a command to run to set the -Xmx parameter for the service (eg. the tomcat service installer). Here is a list of app servers, and how to set the parameter for each:

AppServer	Startup script	Variable to edit	Example
Orion	n/a	n/a	`java -Xmx512m -jar orion.jar -userThreads`
Weblogic	$BEA_HOME/domain/<domain>/startWebLogic.sh	JAVA_OPTIONS	`export JAVA_OPTIONS="$JAVA_OPTIONS -Xmx256m"`
JBoss	bin/run.sh	JAVA_OPTS	`export JAVA_OPTS=-Xmx256m`
Resin 3.0.x	bin/httpd.sh	n/a	Start with httpd.sh -J-Xmx256m

Note that the above instructions configure JIRA to use 256 MB of memory only when it needs to. JIRA will start using less memory and will grow to maximum of 256MB if required. So after launching JIRA the System Info page will likely report that JIRA is using much less than 256MB of memory.

Permanent Generation Memory

In some cases, such as when running JIRA and Confluence together, the server can run out of a different type of memory (Permanent Generation), and if this is the case the above settings may not help solve the memory issue. A problem like this may occur when running multiple applications on the same server.

If you are experiencing this type of error, you would most likely see the following error in your logs:

```
java.lang.OutOfMemoryError: PermGen space
```

To increase the level of this memory another java parameter will need to be added:

```
-XX:MaxPermSize=128m
```

Tomcat on Windows (started manually)

If you run Tomcat (eg. from JIRA Standalone) on Windows, and are starting it manually by running `bin\startup.bat`, edit `bin\setenv.bat` and add the line:

setenv.bat

```
set JAVA_OPTS=-Xms256m -Xmx512m -XX:MaxPermSize=128m
```

and then restart. Adjust `512` to the maximum memory you want to allocate.

If `bin\setenv.bat` does not exist, create it.

Tomcat on Unix

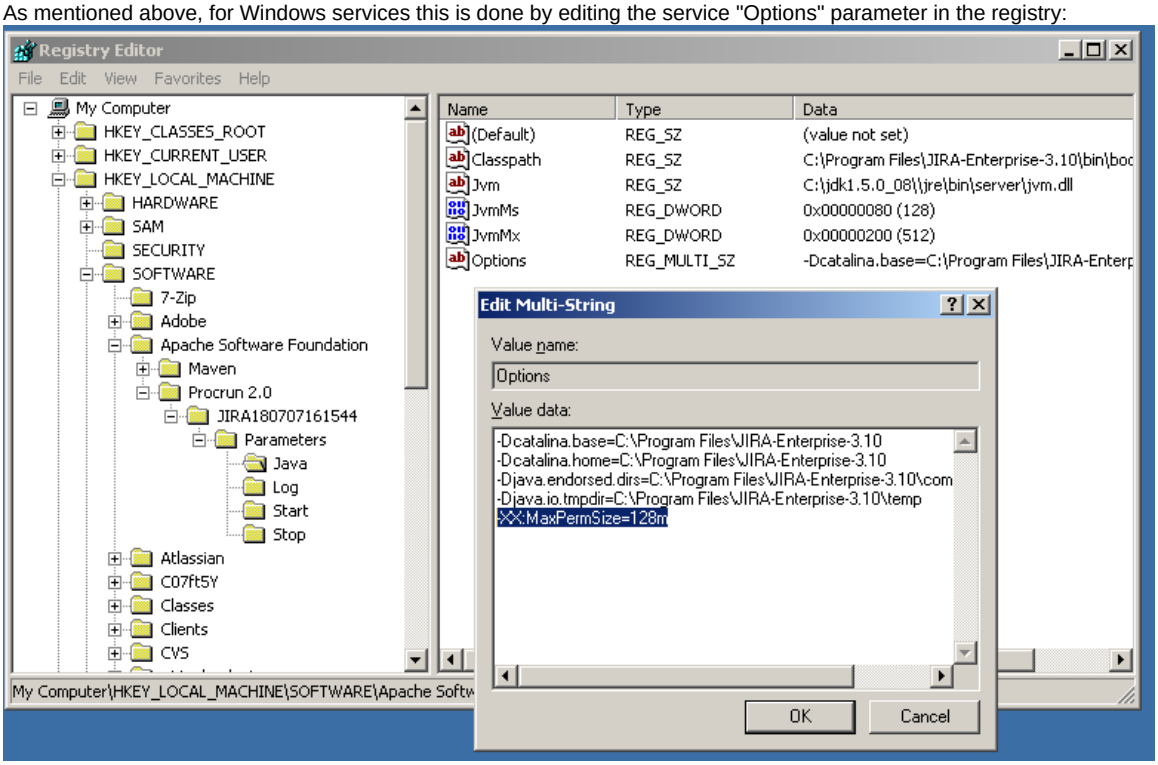
Edit the `bin/setenv.sh` file and add:

setenv.sh

```
export JAVA_OPTS="-Xms256m -Xmx512m -XX:MaxPermSize=128m"
```

and then restart.

As mentioned above, for Windows services this is done by editing the service "Options" parameter in the registry:



Database Indexing

JIRA 3.0 and later creates database indices automatically when the underlying table is created in the database. This means that if you are doing a fresh install of JIRA 3.0 (or later) you do not need to create indices manually. If you are upgrading JIRA from an earlier version (e.g. JIRA 2.6) and do not wish to create the indices manually, please follow these instructions and recreate (drop and create) JIRA's database (or remove all tables in the database) **AFTER** successfully exporting your data and before doing the import into the new version of JIRA. Removing the database will force JIRA to recreate all tables in the database and hence create all required indices.

If upgrading from JIRA 2.6.1 or earlier to JIRA 3.0 (or above), JIRA will not create indices automatically, unless the database is removed and recreated.

If you do not wish to drop and recreate JIRA's database, you can add the indices manually by running the SQL statements shown below.

The syntax for creating indices differs between databases, so consult your documentation for the your database. In addition, if you change the database tables or fields that you use in `entitymodel.xml` , you will need to change the shown SQL statements.

Below is the SQL for creating indices on PostgreSQL (you will probably need to alter this for your database):

```
CREATE INDEX action_issue ON jiraaction (issueid, actiontype);

CREATE INDEX chggroup_issue ON changegroup (issueid);

CREATE INDEX chgitem_chggrp ON changeitem (groupid);

CREATE INDEX cf_cfoption ON customfieldoption (CUSTOMFIELD);

CREATE INDEX cfvalue_issue ON customfieldvalue (ISSUE, CUSTOMFIELD);

CREATE INDEX attach_issue ON fileattachment (issueid);

CREATE INDEX subscrpt_user ON filtersubscription (FILTER_I_D, USERNAME);
CREATE INDEX subscrptn_group ON filtersubscription (FILTER_I_D, groupname);

CREATE INDEX issue_key ON jiraissue (pkey);

CREATE INDEX issuelink_src ON issuelink (SOURCE);
CREATE INDEX issuelink_dest ON issuelink (DESTINATION);
CREATE INDEX issuelink_type ON issuelink (LINKTYPE);

CREATE INDEX linktypename ON issuelinktype (LINKNAME);
CREATE INDEX linktypestyle ON issuelinktype (pstyle);

CREATE INDEX node_source ON nodeassociation (SOURCE_NODE_ID, SOURCE_NODE_ENTITY);
CREATE INDEX node_sink ON nodeassociation (SINK_NODE_ID, SINK_NODE_ENTITY);

CREATE INDEX ntfctn_scheme ON notification (SCHEME);

CREATE INDEX osgroup_name ON groupbase (groupname);

CREATE INDEX mshipbase_user ON membershipbase (USER_NAME);
CREATE INDEX mshipbase_group ON membershipbase (GROUP_NAME);

CREATE INDEX osproperty_all ON propertyentry (ENTITY_NAME, ENTITY_ID);

CREATE INDEX osuser_name ON userbase (username);

CREATE INDEX sec_scheme ON schemeissuesecurities (SCHEME);

CREATE INDEX sec_security ON schemeissuesecurities (SECURITY);

CREATE INDEX prmssn_scheme ON schemepermissions (SCHEME);

CREATE INDEX sr_author ON searchrequest (authorname);
CREATE INDEX sr_group ON searchrequest (groupname);

CREATE INDEX user_source ON userassociation (SOURCE_NAME);
CREATE INDEX user_sink ON userassociation (SINK_NODE_ID, SINK_NODE_ENTITY);

CREATE INDEX workflow_scheme ON workflowschemeentity (SCHEME);
```

Once you have created the index, you may need to tell your database to recompute its indices. For PostgreSQL, the command is `vacuumdb -U username -z -v database-name` . Consult your database documentation for your database specific command.

Generating a Thread Dump

Occasionally, JIRA may appear to 'freeze' during execution of an operation. During these times, it is helpful to retrieve a **thread dump** — a log containing information about currently running threads and processes within the Java Virtual Machine. Taking thread-dumps is a non-destructive process that can be run on live systems. This document describes the steps necessary to retrieve a **thread dump.**

The steps necessary to retrieve the **thread dump** are dependant on the operating system JIRA is running in — please follow the appropriate steps:

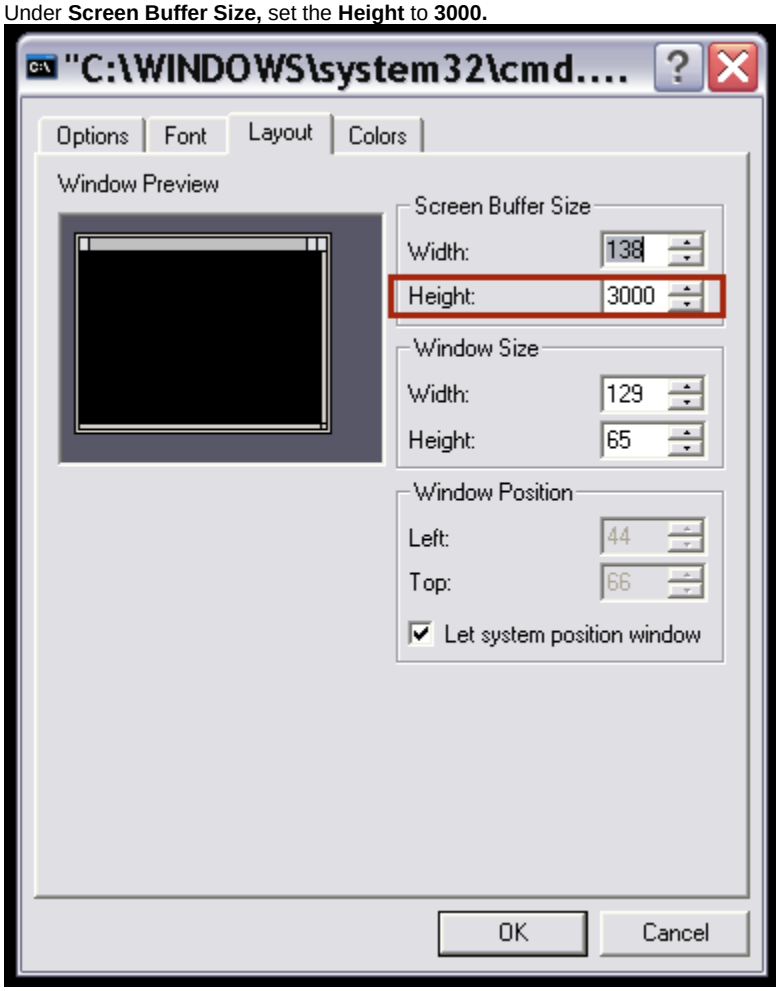
- Windows Environment

    - JIRA running from startup.bat
    - JIRA running as a Windows Service
- UNIX Environment

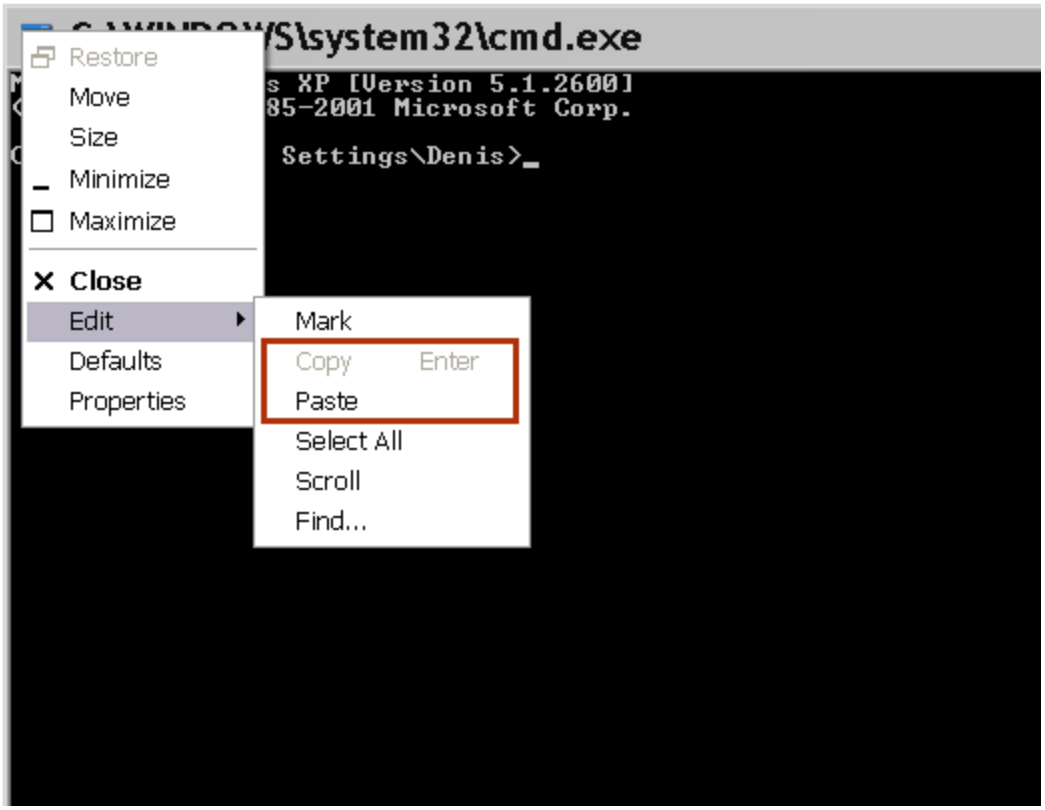
Windows Environment

JIRA running from startup.bat

1. In the **Command Console** window where JIRA is running, open the properties dialog box by right clicking on the title bar and select **"Properties".**
2. Select the **Layout** tab.
3. Under **Screen Buffer Size,** set the **Height** to **3000.**



4. Click **OK.**
5. With the same command console in focus, press **CTRL-BREAK.** This will output the thread dump to the command console.
6. Scroll back in the command console until you reach the line containing "Full thread dump". A sample thread dump is available here.
7. Right click the title bar and select **Edit -> Mark.** Highlight the entire text of the thread dump.
8. Right click the title bar and select **Edit -> Copy.** The thread dump can then be pasted into a text file.



JIRA running as a Windows Service

1. Visit http://www.adaptj.com/root/main/download and click '**Launch**'.
2. Click '**Run**' for any security warnings.
3. Select '**Process -> Thread Dump**'.
4. Under **Process Id**, select the **'...'** button.
5. From the drop-down list, select the JIRA process. Users running JIRA Standalone should select the 'Java (Tomcat) ...' option. Users running JIRA EAR/WAR should select their application server process.
6. Click '**OK**' to capture the thread dump.
7. Save the output to a file, e.g. 'threaddump.log'

UNIX Environment

1. Identify the **java** process that JIRA is running in. This can be achieved by running a command similar to:

   ```
   ps -ef | grep java
   ```

   The process will appear similarly as follows:

   ```
   keithb     910   873  1 17:01 pts/3    00:00:18 /usr/java/jdk/bin/java -Xms128m -Xmx256m
   -Xms128m -Xmx256m -Djava.awt.headless=true
   -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
   -Djava.awt.headless=true
   -Djava.endorsed.dirs=/tmp/atlassian-jira-enterprise-3.6-standalone/common/endorsed
   -classpath :
   ```

2. In order to retrieve the thread dump, execute the command

   ```
   kill -3 <pid>
   ```

   where **pid** is the process id — in this case, 910.
3. The thread dump is logged to the console in which JIRA was started. A sample thread dump is available here.

Using robots.txt to hide from Search Engines

The robots.txt protocol is used to tell search engines (Google, MSN, etc) which parts of a website should not be crawled.

For JIRA instances where non-logged-in users are able to view issues, a robots.txt file is useful for preventing unnecessary crawling of the Issue Navigator views (and unnecessary load on your JIRA server).

Editing robots.txt

JIRA (version 3.7 and later) installs the following `robots.txt` file at the root of the JIRA webapp:

```
# robots.txt for JIRA
# You may specify URLs in this file that will not be crawled by search engines (Google, MSN, etc)
#
# By default, all SearchRequestViews in the IssueNavigator (e.g.: Word, XML, RSS, etc) and all
IssueViews
# (XML, Printable and Word) are excluded by the /sr/ and /si/ directives below.

User-agent: *
Disallow: /sr/
Disallow: /si/
```

Alternatively, if you already have a `robots.txt` file, simply edit it and add **`Disallow: /sr/`** and **`Disallow: /si/`**.

Publishing robots.txt

The `robots.txt` file needs to be published at the root of your JIRA internet domain, e.g. **`jira.mycompany.com/robots.txt`**.

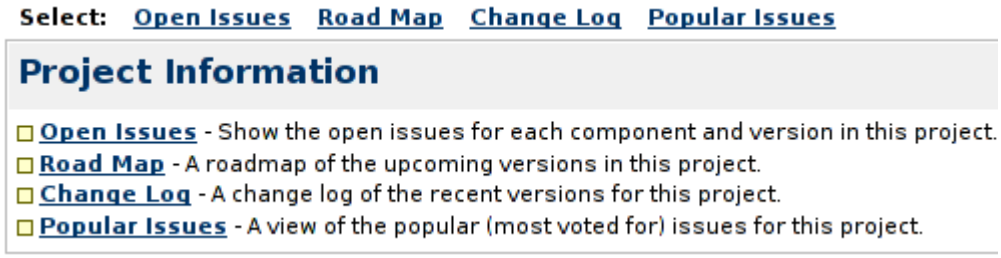
If your JIRA instance is published at **`jira.mycompany.com/jira`**, change the contents of the file to `Disallow: /jira/sr/` and `Disallow: /jira/sr/`. However, you still need to put `robots.txt` file in the root directory, i.e. `jira.mycompany.com/robots.txt` (not `jira.mycompany.com/jira/robots.txt`).

Appendix A - Extending JIRA

JIRA is very flexible, and has a number of extension points where JIRA's data can be queried or its functionality extended. You can also download the JIRA Plugin Development Kit. This contains full source code for seven working plugins (and growing), skeleton project templates for creating your own plugins, full JIRA API documentation and all library dependencies.

JIRA Plugins
For information on installing and/or enabling existing plugins, please read the Managing JIRA's Plugins documentation.

Custom Field Types	JIRA comes with various custom field types defined. New types can be written and plugged into JIRA. See the Writing Custom Field Types Tutorial for more information.
User Formats	JIRA comes with many options to change the look and feel of features in the system. User formats are a feature that can be customised by plugins. You can write your own User Format plugin to change the display of user details in JIRA, e.g. display a profile picture. See the User Format Plugin Guide for more information.
Gadgets	New gadgets can be created by writing an XML descriptor file, packaged as an Atlassian plugin. See Writing an Atlassian Gadget for more information.
Reports	JIRA comes with various reports built-in. Using the plugin system, new reports can be written, providing new ways of viewing and summarising JIRA's data.
Workflow functions and conditions	JIRA's issue workflow (states and state transitions an issue can go through) can be customized through the web interface (see the workflow documentation. The workflow engine (OSWorkflow) provides hooks where you can plug in your own behaviour: • Run arbitrary Java when a certain transition occurs, via post-functions • Limit visibility of transitions to certain users, via conditions • Validate input on transition screens (eg. in comments), via validators. See the guide to creating custom workflow elements for how to write your own workflow post-functions, conditions and validators. Once written, these can be packaged as plugins and reused.

Issue and Project Tabs	When viewing an issue, some issue information (comments, change history) is displayed in tabs: Likewise, the 'Browse Project' page contains tab panels displaying project information:  By writing a plugin, you can add new issue or project tab panels to JIRA. For instance, you may wish to display project/issue data pulled in from an external source. This is how JIRA's Subversion. See the plugin guide for more information on writing these plugin types.
Listeners	JIRA has a complete event subsystem which fires events whenever anything happens. For example an `ISSUE_CREATED` event is fired whenever an issue is created.A listener is just a class which implements a `JiraListener` interface and is called whenever events occur in JIRA. Using those events, you can then perform any action you want. For example the email sent by JIRA is driven by the `MailListener`. This is useful when you want to drive or affect external systems from events which occur within JIRA - usually used to *push* data into outside systems. For more information, read the listeners documentation.
Services	Services are classes which implement the `JiraService` interface. When installed, you specify an update period and JIRA will call the `run()` method of your service periodically. A sample service is provided called `POPCommentService`. This service checks a particular POP mailbox periodically and if it finds messages, tries to extract an issue key from the subject. If the subject contains a key, the body of the mail is added as a comment to the message. Services are useful when you want to periodically *pull* data into JIRA from outside systems. For more information, see the services guide.
SOAP and XML-RPC remote interfaces	JIRA has a growing SOAP and XML-RPC interface. This enables you to drive JIRA automatically from external systems. For example you can have a Java program, Perl script or C# client add issues to JIRA. See the JIRA RPC overview for general information. For building RPC clients, check out the SOAP client tutorial and XML-RPC client tutorial. New RPC endpoints can also be added to JIRA as plugins - see RPC Endpoint Plugins.
Java	JIRA has a full set of Java APIs that can be used to update information with in JIRA. You can view the API here. JIRA commercial customers get full access to the JIRA source (see bottom of the downloads page), so you can modify JIRA itself if necessary. See the Building JIRA from Source page for more information.

Managing JIRA's Plugins

On this page:

- About Plugins
  - About 'Version 1' Plugins and 'Version 2' Plugins
- Installing a JIRA Plugin
- Updating a JIRA Plugin
- Disabling a JIRA Plugin
- Enabling a JIRA Plugin

About Plugins

A plugin is a .jar file (containing code, resources and a configuration file) that can be installed into an Atlassian product to add new functionality or change the behaviour of existing features.

You can use plugins to customise and extend the functionality of JIRA in a variety of ways, including:

- creating new dashboard gadgets (previously known as 'portlets')
- creating new reports
- creating new types of custom fields
- customising workflow
- modifying the availability of 'issue operations' links ('Create Issue', etc)

JIRA ships with a number of preinstalled plugins, and many more are available for download from the Atlassian Plugin Exchange and the JIRA Extensions site.

You can also create your own plugins — please visit the JIRA Development Hub for information on building '*Version 1*' plugins, or visit the Plugin Framework Documentation for information on building '*Version 2*' (OSGi) plugins. These plugins can be installed in your JIRA instance as described below.

Installed JIRA plugins can be enabled or disabled.

Every plugin is made up of one or more *plugin modules*. A single plugin may do many things, and each module represents a single function of the plugin. Individual modules can be enabled or disabled, although this is not recommended as modules can often depend on each other. To do so safely would require knowledge of the module interdependencies.

About 'Version 1' Plugins and 'Version 2' Plugins

There are two types of Atlassian plugins:

- **'Version 1' Plugins** — These plugins are compatible with the 'old' plugin framework that was used in JIRA prior to version 4.0, and also with the 'new' plugin framework used in JIRA version 4.0 and later. 'Version 1' plugins are installed into your JIRA installation directory under `atlassian-jira/WEB-INF/lib/`.
- **'Version 2' Plugins** — These plugins are compatible with the 'new' plugin framework used in JIRA version 4.0 and later. They are installed into your JIRA home directory under `plugins/installed-plugins/`. 'Version 2' plugins are deployed in an internal OSGi container to isolate the plugin from the core code and other plugins, and to allow sophisticated dependency management. 'Version 2' plugins have to be specifically declared as such, using the `plugins-version="2"` attribute in the plugin's configuration file (`atlassian-plugin.xml`).

Installing a JIRA Plugin

Once you have downloaded or created your plugin jar, follow these steps to install it in your JIRA instance:

1. Shut down JIRA.
2. Copy your plugin jar (e.g. '**EXAMPLE_PLUGIN.jar**') into the appropriate plugins directory:
    - If you are installing a '**Version 1**' plugin, copy the jar into your JIRA installation directory under `atlassian-jira/WEB-INF/lib/`.
    - If you are installing a '**Version 2**' (OSGi) plugin, copy the jar into your JIRA home directory under `plugins/installed-plugins/`.
3. Start up JIRA.
4. Go to 'Administration > Plugins'. Your plugin (e.g. '**EXAMPLE_PLUGIN**') will be listed and enabled.

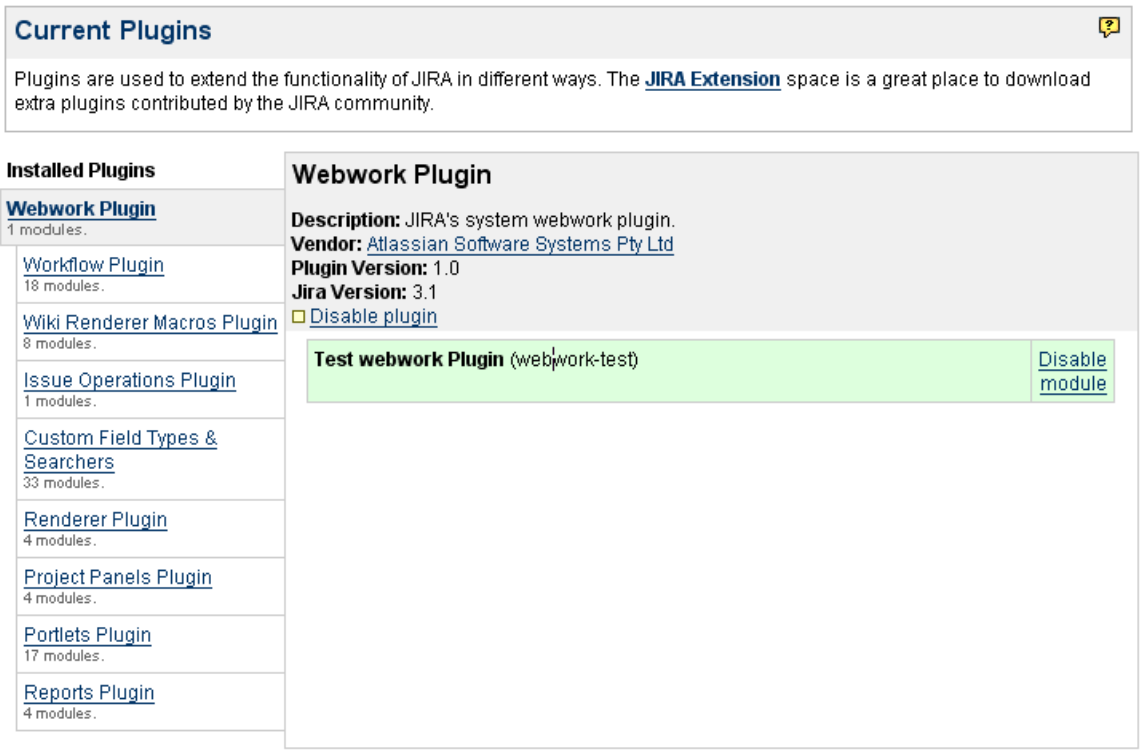
Updating a JIRA Plugin

The process for updating a plugin is similar to installing a new one. Simply download or create your updated plugin jar, then follow these steps to install it in your JIRA instance:

1. Shut down JIRA.
2. Remove the old version of the plugin jar and copy your updated plugin jar (e.g. '**EXAMPLE_PLUGIN.jar**') into the appropriate plugins directory:
    - If you are installing a '**Version 1**' plugin, copy the jar into your JIRA installation directory under `atlassian-jira/WEB-INF/lib/`.
    - If you are installing a '**Version 2**' (OSGi) plugin, copy the jar into your JIRA home directory under `plugins/installed-plugins/{}`.
3. Start up JIRA.
4. Go to 'Administration > Plugins'. The updated version of your plugin (e.g. '**EXAMPLE_PLUGIN**') will be listed and enabled.

Disabling a JIRA Plugin

1. Log in as a user with the **'JIRA Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. On the panel on the left, under the title 'System', click the link labelled '**Plugins**'.
4. This will bring up the '**Current Plugins**' page. The page lists all the installed plugins, e.g.:



5. Locate the plugin that you want to disable, and click the '**Disable plugin**' link.

Also note that JIRA can automatically disable plugins if they cause errors on startup or initialisation. This is usually caused by having an old version of the plugin that is not compatible with your current version of JIRA. If this occurs, it will be indicated on the 'Current Plugins' page (shown above). You will need to stop JIRA, install the newer version of the plugin and restart JIRA.

Enabling a JIRA Plugin

Note: newly installed plugins are enabled by default.

To enable a plugin:

1. Follow the above steps for disabling a plugin, but click '**Enable plugin**' instead.

Listeners

Listeners are unique to JIRA, and a very powerful way to extend it.

JIRA has a complete event subsystem which fires events whenever anything happens inside the application. For example an `ISSUE_CREATED` event is fired whenever an issue is created.

A Listener is a class that implements one of the Listener interfaces. It is then called whenever events occur in JIRA. Using those events, you can then perform any action you want. For example the email sent by JIRA is driven by the MailListener.

Listeners are most useful when you want to drive or affect external systems from events which occur within JIRA.

On this page:

- Listener Interfaces
- Example Listeners
- Registering a Listener
- Editing Listener Properties
- Removing a Listener
- Custom Events

Listener Interfaces

There are currently two different concrete Listeners within JIRA (both of which extend the base JiraListener interface).

com.atlassian.jira.event. **JiraListener**	The base interface which all other JIRA listener interfaces extend. Covers core listener properties like uniqueness, description, parameters etc. API doc
com.atlassian.jira.event.issue. **IssueEventListener**	The main listener interface in JIRA, used whenever anything happens to an issue. API doc
com.atlassian.jira.event.user. **UserEventListener**	This listener is called whenever anything happens to a user within JIRA. API doc

Example Listeners

The examples provided may be freely used and modified for use in your own environment. The source of all examples is available and should give you good overview of how simple it is to write your own listeners. Both example listeners are included with JIRA 2.1, and both implement `UserEventListener` and `IssueEventListener`.

- **DebugListener** (source) — This is a very simple listener that prints events and their content to System.out whenever they are received. To test this listener, add a listener with the class `com.atlassian.jira.event.listeners.DebugListener` .
- **MailListener** (source) — This listener is how mail notifications are currently sent from within JIRA, and a good example of a more complex listener. It basically listens for events, and turns them into email notifications using Velocity templates to generate the mail bodies.
  This listener is usually always turned on in JIRA — see Email Notifications for more details. If you want to write more complex or more specific notifications, you can disable the internal MailListener and add your own.

Other examples of useful tasks that can be accomplished with listeners are:

- **Send SMS or IM notifications** — A listener could easily send notifications for various events via SMS or instant messenger (e.g. ICQ or AIM) - or anywhere that you have a Java library to send messages.
- **Group notifications** — A listener could notify certain groups of issue changes, depending on the content of the issue. For example any issue containing "windows" in the environment could notify your "windows-developers" group.

Registering a Listener

To register a listener:

1. Make sure your listener class is in the classpath where JIRA can see it - the best locations are usually the `WEB-INF/classes` or `WEB-INF/lib` (as a JAR) directories within the JIRA web application.
2. Log in as a user with the **'JIRA System Administrators'** global permission.
3. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
4. Click the **'Listeners'** link in the **'System'** section of the left-hand navigation column.
5. Enter a **'Name'** (an appropriate name for this listener) and **'Class'** (the fully qualified class of your listener) for your listener and click **'Add'**.
6. The listener will now be added. Click **'Edit'** for your listener to edit its properties.

Editing Listener Properties

You can edit a listeners properties by clicking **'Edit'** for that listener in the **'Listeners'** section of the **'Administration'** tab.

Removing a Listener

To remove a listener, click **'Del'** for that listener in the **'Listeners'** section of the **'Administration'** tab.

Custom Events

With the ability to add custom events to JIRA, the Listener must be updated to deal with the event as appropriate. This is possible by providing an implementation for the method `customEvent(IssueEvent event)` in the Listener. For example, the MailListener implementation passes the custom event on for notification processing. The DebugListener logs that the custom event has been fired.

Services

A service is a class that runs periodically within JIRA. Since a service runs inside JIRA, it has the ability to use all of the JIRA API — and, as it is written in Java, it can use any Java libraries.

Services are useful because they enable you to integrate with external systems by pulling data into JIRA periodically. JIRA comes with a number of pre-written services, and custom services can be written and plugged in at runtime. If you want a service to perform typical operations on JIRA issues (eg. close a list of issues meeting certain criteria), then the Jelly Service can be configured to run a custom Jelly script.

On this page:

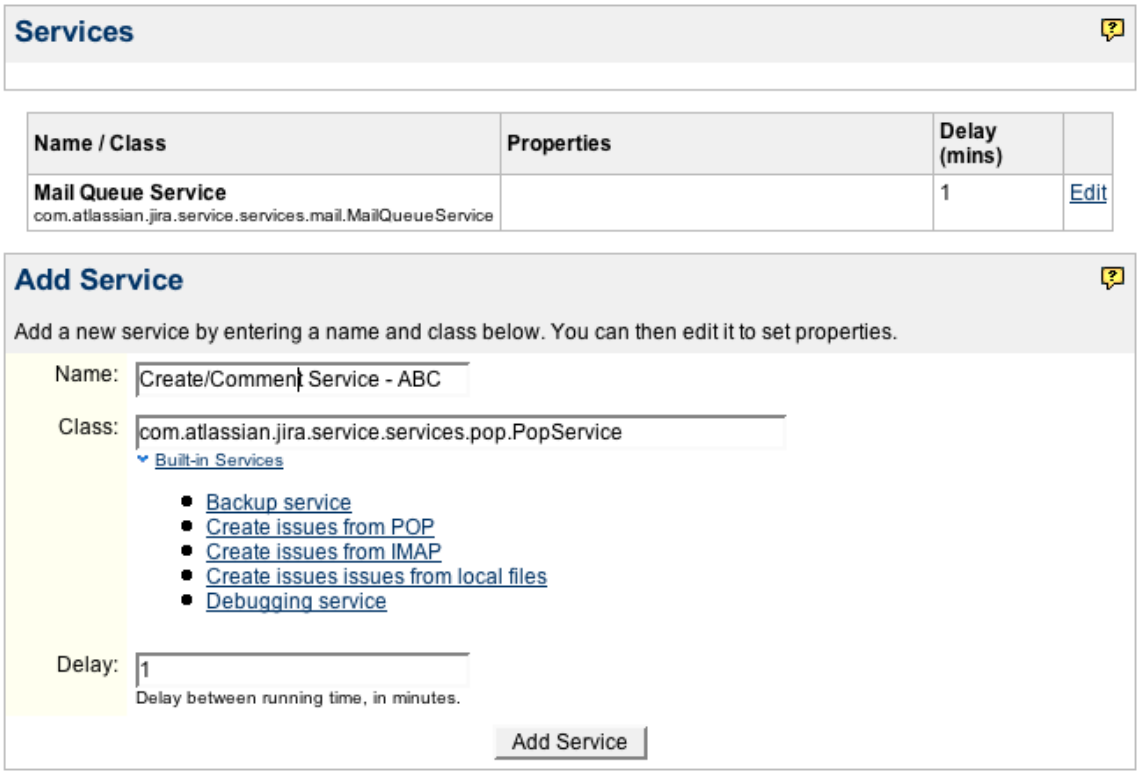
- Registering a Service
  - Mail Service Properties
- Editing Service Properties
- Removing a Service
- Built-in Services
- Message Handlers
  - CreateIssueHandler
  - CreateOrCommentHandler
  - FullCommentHandler
  - NonQuotedCommentHandler
  - RegexCommentHandler
  - CVSLogHandler
- Message Handlers and Events
- Building Services from source

Registering a Service

Services are set up as follows.

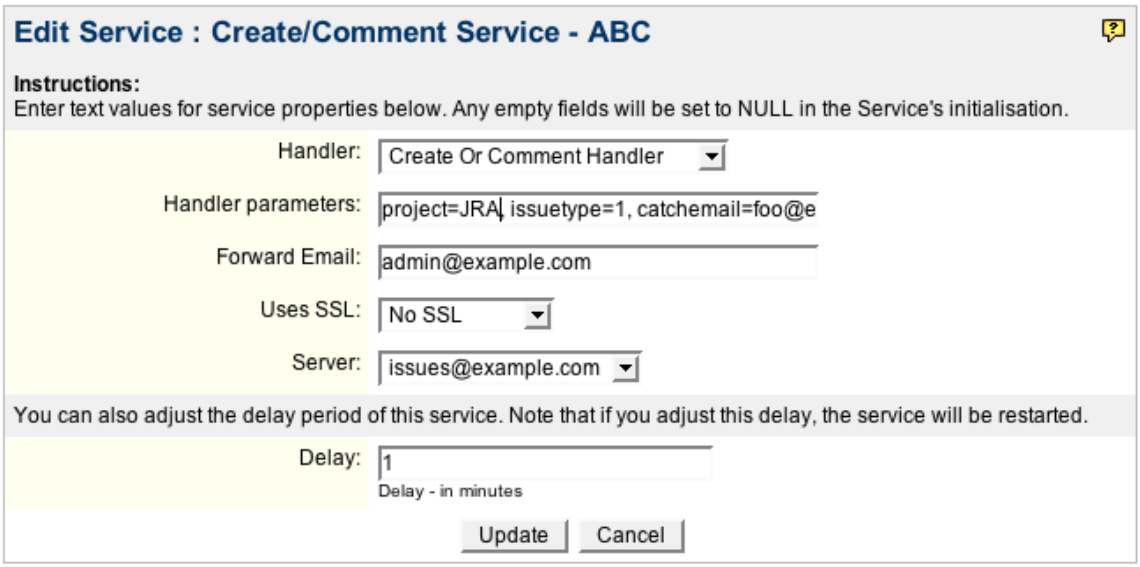
1. For custom-written services, make sure your service class is in the classpath where JIRA can see it — the best locations are usually the **WEB-INF/classes** or **WEB-INF/lib** (as a JAR) directories within the JIRA web application.
2. Log in as a user with the **'JIRA System Administrators'** global permission.
3. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
4. Under the **'System'** sub-menu on the left, click the **'Services'** link.
5. Fill out the **Add Service** form with the following parameters:
   - **Name** — a descriptive name for this service.
   - **Class** — the fully qualified class of your service. Will likely be of the form **com.atlassian.jira.service.services.*type.Type* Service.** See Sample services for provided service class names.
   - **Delay** — the delay (in minutes) between service runs.

   For example, adding a POP Service:



   Now click **Add Service**
6. This will bring up the **'Edit Service'** screen to configure the service. Depending on the service, you may now be required to specify a MessageHandler, a helper class that processes email messages. MessageHandlers are configured with a parameter string, a

comma-separated list of name=value pairs. Consult the tables below as to what parameters each MessageHandler accepts. The following screenshot shows a CreateIssueHandler being attached to a POP Service:



Mail Service Properties

In addition to Message Handlers, the mail services POP Service and IMAP Service can be further configured with further properties on how the mail is found and handled.

Forward Emails
If the mail service is unable to handle a message you can define an email address to forward these messages to. Just add the desired email address into the 'ForwardEmail' textbox.

> **Note**
> You will need to configure a SMTP mail server before this functionality can be used.

SSL
The mail service can be configured to connect to the email server using an SSL connection. To do this select the appropriate SSL connection in the 'Use SSL' select list. If you do not want JIRA to use SSL, select 'No SSL'.

If you are using SSL, you will need to preload the IMAPS/POPS server's public key in JIRA (actually, the Java virtual machine running JIRA). See Connecting to SSL Services for information on how to do this.

Folder (IMAP Only)
For the IMAP Service it is possible to specify the folder you wish JIRA to read when scanning for messages. To do this, add the desired folder name to the 'Folder' field.

> If a folder is not specified the mail service will default to 'INBOX'.

Editing Service Properties

To edit a service's properties,

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu on the left, click the **'Services'** link.
4. Click the **Edit** for the relevant service.

For example, to change the interval at which email is sent from JIRA, edit the **Mail Queue Service** and change the **Delay** from the default value of 1 minute.

Removing a Service

To remove a service,

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'System'** sub-menu on the left, click the **'Services'** link.
4. Click the **Del** for the relevant service.

Built-in Services

JIRA has some useful services out of the box, which may be used as-is or modified for use in your own environment. The source code for all built-in services is available and should give you a good overview of how simple it is to write your own services. All built-in services are included with JIRA, and need only be configured to be used.

Export Service (source)
The Export Service is useful for periodically backing up JIRA. It exports all data from JIRA every time it is run, into a directory supplied as a parameter. The export files are timestamped, thus the service can act as a backup system.

To test this service, add a service with the class **com.atlassian.jira.service.services.export.ExportService**. JIRA sets up an ExportService during the setup wizard, so you may find you already have one.

POP Service (source)
The POP service reads messages from a defined POP3 mail box, and then performs operations within JIRA based on the message contents. A MessageHandler (see below for more information) is configured for each instance of the POP Service, which determines how the message is handled.

To test this service, add a service with the class **com.atlassian.jira.service.services.pop.PopService** and configure the POP details and message handler in the properties of the service.

To make the POP service even more useful, you can connect it with the email notifications sent by JIRA. Simply set the `FROM` address in the MailListener to be the same address as the POP mailbox being monitored. This allows you to do things like reply to email notifications and have your replies added as comments on the issue.

IMAP Service (source)
Similar to POP service, except that it reads from an IMAP mailbox's "INBOX" instead. Like the POP service, it removes emails after reading.

To test this service, add a service with the class **com.atlassian.jira.service.services.imap.ImapService** and configure the IMAP details and message handler in the properties of the service.

File Service (source)
The File service is very much like the POP service above, except that instead of reading emails from a POP mailbox, they are read from a directory on disk. This is useful because you do not need an anonymous POP mail box (which could be a potential security risk) to use it. Instead you can simply get your mail server to dump incoming email messages into a particular directory, which the File service scans periodically.

The setup of this service is identical to the POP Service above, except that the service class is **com.atlassian.jira.service.services.file.FileService** and the service is configured with the directory to watch instead of the POP mailbox details. Both File and POP services can use the same MessageHandlers.

Jelly Service (source)
Jelly is a scripting language which allows tasks in JIRA to be automated. The Jelly Service periodically runs a Jelly script. For example, you could use this to periodically run a search request, loop through the results and add a comment, change the issue state (see the Jelly examples. If you're considering writing a custom service, often a periodically invoked Jelly script may be an easier alternative.

JIRA Service classes must all extend com.atlassian.jira.service.JiraService. Most do so by extending com.atlassian.jira.service.AbstractService or some more specialised subclass.

Message Handlers

POPService, IMAPService and FileService above use MessageHandlers (API doc) to perform operations within JIRA based on the format of incoming email messages.

You can design your own MessageHandlers to integrate JIRA with your own processes, and plug them into any of these three services via the **Administration** interface. (Please also see Adding your own email handling classes).

MessageHandlers are configured with a comma-separated list of name-value pairs, known as the **handler parameters.**

There are a number of default message handlers that ship with JIRA, described below:

CreateIssueHandler

com.atlassian.jira.service.util.handler. **CreateIssueHandler** | API doc | Source

This message handler creates a new issue for each incoming message.

CreateIssueHandler parameters

Parameter	Meaning	Value
project	Default project where new issues are created.	Project key, e.g. `JRA`
issuetype	Default type for new issues.	Integer representing issue type: 1. Bug 2. New Feature 3. Task 4. Improvement
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:) are processed. **Note:** if this parameter is set, emails addressed to anyone other than the specified recipient will be **deleted.** Please note, setting this parameter also suppresses the generation of the addressee as watcher.	e.g. `issues@mycompany.com`
ccassignee	If an email has a Cc address listing an assignable user already present in JIRA, by default JIRA will **assign** the issue to that user. In JIRA 3.1 and above, if you do not want this behaviour, set **ccassignee** to **false.** By default (if not supplied), `ccassignee` is set to true.	`true` or `false`
ccwatcher	If an email has a **To, Cc** or **Bcc** address listing a user already present in JIRA, it is possible to automatically add that user to an issue's watchers list, by setting the `ccwatcher` parameter to `true` . Please note that users created by the `createusers` parameter cannot be added to an issue's watchers list by the `ccwatcher` parameter. Users must already exist in JIRA's userbase, and must have an email address. By default (if not supplied), `ccwatcher` is set to false.	`true` or `false`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: • `ignore` — Ignore the email and do nothing • `forward` — Forward the email to the address set in the "Forward Email" text field • `delete` — Delete the email permanently. Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

CreateOrCommentHandler

com.atlassian.jira.service.util.handler. **CreateOrCommentHandler** | API doc | Source

This message handler creates a new issue, or adds a comment to an existing issue. If the subject contains an issue key, the message is added as a comment to that issue. If no issue key is found, a new issue is created in the default project.

CreateOrCommentHandler parameters

Parameter	Meaning	Value
project	Default project where new issues are created. The `project` parameter is only relevant for issue creation, not for issue commenting. If an email contains an issue key in the email subject, and that issue exists in the JIRA instance, the handler will add the email as a comment on the issue, regardless of which project the issue is in.	Project key, e.g. `JRA`
issuetype	Default type for new issues.	Integer representing issue type: 1. Bug 2. New Feature 3. Task 4. Improvement
stripquotes	If set (to anything), quoted text is removed from comments.	(anything)
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	e.g. `issues@mycompany.com`
ccassignee	If an email has a Cc address listing an assignable user already present in JIRA, by default JIRA will **assign** the issue to that user. In JIRA 3.1 and above, if you do not want this behaviour, set **ccassignee** to **false.** By default (if not supplied), `ccassignee` is set to true.	`true` or `false`
ccwatcher	If an email has a **To, Cc** or **Bcc** address listing a user already present in JIRA, it is possible to automatically add that user to an issue's watchers list, by setting the `ccwatcher` parameter to `true` . Please note that users created by the `createusers` parameter cannot be added to an issue's watchers list by the `ccwatcher` parameter. Users must already exist in JIRA's userbase, and must have an email address. By default (if not supplied), `ccwatcher` is set to false.	`true` or `false`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` — Ignore the email and do nothing 2. `forward` — Forward the email to the address set in the "Forward Email" text field 3. `delete` — Delete the email permanently Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

FullCommentHandler

com.atlassian.jira.service.util.handler. **FullCommentHandler** | API doc | Source

This message handler creates a comment based on the entire body of the email received.

The issue to use is chosen from the first issue key found in the email subject. The author of the comment is taken from the from address of the email.

FullCommentHandler parameters

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	e.g. `issues@mycompany.com`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` --- Ignore the email and do nothing 2. `forward` --- Forward the email to the address set in the "Forward Email" text field 3. `delete` --- Delete the email permanently. Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

NonQuotedCommentHandler

com.atlassian.jira.service.util.handler. **NonQuotedCommentHandler** | API doc | Source

This message handler also creates a comment, but only uses the "non quoted" lines of the email body. A quoted line is any line that starts with">" or "|".

The issue to use is chosen from the first issue key found in the email subject. The author of the comment is taken from the from address of the email.

NonQuotedCommentHandler parameters

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	e.g. `issues@mycompany.com`

bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` — Ignore the email and do nothing 2. `forward` — Forward the email to the address set in the "Forward Email" text field 3. `delete` — Delete the email permanently. Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

RegexCommentHandler

com.atlassian.jira.service.util.handler. **RegexCommentHandler** | API doc | Source

This message handler creates a comment from an email body - but ignores any part of the email body past a specified marker or separator. For mail systems like Lotus Notes and Outlook, emails are separated from the quoted email by some predictable text like "—– Original Message -–" or "Extranet\n email.address/DOM/REG/CONT/CORP@CORPMAIL" The RegexCommentHandler can take any valid regular expression — and in fact filter quoted mails from various different mail systems simultaneously.

- If the pattern is found, it returns the text before the first match — and discards the rest of the email body
- If the pattern is not found, it returns the email body unchanged
- If the regex is not specified, it returns the email body unchanged
- If there is any error (i.e. regex expression error), it returns the email body unchanged.

RegexCommentHandler parameters

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:)are processed.	e.g. `issues@mycompany.com`
splitregex	Regular expression matching the text separating the mail from any previous mails. **Note that the regexp must begin and end with a delimiter character, typically '/'..** Also note that currently, commas are not allowed in regexps, as commas are used to separate handler parameters and there is not (as yet) an escape syntax.	e.g. `/-{}{}{}{}—{}\s*Original Message\s*{}—/` or `/_____*/`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` — Ignore the email and do nothing 2. `forward` — Forward the email to the address set in the "Forward Email" text field 3. `delete` — Delete the email permanently Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

CVSLogHandler

com.atlassian.jira.service.util.handler. **CVSLogHandler** | API doc | Source

This message handler parses CVS Log messages and adds the relevant sections as a comment.

The comment is added to *any* issue whose key is mentioned in the CVS commit message.

For instance if you commit to CVS with the message " *This commit fixes JRA-57 and JRA-58.* ", a comment will be added to issues JRA-57 and JRA-58.

The body of the comment includes the commit message entered by the developer and the files involved in the commit.

> **Warning**
> JIRA no longer uses CVSLogHandler for its CVS integration — this service is kept here purely as an example.

To use this message handler, setup your CVS server to email commit messages using something like SyncMail.

CVSLogHandler parameters

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from an unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername` . If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc are processed.	e.g. `issues@mycompany.com`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` — Ignore the email and do nothing 2. `forward` — Forward the email to the address set in the "Forward Email" text field 3. `delete` — Delete the email permanently Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Message Handlers and Events

The message handlers will dispatch a JIRA event depending on the actions they perform. For more information on JIRA events please refer to the Notification Schemes section.

The CreateIssueHandler will dispatch an 'Issue Created' event whenever it creates a new issue.

The CreateOrCommentHandler will dispatch one of 'Issue Created', 'Issue Commented' or 'Issue Updated' events:

- 'Issue Created' event is dispatched whenever it creates a new issue.
- 'Issue Commented' event is dispatched if the issue already exists and a comment is added only.
- 'Issue Updated' event is dispatched if the issue already exists and a comment with attachment(s) is added.

Each of the comment handlers (FullCommentHandler, NonQuotedCommentHandler, RegexCommentHandler and CVSLogHandler) will dispatch the 'Issue Commented' event if the message only contains a comment. However, if the message contains an attachment as well, it will dispatch the 'Issue Updated' event instead.

Building Services from source

To build any of the linked sample code:

- If you have JIRA Standalone, download to the **external-source/src/** directory, and read the instructions in **external-source/README.txt** for build instructions.
- If you have the JIRA WAR/Webapp distribution, download the source to the **src/** directory. The code will be compiled into the webapp when `build.sh/build.bat` is run.

Jelly Tags

Jelly is a scripting and templating language from Apache's Jakarta project. It is similar to Ant, in that scripts are XML, and each tag maps to a Java class, but has a more sophisticated internal pipeline model for tag interaction, much like JSP taglibs. See the Jelly website for more details.

JIRA comes with a number of Jelly tags implementing core operations in JIRA. This provides a scriptable interface to JIRA. There are many possible uses for JIRA Jelly tags, the most common being importing data into JIRA from other systems, and automating common administrative tasks (see the examples below).

On this page:

- Enabling Jelly
- Running a Jelly script
- Writing a Jelly script
    - jira:AddComment
    - jira:AddComponent
    - jira:SelectComponentAssignees
    - jira:AddUserToGroup
    - jira:AddVersion
    - jira:CreateGroup
    - jira:AssignIssue
    - jira:CreateIssue
    - jira:LinkIssue
    - jira:TransitionWorkflow
    - jira:CreateProject
    - jira:CreateUser
    - jira:RemoveUser
    - jira:CreatePermissionScheme
    - jira:AddPermission
    - jira:Login
    - jira:CreateCustomField
    - jira:AddFieldToScreen
    - jira:AttachFile
    - jira:RunSearchRequest
    - jira:AddActorsToDefaultProjectRole
    - jira:AddActorsToProjectRole
    - jira:CreateProjectRole
    - jira:DeleteProjectRole
    - jira:GetDefaultRoleActors
    - jira:GetProjectRole
    - jira:GetProjectRoleActors
    - jira:IsProjectRoleNameUnique
    - jira:RemoveActorsFromProjectRole
    - jira:RemoveActorsFromDefaultProjectRole
    - jira:UpdateProjectRole
- Beta Tags
- Sample scripts

Enabling Jelly

JIRA's Jelly support is disabled by default, as it is a potential security hazard. To enable Jelly support, set the **jira.jelly.on** system property when starting your application server. System properties are set with parameters to the `java` command, e.g. `java -Djira.jelly.on=true ...`.

How to set this property depends on your application server. For example, with Tomcat (JIRA standalone distribution), set the environment variable **`JAVA_OPTS=-Djira.jelly.on=true`**, or when running JIRA Standalone as a service, set the service JVM parameter.

Running a Jelly script

To run a Jelly script once:

1. Log in as a user with the **'JIRA System Administrators'** global permission.
2. Bring up the administration page by clicking either the **'Administration'** link on the top bar or the title of the Administration box on the dashboard.
3. Under the **'Options & Settings'** sub-menu in the left-hand navigation column, click the **'Jelly Runner'** link.
4. Paste your Jelly script into the text area.

To run a Jelly script periodically:

Configure a service with the following class: `com.atlassian.jira.jelly.service.JellyService`.

Writing a Jelly script

Scripts are generally of the form:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<!--
    Add your own Jelly XML here
-->
</JiraJelly>
```

JIRA Enterprise has a few extra tags available; to use them, the outer tag should instead be:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
<!--
    Add your own Jelly XML here
-->
</JiraJelly>
```

In addition to the JIRA tags, you can use tags from the email, http, soap, sql and core Jelly taglibs. More can be added by the user if necessary.

Many of JIRA's Jelly tags set *context variables,* so subsequent tags can refer to their output by dereferencing the context variable (e.g. `${jira.new.username}`). Other tags let you explicitly specify the name of a variable to store some output in, e.g., <jira:CreateUser> has `issueKeyVar` and `issueIdVar` parameters:

```
<jira:CreateIssue project-key="TP" summary="Issue One" issueKeyVar="issuekey" issueIdVar="issueid"/>
          Raised issue ${issuekey} with id ${issueid}
```

Note that the variable is only set **after** the tag is closed, not inside the tag.

Due to this variable interpolation, if your text contains anything of the form `${something}` , you need to escape this as `$${something}` to prevent the 'variable' being expanded to a blank string.

The list of currently available tags:

- jira:AddComment
- jira:AddComponent
- jira:SelectComponentAssignees
- jira:AddUserToGroup
- jira:AddVersion
- jira:CreateGroup
- jira:AssignIssue
- jira:CreateIssue
- jira:LinkIssue
- jira:TransitionWorkflow
- jira:CreateProject
- jira:CreateUser
- jira:RemoveUser
- jira:CreatePermissionScheme
- jira:AddPermission
- jira:Login
- jira:CreateCustomField
- jira:AddFieldToScreen
- jira:AttachFile
- jira:RunSearchRequest
- jira:AddActorsToDefaultProjectRole
- jira:AddActorsToProjectRole
- jira:CreateProjectRole
- jira:DeleteProjectRole
- jira:GetDefaultRoleActors
- jira:GetProjectRole
- jira:GetProjectRoleActors
- jira:IsProjectRoleNameUnique

- jira:RemoveActorsFromProjectRole
- jira:RemoveActorsFromDefaultProjectRole
- jira:UpdateProjectRole

jira:AddComment

This function adds a comment to an Issue.

Attributes

Attribute Name	Type	Default Value	Description
issue-key	string		The issue to add the comment to (required).
commenter	string	Currently logged in user	Username of the user to make the comment (Must have browse and comment permissions).
comment	string		Comment to be added to the issue (required).
groupLevel	string	none	Name of group that can see this comment. NOTE: If this is specified you can not specify the roleLevel parameter.
roleLevel	string	none	Name or Id of Project Role that can see this comment. NOTE: If this is specified you can not specify the groupLevel parameter.
created	string	Current Date/Time	Date/Time the Comment was created in format yyyy-MM-dd hh:mm:ss.0
updated	string	Current Date/Time	Date/Time the Comment was last updated in format yyyy-MM-dd hh:mm:ss.0. This can be used if you are trying to import a comment with specific pre-existing values.
editedBy	string	Currently logged in user	Username of the user who last updated the comment. This can be used if you are trying to import a comment with specific pre-existing values.

Examples

Create comment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComment comment="Issue comment" issue-key="ABC-1"
groupLevel="admin-group"/>
</JiraJelly>
```

Create Issue and Comment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreateIssue project-key="TP" issueType="Bug" summary="Issue summary" issueKeyVar="key"/>
  <jira:AddComment issue-key="${key}" comment="A comment on ${key}"/>
</JiraJelly>
```

jira:AddComponent

Adds a component to a project.

Attributes

Attribute Name	Type	Default Value	Description
project-key	string		The key of the project you want to add the component to (not required if nested inside atag).
name	string		Name of the component (required).
description	string		Description of the component.
componentLead	string		For JIRA Enterprise only. The username of the Component's lead. Leave blank for no lead.

Examples

Create Component

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComponent project-key="ABC" name="Comp 1" description="Comp 1 description"/>
</JiraJelly>
```

Create Component in a Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:AddComponent name="Comp 1"/>
</jira:CreateProject>
</JiraJelly>
```

Create Component with a Component Lead (Enterprise-only)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComponent project-key="ABC" name="Comp 1" description="Comp 1 with lead" componentLead=
"user-name"/>
</JiraJelly>
```

jira:SelectComponentAssignees

Selects the default assignees for newly created issues of the component. This tag is only available in JIRA Enterprise.

Attributes

Attribute Name	Type	Default Value	Description
project-key	string		The key of the project you want to add the component to (required).
componentName	string		Name of the component (required).
assigneeType	string		Default assignee type (required). ***Assignee Types:*** • projectDefault • componentLead • projectLead • unassigned

Examples

Select a Component Assignee

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
<jira:SelectComponentAssignees project-key="ABC" componentName="Comp 1" assigneeType="componentLead"/>
</JiraJelly>
```

jira:AddUserToGroup

Makes a user a member of a Group. Adds the username and/or group name into the context if specified.

Attributes

Attribute Name	Type	Default Value	Description
username	string		Username to add to Group (required if not in atag).
group-name	string		Group to add User to (required if not in atag). Note: if the group has the 'JIRA System Administrators' global permission, and the logged-in user does not, an error message will be displayed and the operation will not succeed.

Username is set in the context if specified in the tag.Group name is set in the context if specified in the tag.h4. Examples

Add User to Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddUserToGroup username="new-user" group-name="new-group"/>
</JiraJelly>
```

Add New User to Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com">
<jira:AddUserToGroup group-name="new-group"/>
</jira:CreateUser>
</JiraJelly>
```

Add User to New Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateGroup group-name="new-group">
<jira:AddUserToGroup username="new-user"/>
</jira:CreateGroup>
</JiraJelly>
```

Add New User to New Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com"/>
<jira:CreateGroup group-name="new-group">
<jira:AddUserToGroup/>
</jira:CreateGroup>
</jira:CreateUser>
</JiraJelly>
```

jira:AddVersion

Adds a version to a project.
Attributes

Attribute Name	Type	Default Value	Description
project-key	string		The key of the project you want to add the component too (not required if nested inside atag).
name	string		Name of the version (required).
description	string		The description of the version.
releaseDate	string		The release date of the version.

schedule	string		Schedule of the version.

Examples

Create a Version

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddVersion project-key="ABC" name="Ver 1"/>
</JiraJelly>
```

Create a Version in a Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:AddVersion name="Ver 1"/>
</jira:CreateProject>
</JiraJelly>
```

jira:CreateGroup

Creates a Group in JIRA.

Attributes

Attribute Name	Type	Default Value	Description
group-name	string		Name of group to create (required).

Context Variables

Context Variable	Type	Description
jelly.group.name	string	Name of group being created.

Examples

Create Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateGroup group-name="new-group"/>
</JiraJelly>
```

jira:AssignIssue

Assigns an issue to a user.

Attributes

Attribute Name	Type	Default Value	Description
key	string		Key of the issue to assign.
assignee	string		User to assign issue to.

Examples

Create and assign issue

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreateIssue project-key="TST" summary="My Issue summary" issueKeyVar="keyvar"/>
  <jira:AssignIssue key="${keyvar}" assignee="testuser"/>
</JiraJelly>
```

jira:CreateIssue

This tag creates a new issue in JIRA and places the issue id in the context.

Attributes

Attribute Name	Type	Default Value	Description
project-key	string		Key of the project to add the issue to (required if not nested in atag).
issueType	string	First issue type	The string name of the Issue Type this issue should be created for (e.g. Bug).
summary	string		Summary of the issue being created (required).
priority	string	First priority	The string name of the Priority (e.g. Major).
components	string		The string name of the Component.
versions	string		The string name of the Affected Version.
fixVersions	string		The string name of the Fix For Version.
assignee	string		The username of the user to assign this issue to (logged in user requires the assign issue permission and user specified requires the assignable permission). Set to "-1" for Automatic assignment.
reporter	string	(see description)	The username of the user who is reporting this issue. The user is logged in and then the issue is created. The user is logged out again when the Create Issue tag closes. If the logged in user does not have Modify Reporter privilege, then the default value of this attribute is the username of the logged in user. If, however, the logged in user does have Modify Reporter privilege, there is not a default value, and this attribute is mandatory. See JRA-12984 for further explanation. (Broken? See JRA-5620.)
environment	string		Description of the environment.
description	string		Detailed description of the issue.

duedate	string		Due date of the issue. The format required is the current JIRA date format.
			Note: As the default JIRA date format is locale-specific (e.g. 12/Jan/05), you may wish to use the yyyy-mm-dd ISO format instead. To do this, set the following in WEB-INF/classes/jira-application.properties:
			`jira.date.picker.java.format = yyyy-MM-d`
			`jira.date.picker.javascript.format = %Y-%m-%e`

created	string	Current Date/Time	Date/Time the Issue was created in format yyyy-MM-dd hh:mm:ss.0
updated	string	Current Date/Time	Date/Time the Issue was updated in format yyyy-MM-dd hh:mm:ss.0
issueIdVar	string		The name of the variable to place the ID of the new Issue.
issueKeyVar	string		The name of the variable to place the Key of the new Issue.
duplicateSummary	string		Setting this attribute to 'ignore' will allow Issue with the same summary to be created.
security-level	string		Enterprise-only - sets the security level of an issue. Value is the name of a level, e.g. 'Secret'.

Examples

Create Issue

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="ABC" assignee="-1" summary="Issue summary">
<!-- other jelly tags -->
</jira:CreateIssue>
</JiraJelly>
```

Create Issue from Project
This example is more complicated as a permission scheme is required for the project before an issue can be created.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:CreatePermissionScheme name="admin-scheme">
<jira:AddPermission permissions="Assignable,Browse,Create,Assign"
type="group"/>
<jira:SelectProjectScheme/>
</jira:CreatePermissionScheme>
<jira:CreateIssue summary="Issue summary">
<!-- other jelly tags -->
</jira:CreateIssue>
</jira:CreateProject>
</JiraJelly>
```

Create Issue with Custom Field values
Use the subtag `jira:AddCustomFieldValue`

Attribute Name	Type	Description
id	long	ID of the custom field with the customfield_ prefix
value	string	string representation of the custom field value. Note that this may be different to the displayed value (e.g. The project picker uses the project id as the String value but displays the project name)
key	string	Key is used for multi-dimensional data. Currently, only Cascading selects supports its use. Omit to specify the value of parent, use "1" as the value for child
name	String	**deprecated** Name of the custom field.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="ABC" summary="Issue summary">
<jira:AddCustomFieldValue id="customfield_10000" value="field value"/>
<jira:AddCustomFieldValue name="Environment Select list" value="Windows XP"/>

<!-- For Cascading Selects : Note also that the value for cascading selects is the optionId-->
<jira:AddCustomFieldValue id="customfield_10001" value="Parent Option Id" />
<jira:AddCustomFieldValue id="customfield_10001" value="Child Option Id" key="1" />

<!-- For Version Pickers and Single Version Pickers : Note also that the value for version pickers is
the versionId-->
<jira:AddCustomFieldValue id="customfield_10002" value="Version Id"/>

<!-- For Multi Selects -->
<jira:AddCustomFieldValue id="customfield_10003" value="Value 1" />
<jira:AddCustomFieldValue id="customfield_10003" value="Value 2" />

<!-- For Multi User Pickers : Note also that the value for multi user pickers is a comma separated
list of users-->
<jira:AddCustomFieldValue id="customfield_10004" value="User 1,User 2" />

</jira:CreateIssue>
</JiraJelly>
```

Using the name attribute has been **deprecated.** While it will work in 3.0 its use is discouraged.

jira:LinkIssue

This tag creates a link from one issue to another issue.

Attributes

Attribute Name	Type	Default Value	Description
key	string		The key of the issue to link from (origin of link - required)
linkKey	string		The key of the issue to link to (destination of link - required)
linkDesc	string		linkDesc is taken from the 'Inward Description' or the 'Outward Description' of the link. (required)

Examples

Create a Link between two existing issues

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:LinkIssue key="TST-1" linkKey="TST-2" linkDesc="duplicates"/>
</JiraJelly>
```

Create two issues and link them

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="HSP" assignee="-1" summary="Issue summary 1" reporter="admin"
issueKeyVar="issuekey1"/>
<jira:CreateIssue project-key="NDT" assignee="-1" summary="Issue summary 2" reporter="admin"
issueKeyVar="issuekey2"/>
<jira:LinkIssue key="${issuekey1}" linkKey="${issuekey2}" linkDesc="duplicates"/>
</JiraJelly>
```

jira:TransitionWorkflow

Broken in 3.3 and 3.3.1 — see JRA-7690

This tag executes a workflow transition on an issue.

Please keep in mind that if you are specifying field attribute/value pairs in your Jelly tag then these fields MUST be on the associted workflow transition screen. If the field is not on the screen then the value will not be set on the issue. For example, if you want to set the resolution attribute in your Jelly XML then your transition MUST have a screen associated with it that includes the resolution field on that screen.

Attributes

Attribute Name	Type	Default Value	Description
user	string	Currently logged in user	Username of the user to execute the workflow transition. The user needs to have the adequate permissions to execute the transition. Please note that the permissions required also depend on the fields that are updated during the transition. (See other attributes below).
key	string		The key of the issue to execute the transition on.
workflowAction	string		The id or name of the workflow transition to execute. If the argument can be converted to a number it is assumed to be an id of the transition. Otherwise it is assumed to be a name.

resolution	string		The id or name of the resolution to set on the issue during the transition. Please note that the transition must expect the resolution to be updated, otherwise an error is generated if this attribute is supplied. If the argument can be converted to a number it is assumed to be an id of the resolution. Otherwise it is assumed to be a name.
assignee	string		The username of the user to assign an issue to during the transition. The "user" executing the transition must have permissions to assign issues if this attribute is supplied. Please note that the transition must expect the assignee to be updated, otherwise an error is generated if this attribute is supplied. Use value "-automatic-" to let JIRA assign the issue to the default assignee.
fixVersions	string		A comma separated list of version ids or names to set as "fix for" versions during the transition. The "user" executing the transition must have permissions to set "fix for" versions if this attribute is supplied. Please note that the transition must expect the "fix for" versions to be updated, otherwise an error is generated if this attribute is supplied. If a value in the provided comma separated list can be converted to a number it is assumed to be an id of a version. Otherwise it is assumed to be a name.
comment	string		The comment to add to the issue during the transition. The "user" executing the transition must have permissions to add comments and the transition must be expecting comments to be added during its execution for the comment to be added successfully.
groupLevel	string		The level for the comment. The level must be a name of a group the user is a member of. NOTE: If this is specified you can not specify the roleLevel parameter.
roleLevel	string		Name or Id of Project Role that can see this comment. NOTE: If this is specified you can not specify the groupLevel parameter.

Examples

Execute Workflow Transition

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
                    <jira:TransitionWorkflow key="TST-6" user="testuser" workflowAction="Resolve
issue"
                    resolution="fixed" fixVersions="version 1,version 3" assignee="-automatic-"
                    comment="Test comment" groupLevel="jira-developers" />
</JiraJelly>
```

jira:CreateProject

This tag creates a new project in JIRA and places the project id in the context.

Attributes

Attribute Name	Type	Default Value	Description
key	string		The project key used to create Issue Keys (required).
name	string		The name of the project (required).
lead	string		The username of the user that is the project lead (required).
url	string		The URL of the site for this project.
description	string		The description of this project.

Context Variables

Context Variable	Type	Description
jelly.project.id	string	Id of the Project that was created.
jelly.project.key	string	Key of the Project that was created.

Examples

Create Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="a-user">
<!-- other jelly tags -->
</jira:CreateProject>
</JiraJelly>
```

jira:CreateUser

Creates a user in JIRA and places their username in the context.

Attributes

Attribute Name	Type	Default Value	Description
username	string		Username of the user being created (required).
password	string		User's password. If the password field is left blank, a random password will be auto-generated.
confirm	string		Confirmation of users password (required).
fullname	string		Descriptive name of the user (required).
email	string		Email address of the user (required).
sendEmail	boolean	false	If provided, specifies whether to send a confirmation email.

Context Variables

Context Variable	Type	Description
jelly.new.username	string	Username of the user being created.

Examples

Create User

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com"/>
</JiraJelly>
```

jira:RemoveUser

Removes an existing JIRA user by their username

Attributes

Attribute Name	Type	Default Value	Description
name	string		Username of the user to remove (required).

Examples

Remove User

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:RemoveUser name="existing-user"/>
</JiraJelly>
```

jira:CreatePermissionScheme

Creates a Permission Scheme

Attributes

Attribute Name	Type	Default Value	Description
name	required string		Name of the permission scheme.
description	string		Permission scheme description.

Context Variables

Context Variable	Type	Description
jelly.permission.scheme.id	string	Id of the created permission scheme

jira:AddPermission

Grants permissions within a permission scheme. Often nested within a JIRADOC:CreatePermissionScheme tag.

Attributes

Attribute Name	Type	Default Value	Description
schemeId	string		If not nested in a CreatePermissionScheme tag, specifies the scheme Id to add the permission to (0 is the default permission scheme).
permissions	required string		A comma-separated list of permissions to grant: **String** — **Permission** • Project — Administer projects • Browse — Browse projects • Create — Create issues • Edit — Edit issues • ScheduleIssue — Schedule issues • Move — Move issues • Assign — Assign issues • Assignable — Assignable user • Resolve — Resolve issues • Close — Close issues • ModifyReporter — Modify reporter • Comment — Add comments • CommentEditAll — Edit all comments • CommentEditOwn — Edit own comments • CommentDeleteAll — Delete all comments • CommentDeleteOwn — Delete own comments • Delete — Delete issues • Work — Work on issues • WorklogEditAll — Edit all worklogs • WorklogEditOwn — Edit own worklogs • WorklogDeleteOwn — Delete own worklogs • WorklogDeleteAll — Delete all worklogs • Link — Link issues • Attach — Create attachments • AttachDeleteAll — Delete all attachments • AttachDeleteOwn — Delete own attachments • ViewVersionControl — View version control • ViewVotersAndWatchers — View voters and watchers • ManageWatcherList — Manage watcher list • SetSecurity — Set issue security level

type	string		Type of recipient for the permission: • group • projectrole • user • lead • assignee • reporter • userCF • groupCF
group	string		If type is 'group' (or type is unspecified), specifies the group name to grant permissions to.
projectroleid	int		If type is 'projectrole', specifies the id of the projectrole to grant permissions to.
user	string		If type is 'user', specifies the user name to grant permissions to.
userCF	string		If type is 'userCF', specifies the id of a User custom field, e.g. 'customfield_10000', identifying the user to be granted the permission.
groupCF	string		If type is 'groupCF', specifies the id of a group-selecting custom field (e.g. a select-list with group names as values) whose members should be granted this permission. E.g. 'customfield_10000'.

Examples

Grant permissions to jira-users and jira-developers in a new permission scheme
(See also the JIRADOC:example scripts)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreatePermissionScheme name="New Permission Scheme">
    <jira:AddPermission group="jira-users" permissions="Browse,Create,Comment,Attach" type="group"/>
    <jira:AddPermission group="jira-developers" permissions="Move,Assignable,Link,ViewVersionControl"
type="group"/>
  </jira:CreatePermissionScheme>
</JiraJelly>
```

*Grant issue reporters the ability to edit/delete their own issues, in a new permission scheme
(Enterprise-only)*

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
  <jira:CreatePermissionScheme name="New Permission Scheme">
    <jira:AddPermission type="reporter" permissions="Delete, Edit"/>
  </jira:CreatePermissionScheme>
</JiraJelly>
```

Make projects using default permission scheme visible to certain users (Enterprise-only)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
  <jira:AddPermission schemeId="0" permissions="Browse" type="user" user="johnc"/>
  <jira:AddPermission schemeId="0" permissions="Browse" type="user" user="ebf"/>
</JiraJelly>
```

Granting a group selector custom field's members the ability to assign/be assigned the issue.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:AddPermission schemeId="10164" type="groupCF" groupCF="customfield_10000"
                      permissions="Assign,Assignable" />
</JiraJelly>
```

jira:Login

This tag logs a user into JIRA using the username and password provided. Use this tag when you are running the Jelly script in a manner in which you are not logged in (for example, if you are running a JellyService instead of using the Jelly Runner), or if you want to run the Jelly script as a different user to the one you are logged in as.

Attributes

Attribute Name	Type	Default Value	Description
username	string		Username of the user to log in.
password	string		Password of the user to log in.

Context Variables

Context Variable	Type	Description
jelly.user	User	User logged in.
jelly.username	string	Username of the User logged in.

Examples

Login a user in with username and password and set in context

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:Login username="misc-user" password="password">
<!-- other jelly tags -->
</jira:Login>
</JiraJelly>
```

jira:CreateCustomField

The tag creates a new Custom Field. Only System custom fields can be added with Jelly tags.

Attributes

Attribute Name	Type	Default Value	Description
fieldType	string		Field type as appears as the key in the plugin descriptor
fieldScope	string		One of global, project or issuetype
fieldName	string		Name of custom field
projectKey	string		Key of the related project. Only valid for scope "project"
issueType	string		Issue type. Only valid for scope "issuetype"
description	string		Description of the field to be displayed when adding a value
searcher	string		A valid related custom field searcher
customFieldIdVar	string		The name of the variable to place the new custom field.

Examples

Create Cascading Custom Field
`jira:AddCustomFieldSelectValue` subtag can be used to add values for select lists. They can also be nested for Cascading Select Lists.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateCustomField fieldType="cascadingselect"
fieldScope="issuetype"
fieldName="Issue cascadingselect Bug"
issueType="Bug"
description="Bank have requested Y2K fixes to be sent as an EBF."
searcher="cascadingselectsearcher"
>
<jira:AddCustomFieldSelectValue value="Parent 1" />
<jira:AddCustomFieldSelectValue value="Parent 2">
<jira:AddCustomFieldSelectValue value="Child 1" />
<jira:AddCustomFieldSelectValue value="Child 2" />
</jira:AddCustomFieldSelectValue>
<jira:AddCustomFieldSelectValue value="Parent 3" />
</jira:CreateCustomField>
</JiraJelly>
```

jira:AddFieldToScreen

Adds a field to a specific tab on a screen. Can also specify in which position to insert the field.

Attributes

Attribute Name	Type	Default Value	Description
ffldId	string		Field ID of the field to add (required). e.g. "description", "duedate", etc.
screen	string		Screen ID or Name (required). e.g. "1" or "Default Screen".
tab	string	0	Tab ID or Name. e.g. "0" or "Field Tab".
fieldPosition	int	last position	Position to insert the field into. Range of values is from 1 to the number of fields on the screen.

Examples

Add Fields to a Screen

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<!-- Adds 'description' field to the 'Field Tab' on 'Default Screen' -->
<jira:AddFieldToScreen fieldId="description" screen="Default Screen" tab="Field Tab"/>

<!-- Adds 'duedate' field to same screen as above. duedate is inserted in position 1 -->
<jira:AddFieldToScreen fieldId="duedate" screen="1" tab="0" fieldPosition="1"/>
</JiraJelly>
```

Create a new Customfield and Add it to a Screen

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateCustomField fieldType="cascadingselect"
fieldScope="issuetype"
fieldName="Issue cascadingselect Bug"
issueType="Bug"
description="Bank have requested Y2K fixes to be sent as an EBF."
searcher="cascadingselectsearcher"
customFieldIdVar="customField"
>
<jira:AddCustomFieldSelectValue value="Parent 1" />
<jira:AddCustomFieldSelectValue value="Parent 2">
<jira:AddCustomFieldSelectValue value="Child 1" />
<jira:AddCustomFieldSelectValue value="Child 2" />
</jira:AddCustomFieldSelectValue>
<jira:AddCustomFieldSelectValue value="Parent 3" />
</jira:CreateCustomField>

<jira:AddFieldToScreen screen="Default Screen" fieldId="${customField.getId()}"/>
</JiraJelly>
```

jira:AttachFile

Attaches a file to an issue.

Attributes

Attribute Name	Type	Default Value	Description
key	string		Key of the issue to attach the file to. (Required)
filepath	string		Path (on the server) of the file to attach. (Required)
option	string	add	Behaviour when a file with same name is already attached. (Optional). The options are: • **skip** — do not attach file if a file with this name is already attached. • **override** — overwrite existing attached file • **add** — add the file as another attachment
created	string	Current Date/Time	Date/Time the attachment was created, in format yyyy-MM-dd hh:mm:ss.0 (Optional)

Examples

Adding an attachment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:AttachFile key="TST-1" filepath="/tmp/somefile" option="override"/>
</JiraJelly>
```

jira:RunSearchRequest

This tag runs a search request against JIRA using a predefined filter.
Note: This tag will return a GenericValue for each issue which matches the search request.
A GenericValue consists of key-value pairs, e.g.

```
[GenericEntity:Issue]
[created,2007-11-01 15:51:25.0]
[summary,Testing]
[component,null]
[workflowId,12530]
[timeoriginalestimate,null]
[fixfor,null]
[type,2]
[timespent,null]
[environment,Windows]
[resolution,null]
[status,1]
[updated,2007-11-01 15:51:25.0]
[timeestimate,null]
[id,11540]
[key,TSTA-5]
[duedate,null]
[description,Test]
[project,10063]
[reporter,admin]
[security,null]
[votes,0]
[assignee,null]
[priority,3]
```

To retrieve a value, e.g. key, you can call gv.getString("key"). For full details, see the OFBiz GenericValue API.

Attributes

Attribute Name	Type	Default Value	Description
filterid	int		The id of the filter which will be used to run the search request.
size-var	string		The variable that will hold the number of issues returned from the search request.
var	string		The variable that will hold the issues returned from the search request.

Examples

Running a search request and iterating through the keys of the returned issues

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RunSearchRequest filterid="10524" var="issues" size-var="issuecount"/>
    <core:forEach var="issue" items="${issues}">
        ${issue.key}
    </core:forEach>
</JiraJelly>
```

jira:AddActorsToDefaultProjectRole

This tag will add 'actors' to the default membership for a given project role. Actors can be defined as groups or users, i.e. you can add both users and groups to a project role.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		This is the id of the project role.
actors	string		A comma delimited list of either users or groups

actortype	string		This defines the type 'actor' you are sending to the tag. Currently this field can contain either 'atlassian-user-role-actor' for users, or 'atlassian-group-role-actor' for groups.

Examples

Adding a list of default users or groups to a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:AddActorsToDefaultProjectRole projectroleid="1" actors="fred,admin,tom"
        actortype="atlassian-user-role-actor" />
</JiraJelly>
```

jira:AddActorsToProjectRole

This tag will add 'actors' to a given project role for a particular project. Actors can be defined as groups or users, ie you can add both users and groups to a project role.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		This is the id of the project role.
actors	string		This a comma delimited list of either user names or group names
actortype	string		This defines the 'actor' type. Currently this field can contain either 'atlassian-user-role-actor' for users, or 'atlassian-group-role-actor' for groups.
projectkey	string		This is the key of the project you wish to add users or groups to for the specified role.

Examples

Adding a list of users or groups to a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:AddActorsToProjectRole projectroleid="1" actors="jira-administrators,jira-users"
        projectkey="MKY" actortype="atlassian-group-role-actor" />
</JiraJelly>
```

jira:CreateProjectRole

This tag will create a project role with the given name and description.

Attributes

Attribute Name	Type	Default Value	Description
name	string		The name for the project role you will be creating
description	string		The description for the project role you will be creating

Context Variables

Context Variable	Type	Description
jelly.role.id	Long	The id of the project role
jelly.role.name	string	The name of the project role
jelly.role.description	string	The description of the project role

Examples

Creating a new project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:CreateProjectRole name="lion-tamer" description="tames the lions">
        ${jelly.role.id} ${jelly.role.name} ${jelly.role.description}
    </jira:CreateProjectRole>
</JiraJelly>
```

jira:DeleteProjectRole

This tag will delete the project role with the given id.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you want to delete.
confirm	string		To delete the project role this value must be set to 'true'.

Examples

Deleting a project role from JIRA

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:DeleteProjectRole projectroleid="1" confirm="true" />
</JiraJelly>
```

jira:GetDefaultRoleActors

This tag will return a ProjectRoleActors object for a given project role for a particular project. This object carries the members of a project role, i.e. users and/or groups. To get the collection of users in this object, use the expression ${roleactors.users} where roleactors is the variable name of the object. For more information on the RoleActors object, consult the JIRA API.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you want to query
var	string		The name of the variable you wish to have the returned role actors placed into

Examples

Returning a List of role actors and iterating over the users in each of these actors.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:core="jelly:core">
    <jira:GetDefaultRoleActors projectroleid="1" var="roleactors" >
        <core:forEach var="actor" items="${roleactors.users}">
            ${actor.name}
        </core:forEach>
    </jira:GetDefaultRoleActors>
</JiraJelly>
```

jira:GetProjectRole

This tag will return the project role with the given id.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you want
var	string		The name of the variable you wish to have the project role assigned to

Examples

Returning a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:GetProjectRole projectroleid="1" var="role" >
            ${role.name}
    </jira:GetProjectRole>
</JiraJelly>
```

jira:GetProjectRoleActors

This tag will return a ProjectRoleActors object for the given project role and project. This object is a placeholder for the internal members of a project role, i.e. users and/or groups. To get the collection of users in this object, use the expression ${roleactors.users} where roleactors is the variable name of the object. For more information on the RoleActors object, consult the JIRA API.

Attributes

Attribute Name	Type	Default Value	Description
projectkey	string		The key of the project you want to query
projectroleid	int		The id of the project role you want to query
var	string		The name of the variable you want the returned 'role actors' object assigned to

Examples

Return a list of users for a given 'Role Actors' object

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:core="jelly:core">
    <jira:GetProjectRoleActors projectkey="MKY" projectroleid="1" var="roleactors" >
        <core:forEach var="actor" items="${roleactors.users}">
            ${actor.name}
        </core:forEach>
    </jira:GetProjectRoleActors>
</JiraJelly>
```

jira:IsProjectRoleNameUnique

This tag will return 'true' or 'false' to let you know if there is already a project role with the given name.

Attributes

Attribute Name	Type	Default Value	Description
name	string		The name of the project role
var	string		The name of the variable you want the returned result assigned to.

Examples

Determining if a project role is unique.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:IsProjectRoleNameUnique name="unique name" var="isUnique" >
        ${isUnique}
    </jira:IsProjectRoleNameUnique>
</JiraJelly>
```

jira:RemoveActorsFromProjectRole

This tag will remove a list of role actors from a given project role for a given project.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you wish to remove members from
actors	string		A comma delimited list of users or groups you wish to remove from the project role
projectkey	string		The key of the project the project role is associated with
actortype	string		The type of 'actor' you are working with. Currently the available options are 'atlassian-group-role-actor' or 'atlassian-user-role-actor'

Examples

Removing a list of groups from a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RemoveActorsFromProjectRole projectroleid="1"
        actors="jira-administrators, jira-users" projectkey="MKY"
        actortype="atlassian-group-role-actor" />
</JiraJelly>
```

jira:RemoveActorsFromDefaultProjectRole

This tag will remove a list of role actors (i.e. users and/or groups) from the default membership of a given project role.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you wish to remove default actors from
actors	string		A comma delimited list of users or groups you wish to remove from the default project role
actortype	string		The type of 'actor' you are removing. Currently the available options are 'atlassian-group-role-actor' or 'atlassian-user-role-actor'

Examples

Removing a list of groups from a default project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RemoveActorsFromDefaultProjectRole projectroleid="1"
        actors="jira-administrators, jira-users" actortype="atlassian-group-role-actor" />
</JiraJelly>
```

jira:UpdateProjectRole

This tag will update the name and description for a given project role id.

Attributes

Attribute Name	Type	Default Value	Description
projectroleid	int		The id of the project role you want to query
name	string		The name you want the project role updated with
description	string		The description you want the project role updated with

Examples

Updating a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:UpdateProjectRole projectroleid="123" name="unique name"
        description="my project role is nice" />
</JiraJelly>
```

Beta Tags

There are also a number of BETA tags that have not been fully tested or documented. The following list contains the tags and the attributes that can be passed to them:

- AddIssueSecurity
  - schemeId (required)
  - security (required)
  - type (required)
- AddIssueSecurityLevel
  - name (required)
  - description (required)
  - Output
    - jelly.issue.scheme.level.id
- CreateIssueSecurityScheme
  - name (required)
  - description (required)
  - Output
    - jelly.issue.scheme.id
- LoadManager
  - var (variable to put manager in)
  - manager (name of manager e.g. IssueManager)
- LoadProject
  - var (variable to put project in)
  - project-name (name of project)
- RemoveGroup
  - name (required)
- RemovePermissionScheme
  - schemeId (required)
  - confirm (required))
- RemoveProject
  - pId (required)
- SelectProjectScheme
  - projectKey (required)
  - permission-scheme (Name of permission scheme)or
  - issue-scheme (Name of issue security scheme)

- StringContains
  - value (String to look in)
  - possiblyContains (String to look for)
  - doesContain (true or false) if value contains possiblyContains == doesContain, the inside of the tag is executed.

If you would like more information on how to use the Beta tags, please read the source and/or post to the JIRA Development Forum.

Sample scripts

Creating a new Project

To properly partition projects, one needs a permission scheme per project, and project-specific groups to allocate permissions to. Setting up a new project can be a time-intensive process. The following sample Jelly scripts automate this:
This script might be used for a publicly visible project:

```
<?xml version="1.0"?>
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:j="jelly:core">

  <j:set var="name" value="Test Project"/>
  <j:set var="key" value="TEST"/>
  <j:set var="lowerkey" value="test"/>
  <j:set var="lead_username" value="joe"/>
  <j:set var="lead_password" value="joe"/>
  <j:set var="lead_fullname" value="Joe Bloggs"/>
  <j:set var="lead_email" value="joe@example.com"/>
  <j:set var="url" value="http://example.com/TestProj"/>

  <jira:CreateUser username="${lead}" password="${lead}" confirm="${lead}"
    fullname="${lead_fullname}" email="${lead_email}"/>
  <jira:CreateGroup group-name="${lowerkey}-developers">
    <jira:AddUserToGroup username="${lead}"/>
  </jira:CreateGroup>

  <jira:CreateProject key="${key}" name="${name}" url="${url}" lead="${lead}">
    <jira:CreatePermissionScheme name="${name} permissions">
      <jira:AddPermission type="reporter" permissions="Close"/>
      <jira:AddPermission group="jira-administrators" permissions="Close,Delete" type="group"/>
      <jira:AddPermission group="jira-users" permissions="Create,Edit,Comment,Link,Attach" type=
"group"/>
      <jira:AddPermission group="${lowerkey}-developers"
        permissions="Project,ScheduleIssue,Move,Assign,Assignable,Resolve,Close,Work" type="group"/>
      <jira:AddPermission group="Anyone" permissions="Browse,ViewVersionControl"/>
      <jira:SelectProjectScheme/>
    </jira:CreatePermissionScheme>
  </jira:CreateProject>
</JiraJelly>
```

This script is more complicated, with multiple groups per project:

```
<?xml version="1.0"?>
<!-- This script handles some of the administrative chores required when adding
a new project to JIRA. It creates the project, groups, permission scheme, and gives
groups the relevant permissions in the permission scheme. -->
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:j="jelly:core">

<!-- Name of the project to create -->
<j:set var="name" value="Jelly Test Project"/>
<!-- Key for the new project -->
<j:set var="key" value="TEST"/>
<!-- Existing user who will become the project lead (default assignee) -->
<j:set var="admin" value="admin"/>

<jira:CreateGroup group-name="${key}-users"/>
<jira:CreateGroup group-name="${key}-developers"/>
<jira:CreateGroup group-name="${key}-managers"/>
<jira:CreateGroup group-name="${key}-bizusers"/>
<jira:CreateGroup group-name="${key}-qa"/>

<jira:CreateProject key="${key}" name="${name}" lead="${admin}">
<jira:CreatePermissionScheme name="${key} Permission Scheme">
<jira:AddPermission type="reporter" permissions="Edit"/>
<jira:AddPermission type="assignee" permissions="Resolve"/>
<jira:AddPermission group="jira-administrators" permissions="Project,Delete" type="group"/>
<jira:AddPermission group="${key}-users" permissions="Browse,Create,Comment,Attach" type="group"/>
<jira:AddPermission group="${key}-developers" permissions="Move,Assignable,Link,ViewVersionControl"
    type="group"/>
<jira:AddPermission group="${key}-managers" permissions="Edit,Assign,Assignable,Resolve,Close,Delete"
    type="group"/>
<jira:AddPermission group="${key}-bizusers" permissions="Assignable" type="group"/>
<jira:AddPermission group="${key}-qa" permissions="Assignable" type="group"/>
<jira:AddPermission group="opsmgrs" permissions="Browse,Edit,Assignable,Comment" type="group"/>
<jira:AddPermission group="dba-user-group" permissions="Browse,Assign,Assignable,Comment" type="group"
/>
<jira:AddPermission group="help-desk-group" permissions="Browse,Assign,Assignable,Comment" type=
"group"/>
<jira:AddPermission group="webadmin-group" permissions="Browse,Assign,Assignable,Comment" type="group"
/>
<jira:AddPermission group="unix-admin-group" permissions="Browse,Assign,Assignable,Comment" type=
"group"/>
<jira:SelectProjectScheme/>
</jira:CreatePermissionScheme>
</jira:CreateProject>
</JiraJelly>
```

For a list of projects, perform a project-specific operation.

This script iterates through a (comma-separated) list of projects, creates a project-specific group, and adds a user to that group.

```
<?xml version="1.0"?>
<!-- Jelly script to create 'support' group per project -->
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:util="jelly:util" xmlns:j=
"jelly:core">
<util:tokenize var="projects" delim=","
>ARM,QWI,DWI,DBOR,DBSQ,LYX,MMM,MOI,TPAI,SEP,AMR,SLA,TP,TRBC,YRD</util:tokenize>
<j:forEach var="proj" items="${projects}">
<jira:CreateGroup group-name="${proj}-support"/>
<jira:AddUserToGroup username="jeff" group-name="${proj}-support"/>
</j:forEach>
</JiraJelly>
```

Create a user, issue, and assign the issue to the user

The following script creates a user (called `new-user`), creates a new issue, adds the user to the `jira-developers` group and assigns the issue to the user. It illustrates the use of context variables.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">

<jira:CreateUser username="new-user" password="password" confirm="password"
    fullname="Full name" email="test@test.com"/>
Username is ${jelly.new.username}
 <jira:CreateIssue project-key="TP" summary="New issue summary" issueKeyVar="ik"/>
 <jira:AddUserToGroup username="new-user" group-name="jira-developers"/>
 <jira:AssignIssue key="${ik}" assignee="${jelly.new.username}"/>

</JiraJelly>
```

Assigning and Starting Progress

Here we create an issue, assign it to 'bob' (who must be in `jira-developers`), and start progress:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">

 <jira:CreateIssue project-key="TP" summary="New issue" issueKeyVar="ik"/>
 <jira:AssignIssue key="${ik}" assignee="bob"/>
 <jira:TransitionWorkflow key="${ik}" user="bob" workflowAction="Start Progress" />
</JiraJelly>
```

Moving unreplied-to issues into an 'Inactive' state

When JIRA is used for interacting with customers, this script is useful for finding issues which are awaiting customer response, and haven't been responded to in a while. It moves such issues into an 'Inactive' state.

You would typically invoke this script periodically with the Jelly Service.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:core="jelly:core"
xmlns:log="jelly:log" >
<jira:Login username="customersupport" password="XXXXXX">
        <log:warn>Running Inactivate issues service</log:warn>
        <core:set var="comment">This issue has not been updated for 5 business days.

If you have an update, please use "Add Comments For Vendor" action to let us know.
If you need more time to gather information please let us know and we will 'freeze' this issue.
If you have no other questions, please Close this issue.

If no update is received in the next 5 business days, this issue will be automatically closed.

Thank you,

  The Support Team</core:set>
        <core:set var="workflowStep" value="Mark Inactive" />
        <core:set var="workflowUser" value="customersupport" />

        <!-- Run the SearchRequestFilter -->
        <jira:RunSearchRequest filterid="11505" var="issues" />

        <core:forEach var="issue" items="${issues}">
                <log:warn>Inactivating issue ${issue.key}</log:warn>
                <jira:TransitionWorkflow key="${issue.key}" user="${workflowUser}" workflowAction=
"${workflowStep}" comment="${comment}"/>
        </core:forEach>
</jira:Login>
</JiraJelly>
```

Where:

- **workflowStep** is the name of a workflow transition, e.g "Close Issue", "Start Progress", just as they appear in the left-hand menu on the issue screen.
- **workflowUser** is the user to run the transition as
- **filterid** is the id of a saved search (filter), which finds issues needing to be inactivated (transitioned). This ID can be discovered from the

filter URL on the "Manage" tab in "Find issues".

The JIRA Toolkit is useful in conjuction with this script, to find issues awaiting customer response.

JIRA Toolkit (Customer Support Extensions)

As an extension to JIRA, Atlassian have developed a set of JIRA custom fields, collectively called the "JIRA Toolkit". It can be found online at:

http://confluence.atlassian.com/display/JIRAEXT/JIRA+Toolkit

These custom fields are particularly useful in customer-facing JIRA instances. They were initially developed for use in Atlassian's own JIRA Support installation at http://support.atlassian.com. See the JIRA Toolkit documentation for details.

Developer Guides

The following Tutorials can be found in the JIRA Development Hub:

- Jelly Examples
- JIRA Custom Installer Guide
- JIRA Plugin Tutorials
    - Available Permissions
    - How to create a JIRA Portlet
    - How to create a JIRA Report
    - How to create a new Custom Field Type
    - How to create Custom Workflow Elements for JIRA 3
- JIRA RPC Services
    - Creating a SOAP Client
    - Creating a XML-RPC Client
    - Enabling the RPC plugin
    - JIRA XML-RPC Overview
    - Frequently Asked RPC Questions and Known Issues
- Modifying JIRA Templates and JSPs
    - Adding Custom Fields to Email
    - Adding Custom Field to Issue Summary
    - Adding JavaScript to all pages E.g. Google Analytics
    - Contents of the Velocity Context
    - Customising interface based on user's role
    - Customising JIRA Excel Output
    - Customizing text
    - Mapping custom events to new email templates
    - Tutorial - How to Create a Custom Release Notes Template Containing Release Comments
    - Velocity Context for Email Templates
- Services, Listeners and Handlers
    - Adding your own email handling classes
- 3.6 Technical Improvements
    - Programmatically determining automatic assignee
    - Resolutions per workflow step
    - Subtask creation form enhancements

Building JIRA from Source

Commercial users receive access to JIRA source. This documentation shows how to build the JIRA source back into an application that can be deployed.

You would only be interested in this documentation if you are making modifications to the JIRA source code. Changes to JSP files do not require rebuilding JIRA. Also, you should be aware of the possibilities the plugin system affords — often changes can be developed and packaged as a plugin without requiring core source modifications.

- Building JIRA WAR from JIRA Source release
- Developing using Intellij IDEA
- Building the Atlassian source dependencies
- Obtaining the source of JIRA's dependencies
- Compiling classes into JIRA Standalone

Building JIRA WAR from JIRA Source release

1. Ensure you have JDK 1.4 or higher.
2. Download Maven 1.0.x from http://maven.apache.org
3. Extract Maven somewhere, say c:\Dev\testing
4. Set MAVEN_HOME:

```
> set MAVEN_HOME=c:\Dev\testing\maven-1.0
```

5. Download Maven 2.0.x from http://maven.apache.org
6. Follow the general instructions for setting up Maven 2.0.x
7. Download JIRA Source zip from http://www.atlassian.com/software/jira/JIRASourceDownloads.jspa. You will need to log in as a user with a commercial licence to access this page.
8. Extract the JIRA Source zip somewhere, say c:\Dev\testing.
9. Your c:\Dev\testing should look somewhat like:

```
C:\Dev\testing>dir

Volume in drive C is COOKIE
Volume Serial Number is 3F3F-14F0

Directory of C:\Dev\testing

31/01/2009  04:30p      <DIR>          .
31/01/2009  04:30p      <DIR>          ..
31/01/2009  04:18p      <DIR>          atlassian-cache-servlet
31/01/2009  04:18p      <DIR>          atlassian-core
31/01/2009  04:18p      <DIR>          atlassian-gzipfilter
31/01/2009  04:18p      <DIR>          atlassian-jdk-utilities
31/01/2009  04:18p      <DIR>          atlassian-ofbiz
31/01/2009  04:18p      <DIR>          atlassian-profiling
31/01/2009  04:18p      <DIR>          atlassian-renderer
31/01/2009  04:18p      <DIR>          atlassian-velocity
31/01/2009  04:18p      <DIR>          bandana
31/01/2009  04:18p      <DIR>          bonnie
31/01/2009  04:18p      <DIR>          configurableobjects
31/01/2009  04:18p      <DIR>          jira
31/01/2009  04:18p      <DIR>          jira-bamboo-plugin-v2
31/01/2009  04:18p      <DIR>          jira-fisheye-plugin
31/01/2009  04:18p      <DIR>          johnson
31/01/2009  04:18p      <DIR>          mail
31/01/2009  04:18p      <DIR>          plugins
31/01/2009  04:18p      <DIR>          rpc-jira-plugin
31/01/2009  04:18p      <DIR>          scheduler
31/01/2009  04:18p      <DIR>          seraph
31/01/2009  04:18p      <DIR>          trackback

0 File(s)              0 bytes
21 Dir(s)  16,352,509,952 bytes free
```

10. Change into the jira\ subdirectory, and build using Maven by executing the following command:

```
C:\Dev\testing\jira> maven war:webapp
```

If you would like to build a closed WAR file, then do not use the command displayed above. You will need to run the following maven command instead:

```
maven -Djira.build.bundle.plugins=false include-rpc-plugin war:war
```

The **-Djira.build.bundle.plugins=false include-rpc-plugin** part prevents JIRA trying to build the Fisheye plugin, which was bundled with 3.12, but which is not buildable from the JIRA source distribution. It is not required in earlier or later releases.

If you are attempting to build JIRA 3.13, you will need to make changes to the **build.properties** file before running your build, as the maven repository information is incorrect. Hence, your build will not be able to find dependent JARs, such as atlassian-mail. See JRA-15648 for detailed instructions. Please note, this issue only affects JIRA 3.13, it does not apply to JIRA 3.13.x.

11. Confirm that the open .war has been created in .\target\atlassian-jira

```
C:\Dev\testing\jira\target\atlassian-jira>dir
Volume in drive C is COOKIE
Volume Serial Number is 3F3F-14F0

Directory of C:\Dev\testing\jira\target\atlassian-jira

24/02/2003  04:41p      <DIR>          .
24/02/2003  04:41p      <DIR>          ..
24/02/2003  04:41p      <DIR>          decorators
24/02/2003  04:41p      <DIR>          images
24/02/2003  04:41p      <DIR>          includes
24/02/2003  04:41p      <DIR>          portlets
24/02/2003  04:41p      <DIR>          secure
24/02/2003  04:41p      <DIR>          styles
24/02/2003  04:41p      <DIR>          template
24/02/2003  04:41p      <DIR>          views
24/02/2003  04:41p      <DIR>          WEB-INF
24/02/2003  04:41p                 8781 500page.jsp
24/02/2003  04:41p                 1593 bugzillasearch.jsp
24/02/2003  04:41p                 328 default.jsp
24/02/2003  04:41p                 894 favicon.ico
24/02/2003  04:41p                 211 login-error.jsp
24/02/2003  04:41p                 203 login.jsp
24/02/2003  04:41p                 733 logoutconfirm.jsp
24/02/2003  04:41p                 939 logout.jsp
8 File(s)  13,682
11 Dir(s)  56931786752 bytes free
```

You should now be able to point a suitably configured Servlet 2.3+ compliant app server at this directory, and run JIRA.

Developing using Intellij IDEA

If you are an IDEA user, you may wish to use the atlassian-idea plugin we have developed to quickly generate a work environment.

Building the Atlassian source dependencies

JIRA's source distribution not only ships with JIRA's source code, it also includes the source of the internal Atlassian projects that JIRA depends on (e.g. atlassian-bonnie, atlassian-core, etc.). These dependencies are included in JIRA in binary format when you build the JIRA source (they are downloaded from the Atlassian maven repository).

You can, however, compile the provided source to generate the binaries yourself. These projects use a mix of Maven 1 and Maven 2 build systems to compile and package their source. You can tell a project uses Maven 1 if the project contains a file called 'project.xml' in the top level directory. If a project uses Maven 2, it will contain a file called 'pom.xml' in the top level directory.

Building a Maven 1 project you will invoke 'maven jar', whereas for a Maven 2 project you will invoke 'mvn package'. In order to run the 'mvn' command you will have to install Maven 2. Please follow the general instructions regarding setting up a development environment. Please note that you will also have to add the Atlassian Maven 2 repository to your Maven 2 configuration. To do this you will need to edit your settings.xml as described here.

Obtaining the source of JIRA's dependencies

Most of JIRA's dependencies are either shipped in binary (compiled) form with the source distribution, or are available on Maven's public repository. Maven will fetch the dependencies that it requires automatically during the build process, so you do not have to do it manually. Hence, you do not need the source of every dependency to build JIRA from source. However, sometimes you might want to "look inside" these dependencies. If so, this section is for you.

The source distribution of JIRA is shipped with a `project.xml` file. All of JIRA's dependencies are listed inside this file. Most of the dependencies are open source libraries but some are Atlassian's code. All of the Atlassian code is included in the source distribution. The source

of the other dependencies is usually available on the library's website (try googling for the library name).

In some cases JIRA uses unofficial 'snapshot' releases of a library, sometimes additionally patched to fix bugs or add features. In these cases the library source can be obtained from Atlassian's repository, at http://repository.atlassian.com/ **dependencyId** /distributions/, where **dependencyId** is the dependency name found in the `project.xml` record.

For example, source for the dependency:

```
<dependency>
      <id>javacvs</id>
      <version>20050531-patched</version>
      <properties>
        <war.bundle>true</war.bundle>
      </properties>
    </dependency>
```

can be found at http://repository.atlassian.com/javacvs/distributions/javacvs-20050531-patched-src.tar.gz. If source modifications were made, a patch is usually available at http://repository.atlassian.com/ **dependencyId** /patches/

If you have any questions regarding the build process, please post to the JIRA Development Forum, which is monitored continually by the development community, and by Atlassian as often as possible.

Compiling classes into JIRA Standalone

If you just want to compile one class (perhaps a service) and you're using the JIRA Standalone distribution, there is an Ant-based mini-build system available in the `external-source` directory. See JIRA Standalone quick source modifications for details.

API Documentation

The JIRA API docs are available online. They are most useful for:

- users writing Plugins, Listeners and Services
- users with commercial licenses who wish to modify JIRA
- partners embedding JIRA as a J2EE component

The latest API docs are available at http://docs.atlassian.com/software/jira/docs/api/latest/. The 3.13 docs are available at http://docs.atlassian.com/software/jira/docs/api/3.13. For previous versions, substitute the appropriate version in the URL.

JIRA Administrators FAQ

JIRA Administrators FAQ

Installation Notes

- Application Server Notes — These pages contain notes on deploying JIRA in various application servers.
    - Geronimo Notes
    - JBoss 4 notes
    - Jetty Notes
    - JOnAS Notes
    - Oracle Weblogic Notes
    - Resin Notes
    - Sun Java App Server (SJAS or Glassfish) Notes
    - Tomcat 5.5 notes
    - Tomcat 6.0 notes
    - Websphere 5.1 Notes
    - Websphere 6.x Notes
- Database Notes — These pages contain notes on configuring JIRA with various databases.
    - JIRA and DB2
    - JIRA and Firebird
        - Modifications to entitymodel.xml for Firebird
    - JIRA and HSQL
        - Running SQL commands in a HSQL database — On rare occasions, one may wish to run raw SQL queries on a JIRA or Confluence database. This page describes how to obtain a SQL console for hsqldb databases, which JIRA and Confluence Standalone editions use by default.
    - JIRA and MS SQL Server
        - Connecting to named instances in SQL Server
        - Error caused by SET NOCOUNT in MS SQL Server
        - MS SQL Server 2000 Startup errors
        - Setting up your JIRA database for MS SQL Server 2005
    - JIRA and MYSQL
        - Configuring MySQL 4.1 to store non-ASCII characters
        - Data Access Exception - Errcode - 17
        - JIRA can't connect to MYSQL with Named Pipes enabled
        - MYSQL Administrator and Data Truncation Errors
        - Setting up JIRA Standalone and MySQL on Linux
    - JIRA and Oracle
        - Configuring Datasource for Oracle 10g JDBC drivers
        - Restoring data using I-Net (Oranxo) Driver for Oracle
        - Store Workflow on Disk with Oracle 8 — A workaround for the problem of > 4000 character workflows in Oracle 8 is to store these on disk, instead of in the database.
    - JIRA and PostgreSQL
        - Setting up JIRA Standalone and PostgreSQL on Linux
    - JIRA and Sybase
    - Connecting JIRA to a different database than the one provided as default
    - Database Disappears
    - Incorrect database type specified
    - Making databases read-only for upgrade backups
    - Restarting from Setup Wizard in JIRA Standalone
    - Surviving Connection Closures
- Configure JIRA as service on Mac OS X
- Configuring IIS with Tomcat
- Deploying JIRA in a clustered environment
- Installation Troubleshooting Guide
- Installing a LDAP server on Debian Linux for use with JIRA
- Installing Java on Ubuntu or Debian
- java.lang.NoClassDefFoundError
- JVM and Appserver configuration info
    - Causes of OutOfMemoryErrors — When memory problems do occur, the following checklist can help you identify the cause.
    - Issues with tomcat 4.x mail and activation jars
    - Java crashes
    - Setting up a database connection in Websphere 5.1 — This page will help you set up a database connection (DataSource) in Websphere 5.1, through the Websphere admin console.
- Known Java Issues — Before installing Java, please take a look at this page for any existing issues.
- LicenseFactory error after upgrading JIRA
- Licensing
- Logging request headers
- Solaris ClassNotFoundException
- Transaction Isolation Change exceptions with JBoss
- Windows cannot find -Xms128m

Performance FAQ

- Finding out how many requests a web application currently has
- Indexing in JIRA
    - Thread Dump in JIRA Log File
        - Sample Thread Dump
- Issue Caching Problems in early JIRA versions
- Is your JIRA Running Slowly
    - Testing Database Access Speed
    - Testing Disk Access Speed
    - Windows Performance Manager
- Profiling Memory and CPU usage with YourKit
    - Setting Up YourKit Agent Parameter
    - Troubleshooting Profiling Memory and CPU usage with YourKit
- Slow JIRA Operations

Usage FAQ

- How do I unlock my JIRA home directory?
- Changing the Project Key
- Mail error - Unable to relay
- Importing user from LDAP
    - Neat JIRA LDAP tricks — Gianugo has an interesting blog up about how to take the pain out of migrating users from LDAP
- JIRA Standalone quick source modifications
- Commonly Asked CSV Questions and Known Issues
- Troubleshooting Issue Creation Via Email — Use this excerpt to provide a brief explanation of what this how-to will do.
- How to clear the resolution field when the issue is reopened
- How to re-order the list of issue operation in an issue
- How to resize Free Text Field customfield
- Using validators to make custom fields required on transition screens — Use the Fields Required workflow validator that is packaged in the JIRA Suite Utilities.
- Project-specific email templates
- How to configure sub-task to have a specific screen?
- User access logging — How to track user actions with page access logging
- Connecting to SSL services
- Editing a custom field option
- Allow editing of Closed Issues
- Updating your JIRA License Details
- Linking to local file under Firefox
- Creating Issues via direct HTML links
- How to create a download link to a file
- Current Reporter Browse Project Permission
- How to display custom field of the sub-task in the parent issue screen?
- How the CreateOrCommentHandler works?
- International characters in notification email subjects
- Sending a JIRA data backup to support
- Letting customers only create issues — This page describes a minor JIRA modification which redirects users to an arbitrary page after creating issues (and potentially other operations). It is mainly of interest to JIRA **Professional** and **Standard** users.
- Asking for an attachment on the Create Issue page
- Limiting the number of issues returned from a search view such as an RSS feed
- Anonymising data
- Single Sign-on
- Scheme Entity Relations Map — This diagram illustrates the relationships between various JIRA entities (Enterprise Edition) and schemes.
- Custom field column not visible in Issue Navigator
- XML format for import & export files
- Configuring project specific security
    - Controlling project visibility
    - Using Project Level Security with Project Roles — This tutorial (for JIRA Enterprise Customers) provides a step-by-step guide for creating project roles and using them in an issue security scheme. We recommend creating a test project and two test users for this tutorial.
    - Using Project Level Security with User Groups
- Changing Custom Field Types
- Re-order workflow transactions
- Creating an Unassigned Issue
- Workflow Properties
- Editing the work log
- Field Layout Schemes in JIRA 3.x
- How to display a different format for the Number customfield
- How to remove 'NONE' from a select list custom field
- Plans for JIRA's LDAP integration
- What are the differences between JIRA Enterprise, Professional and Standard editions?

- Obsolete guides — Administration guides applying to older versions of JIRA
    - Changing scope of existing custom fields
    - Large text fields in Oracle
    - Removing invalid characters from XML backups — In older versions of JIRA it was possible to cut & paste text containing control characters into JIRA issue fields. This causes problems, because JIRA's backup format is XML, and XML does not allow for the storage of most control characters .
- Changing JIRA's log output — A common task when identifying JIRA problems is to turn up the log level, to get more debug-level logs.
    - Logging email protocol details — How to enable email protocol (SMTP, IMAP, POP) logging in JIRA
- Changing the username in JIRA
- Changing the Due Date input format
- Extra CSV Import Features
- Allowing users to create issues anonymously
- Deleting a user account
- Changing the Issue Key format
- QuickSearch guesses the issue key prefix (sometimes)
- Outward Link Description and Inward Link Description
- What does JIRA mean?
- Can I store customer details, like company, address and contact information, in JIRA?
- Words ignored when searching
- Using JIRA to Manage reusable modules
- Retrieving the JIRA Administrator
- Merging 2 JIRA instances
- Group Name Guidelines for JIRA
- Where are the JIRA logs? — A decision tree for finding JIRA logs on your system
- Escalating issues (or sending email notifications) when the set turnaround time is exceeded
- Why doesn't JIRA have a Severity field like Bugzilla?
- How to search by number range in the Issue Navigator
- Troubleshoot Mail Notification Problems
- Resolved issues appearing in Open issues filters
- Parsing utf-7 emails
- Adding custom content to the front page
- Changing the default session timeout
- How is JIRA pronounced?
- Translating JIRA
- How to ensure the Road Map tab is visible
- How to have long component version names display properly in the Issue Navigator
- Login problems
- We already have users & groups defined elsewhere - can JIRA make use of these?
- Associating a new screen with issue operations — When you spend some quality time creating a new screen with custom fields, you want to make people use it. This quick and dirty how-to helps make your screen implementation dreams come true!
- CVS ssh Jira Integration
- JIRA as a Support System
    - Jelly Escalation
    - Simple Escalation
- Changing the default attachment size limit
- Importing data

Tips of the Trade

Also check out the JIRA Community Space and the Forums

Installation Notes

JIRA Administrators FAQ

Installation Notes

- Application Server Notes — These pages contain notes on deploying JIRA in various application servers.
  - Geronimo Notes
  - JBoss 4 notes
  - Jetty Notes
  - JOnAS Notes
  - Oracle Weblogic Notes
  - Resin Notes
  - Sun Java App Server (SJAS or Glassfish) Notes
  - Tomcat 5.5 notes
  - Tomcat 6.0 notes
  - Websphere 5.1 Notes
  - Websphere 6.x Notes
- Database Notes — These pages contain notes on configuring JIRA with various databases.
  - JIRA and DB2
  - JIRA and Firebird
    - Modifications to entitymodel.xml for Firebird
  - JIRA and HSQL
    - Running SQL commands in a HSQL database — On rare occasions, one may wish to run raw SQL queries on a JIRA or Confluence database. This page describes how to obtain a SQL console for hsqldb databases, which JIRA and Confluence Standalone editions use by default.
  - JIRA and MS SQL Server
    - Connecting to named instances in SQL Server
    - Error caused by SET NOCOUNT in MS SQL Server
    - MS SQL Server 2000 Startup errors
    - Setting up your JIRA database for MS SQL Server 2005
  - JIRA and MYSQL
    - Configuring MySQL 4.1 to store non-ASCII characters
    - Data Access Exception - Errcode - 17
    - JIRA can't connect to MYSQL with Named Pipes enabled
    - MYSQL Administrator and Data Truncation Errors
    - Setting up JIRA Standalone and MySQL on Linux
  - JIRA and Oracle
    - Configuring Datasource for Oracle 10g JDBC drivers
    - Restoring data using I-Net (Oranxo) Driver for Oracle
    - Store Workflow on Disk with Oracle 8 — A workaround for the problem of > 4000 character workflows in Oracle 8 is to store these on disk, instead of in the database.
  - JIRA and PostgreSQL
    - Setting up JIRA Standalone and PostgreSQL on Linux
  - JIRA and Sybase
  - Connecting JIRA to a different database than the one provided as default
  - Database Disappears
  - Incorrect database type specified
  - Making databases read-only for upgrade backups
  - Restarting from Setup Wizard in JIRA Standalone
  - Surviving Connection Closures
- Configure JIRA as service on Mac OS X
- Configuring IIS with Tomcat
- Deploying JIRA in a clustered environment
- Installation Troubleshooting Guide
- Installing a LDAP server on Debian Linux for use with JIRA
- Installing Java on Ubuntu or Debian
- java.lang.NoClassDefFoundError
- JVM and Appserver configuration info
  - Causes of OutOfMemoryErrors — When memory problems do occur, the following checklist can help you identify the cause.
  - Issues with tomcat 4.x mail and activation jars
  - Java crashes
  - Setting up a database connection in Websphere 5.1 — This page will help you set up a database connection (DataSource) in Websphere 5.1, through the Websphere admin console.
- Known Java Issues — Before installing Java, please take a look at this page for any existing issues.
- LicenseFactory error after upgrading JIRA
- Licensing
- Logging request headers
- Solaris ClassNotFoundException
- Transaction Isolation Change exceptions with JBoss
- Windows cannot find -Xms128m

Application Server Notes

These pages contain notes on deploying JIRA in various application servers. They are supplementary to the JIRA documentation. If you've ever thought "I wish I'd known that when I started", please help others by adding a note to the relevant application server page.

- Geronimo Notes
- JBoss 4 notes
- Jetty Notes
- JOnAS Notes
- Oracle Weblogic Notes
- Resin Notes
- Sun Java App Server (SJAS or Glassfish) Notes
- Tomcat 5.5 notes
- Tomcat 6.0 notes
- Websphere 5.1 Notes
- Websphere 6.x Notes

Geronimo Notes

This page has general notes on installing JIRA on Apache Geronimo.

Geronimo is an unsupported environment
Geronimo is an **unsupported environment**. Please note that these instructions have only been followed on Geronimo 2.1.3 bundled with Tomcat 6 using JIRA 3.13.2. Other versions of Geronimo/JIRA may or may not work.

1. Unpack JIRA

Download and unzip JIRA (but not with XP's unzipper nor the default tar utility on Solaris). Ensure that you download the **WAR/EAR version,** not the Standalone version that is recommended on the Downloads page.

Avoid the Windows XP built-in unzip tool! The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

Avoid the Solaris default tar utility! On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

A new directory containing JIRA will be created, hereafter referred to as `$JIRA_INSTALL`.

If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
```
$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

2. Configure JIRA

Create geronimo-web.xml

Geronimo requires a geronimo-web.xml deployment descriptor in order to deploy WAR applications. Create a new file `$JIRA_INSTALL/webapp/WEB-INF/geronimo-web.xml` and add the following:

geronimo-web.xml

```
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1"
xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.2"
xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0"
xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.2">

        <!-- version information about the JIRA installation -->
        <sys:environment>
                <sys:moduleId>
                        <sys:groupId>com.atlassian.jira</sys:groupId>
                        <sys:artifactId>Jira</sys:artifactId>
                        <sys:version>3.13.2</sys:version>
                        <sys:type>war</sys:type>
                </sys:moduleId>

                <!-- database dependency - JIRA won't start without a database, so lets tell geronimo
about this -->
                <sys:dependencies>
                        <sys:dependency>
                                <sys:groupId>console.dbpool</sys:groupId>
                                <sys:artifactId>JiraDS</sys:artifactId>
                        </sys:dependency>
                </sys:dependencies>
        </sys:environment>

        <!-- the context path to deploy the war too - in this case it will be http:
//localhost:8080/jira -->
<context-root>/jira</context-root>
        <naming:resource-ref>
                <!-- the name of the database pool in web.xml -->
                <naming:ref-name>jdbc/JiraDS</naming:ref-name>
                <!-- the name of the database pool you created in Geronimo console -->
                <naming:resource-link>jdbc/JiraDS</naming:resource-link>
        </naming:resource-ref>
</web-app>
```

Edit entityengine.xml

In addition to setting the appropriate field type for you database, you also need to configure the Transaction factory for Geronimo. In `$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml` find the <transaction-factory> element and replace it with the following:

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
    <user-transaction-jndi jndi-server-name="default" jndi-name="java:comp/UserTransaction"/>
    <transaction-manager-jndi jndi-server-name="default" jndi-name="java:/TransactionManager"/>
</transaction-factory>
```

Also find the <datasource> element and change the <jndi-jdbc> to have the following:

```
<jndi-jdbc jndi-server-name="default" jndi-name="java:comp/env/jdbc/JiraDS"/>
```

The jndi-name attibute is a "java:comp/env/" plus whatever you specified for `ref-name` in geronimo-web.xml.

Edit web.xml

The web.xml file needs to be edited to tell Geronimo where it can find the data source for the JIRA database. Copy `$JIRA_INSTALL/webapp/WEB-INF/web.xml` to `$JIRA_INSTALL/edit-webapp/WEB-INF/web.xml` and add the following under the <web-app> element:

```
<resource-ref>
    <res-ref-name>jdbc/JiraDS</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

3. Build JIRA

JIRA can now be built by typing **build** (Windows) or **./build.sh** (Unix) on the command line in the $JIRA_HOME directory. This will produce the deployable WAR file in the **$JIRA_INSTALL/dist-*** directories.

4. Deploy JIRA

Create Database Connection Pool in Geronimo Console

The simplest way to do this is by the GUI wizard under Services -> Databases in the Geronimo console. Name the pool `jdbc/JiraDS` (or whatever you specified for `resource-link` in geronimo-web.xml). Please refer to http://cwiki.apache.org/GMOxDOC22/configuring-database-pools.html for more information on how to configure a database connection pool.

Deploy WAR in Geronimo Console

Select Application -> Deploy New from inside the Geronimo Console. Click the browse button next to the Archive and select the `$JIRA_INSTALL/dist-generic/atlassian-jira-<version>.war`. Leave the plan blank and click install.

5. Set JIRA Home

To specify the location of your JIRA Home Directory (note that you need to do this *before* you build JIRA):

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

6. Set mail.mime.decodeparameters

The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

```
mail.mime.decodeparameters=true
```

System properties are set in different ways depending on your application server.

7. Access JIRA

You should now be able to visit the JIRA setup page at `http://localhost:8080/jira` (assuming Geronimo is running on the local machine on port 8080).

8. Known Issues

- When deploying JIRA it fails and I get the following error in the logs:

```
14:08:32,349 ERROR [Deployer] Deployment failed due to
java.lang.ArrayIndexOutOfBoundsException: 12437
 at org.objectweb.asm.ClassReader.<init>(Unknown Source)
 at org.objectweb.asm.ClassReader.<init>(Unknown Source)
 at org.objectweb.asm.ClassReader.<init>(Unknown Source)
 at org.apache.xbean.finder.ClassFinder.readClassDef(ClassFinder.java:689)
 at org.apache.xbean.finder.ClassFinder.<init>(ClassFinder.java:139)
 at
org.apache.geronimo.jaxws.builder.WARWebServiceFinder.discoverWebServices(WARWebServiceFinder.java
at
org.apache.geronimo.jaxws.builder.AdvancedWARWebServiceFinder.discoverPOJOWebServices(AdvancedWARW
at
org.apache.geronimo.jaxws.builder.AdvancedWARWebServiceFinder.discoverWebServices(AdvancedWARWebSe
at
org.apache.geronimo.jaxws.builder.WARWebServiceFinder.discoverWebServices(WARWebServiceFinder.java
```

The solution is to tell Geronimo to use the Simple Web Service finder by adding the `-Dorg.apache.geronimo.jaxws.builder.useSimpleFinder=true"` property to the $JAVA_OPTS environment variable.
http://www.nabble.com/Problem-with-Deploy-in-Geronimo-2.1.2-td18915589s134.html

- Geronimo 2.1.3 will raise warnings about invalid taglib descriptors

```
14:59:09,326 WARN  [JspModuleBuilderExtension] Invalid transformed taglib
org.apache.xmlbeans.XmlException: Invalid deployment descriptor: errors:

/Users/amyers/Applications/appservers/geronimo-tomcat6-javaee5-2.1.3/repository/com/atlassian/jira
cvc-datatype-valid.1.1: string value 'SiteMesh Decorator Tags' does not match pattern
for tld-canonical-nameType in namespace http://java.sun.com/xml/ns/javaee
```

JIRA still seems to work depite these warnings.

add your notes

JBoss 4 notes

This page has general notes on installing JIRA on JBoss 4.

Attached are console logs from installing JIRA on JBoss on a Linux (Ubuntu) server:

- jboss-4.0.5-hsql-installlog.txt
- jboss-4.2.0-postgresql-installlog.txt

JBoss does odd things with classloaders, and as a result its own log4j configuration file is found before JIRA's own. This means (on 4.0.x) that you will get lots of DEBUG-level logs in stdout.

Jetty Notes

This page has general notes on installing JIRA on Jetty.

A user reported:

> *We use a custom classloader with Jetty 5.0. Probably because of Jetty's dependence on commons logging, Jetty was confused over Jira's inclusion of the same jar. Also, since Jira uses log4j to implement the logging, there was some classloading confusion over where to find it.*
>
> *To get Jira running, I removed the commons-logging jar from Jira and placed log4j in jetty/ext/ (where commons-logging already was). Also, I symlinked the Postgres JDBC jar into Jira's WEB-INF/lib directory.*

JOnAS Notes

This page has general notes on installing JIRA on JOnAS. JOnAS is currently an **unsupported environment.**

add your notes

A user reports:

> *Concerning the problem with the autocommit error, it seems to be coming from JOnAS.*
> *This application can use two different modes to connect to a database :*
>
> - *a .properties file using a service called "dbm"*
> - *a .rar file using a service called "resource"*
>
> *The two modes don't manage the autocommit in the same way.*
> *The first one calls setAutocommit(true) only if getAutocommit() != true*
> *The second one always calls setAutocommit(true)*
>
> *We were using the second one and that's why we got the error message :*
> *SEVERE : main : ConnectionImpl.setAutoCommit : ConnectionImpl.setAC: error*
>
> *Now we are using the first one and we don't have the error message anymore.*

Oracle Weblogic Notes

This page has general notes on installing JIRA on Weblogic. It supplements the official Weblogic installation docs

add your notes

- **JIRA 3.7.2** appears to work under WebLogic 8.1 SP 6 (tested against Postgres 8.2).
- Currently there are issues with running JIRA under WebLogic 8.1.x with Oracle 10.x.

Resin Notes

This page has general notes on installing JIRA on Resin.

- keep-alive bug with Apache 2 affecting JIRA performance
- Error on startup using schema name 'JIRA'
- Error when restoring from backup XML using Resin 3.0.x
- JIRA and SOAP on Resin
- Trusted Application feature fails on Resin 2.1.x

keep-alive bug with Apache 2 affecting JIRA performance

On http://jira.atlassian.com (running Resin behind Apache), we found that Resin was serving each hit with a `Connection: Close` header. This meant that for each resource requested (image, CSS, html etc) the browser needed to open a new connection, slowing things down.

This turned out to be a bug in Resin when used with Apache. We upgraded to Resin 3.0.19 and this fixed the problem. Resources are now served with `Connection: Keep-Alive`.

To check if you are experiencing this problem, download livehttpheaders for Firefox, and look at the response headers for a JIRA page request.

Error on startup using schema name 'JIRA'

On startup, Jira reports that tables are missing from the database even though the database contains the specified tables. The problem only occurs with tables whose names start with "jira" No such errors are reported for other tables. Here is the reported exception in the logs:

```
2006-10-04 16:57:56,959 [core.entity.jdbc.DatabaseUtil] Entity "Action" has no table in the database
2006-10-04 16:57:56,972 [core.entity.jdbc.DatabaseUtil] Could not create table "JIRA.jiraaction"
2006-10-04 16:57:56,973 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE JIRA.jiraaction (ID DECIMAL(18,0) NOT NULL, issueid DECIMAL(18,0), AUTHOR VARCHAR(255),
actiontype VARCHAR(255), actionlevel VARCHAR(255), actionbody CLOB, CREATED TIMESTAMP, actionnum
DECIMAL(18,0), CONSTRAINT PK_jiraaction PRIMARY KEY (ID))
Error was: com.ibm.db2.jcc.c.SqlException: DB2 SQL error: SQLCODE: -601, SQLSTATE: 42710, SQLERRMC:
JIRA.JIRAACTION;TABLE
2006-10-04 16:57:56,974 [core.entity.jdbc.DatabaseUtil] Entity "EventType" has no table in the
database
2006-10-04 16:57:56,976 [core.entity.jdbc.DatabaseUtil] Could not create table "JIRA.jiraeventtype"
2006-10-04 16:57:56,977 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE JIRA.jiraeventtype (ID DECIMAL(18,0) NOT NULL, TEMPLATE_ID DECIMAL(18,0), NAME
VARCHAR(255), DESCRIPTION CLOB, TYPE VARCHAR(60), CONSTRAINT PK_jiraeventtyp PRIMARY KEY (ID))
Error was: com.ibm.db2.jcc.c.SqlException: DB2 SQL error: SQLCODE: -601, SQLSTATE: 42710, SQLERRMC:
JIRA.JIRAEVENTTYPE;TABLE
2006-10-04 16:57:56,979 [core.entity.jdbc.DatabaseUtil] Entity "Issue" has no table in the database
2006-10-04 16:57:56,981 [core.entity.jdbc.DatabaseUtil] Could not create table "JIRA.jiraissue"
2006-10-04 16:57:56,981 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE JIRA.jiraissue (ID DECIMAL(18,0) NOT NULL, pkey VARCHAR(255), PROJECT DECIMAL(18,0),
REPORTER VARCHAR(255), ASSIGNEE VARCHAR(255), issuetype VARCHAR(255), SUMMARY VARCHAR(255),
DESCRIPTION CLOB, ENVIRONMENT CLOB, PRIORITY VARCHAR(255), RESOLUTION VARCHAR(255), issuestatus
VARCHAR(255), CREATED TIMESTAMP, UPDATED TIMESTAMP, DUEDATE TIMESTAMP, VOTES DECIMAL(18,0),
TIMEORIGINALESTIMATE DECIMAL(18,0), TIMEESTIMATE DECIMAL(18,0), TIMESPENT DECIMAL(18,0), WORKFLOW_ID
DECIMAL(18,0), SECURITY DECIMAL(18,0), FIXFOR DECIMAL(18,0), COMPONENT DECIMAL(18,0), CONSTRAINT
PK_jiraissue PRIMARY KEY (ID))
Error was: com.ibm.db2.jcc.c.SqlException: DB2 SQL error: SQLCODE: -601, SQLSTATE: 42710, SQLERRMC:
JIRA.JIRAISSUE;TABLE
2006-10-04 16:57:56,984 [core.entity.jdbc.DatabaseUtil] Entity "Permission" has no table in the
database
2006-10-04 16:57:56,986 [core.entity.jdbc.DatabaseUtil] Could not create table "JIRA.jiraperms"
2006-10-04 16:57:56,986 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE JIRA.jiraperms (ID DECIMAL(18,0) NOT NULL, permtype DECIMAL(18,0), projectid
DECIMAL(18,0), groupname VARCHAR(255), CONSTRAINT PK_jiraperms PRIMARY KEY (ID))
Error was: com.ibm.db2.jcc.c.SqlException: DB2 SQL error: SQLCODE: -601, SQLSTATE: 42710, SQLERRMC:
JIRA.JIRAPERMS;TABLE
2006-10-04 16:57:56,990 [core.entity.jdbc.DatabaseUtil] Entity "Workflow" has no table in the database
2006-10-04 16:57:56,992 [core.entity.jdbc.DatabaseUtil] Could not create table "JIRA.jiraworkflows"
2006-10-04 16:57:56,992 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE JIRA.jiraworkflows (ID DECIMAL(18,0) NOT NULL, workflowname VARCHAR(255), creatorname
VARCHAR(255), DESCRIPTOR CLOB, ISLOCKED VARCHAR(60), CONSTRAINT PK_jiraworkflow PRIMARY KEY (ID))
Error was: com.ibm.db2.jcc.c.SqlException: DB2 SQL error: SQLCODE: -601, SQLSTATE: 42710, SQLERRMC:
JIRA.JIRAWORKFLOWS;TABLE
2006-10-04 16:57:56,993 [core.entity.jdbc.DatabaseUtil] Table named "JIRAACTION" exists in the
database but has no corresponding entity
```

We got this error just after installation and first restart. This was before we went to the Jira start up page.

We noticed that the problem only occurs with schema name "JIRA" - no errors were reported when we used another schema like "JIRATEST".

Error when restoring from backup XML using Resin 3.0.x

Importing a backup XML via (Administration -> Import & Export -> Backup Data to XML) produces the following error on Resin 3.0.19

```
[11:25:27.312] Closing dangling connections.  All connections must have a close() in a finally block.
[11:25:27.312] java.lang.IllegalStateException: Connection
UserConnection[com.caucho.sql.ManagedConnectionImpl@12398ed] was not closed. Connections must have a
close() in a finally block.
```

which was a bug raised here. Upgrading to Resin 3.0.21 seems to have fixed this.

JIRA and SOAP on Resin

Symptom

The WSDL generated by JIRA on the Resin application server is invalid. Attempting to parse it gives you an error like this (using Axis as the

client):

> *WSDLException (at /wsdl:definitions/types): faultCode=INVALID_WSDL: Encountered illegal extension element 'types' in the context of a 'javax.wsdl.Definition'. Extension elements must be in a namespace other than WSDL's.:*

Cause

The default XML parser included with the Resin application server is incompatible with the Axis SOAP library, and causes Axis to generate an invalid WSDL file. You can find information about this issue here:

- http://issues.apache.org/jira/browse/AXIS-2235

Solution

The solution is to override Resin's default XML parser by placing the following within the `<server>` section of `resin.conf`:

```
<system-property javax.xml.parsers.DocumentBuilderFactory=
"org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"/>
<system-property javax.xml.parsers.SAXParserFactory="org.apache.xerces.jaxp.SAXParserFactoryImpl"/>
```

(You can also make this configuration per web-app by putting it in the `<web-app>` section. For more information see this page: JAXP - specifying XML and XSLT implementations)

The Confluence issue for this error is here:

- http://jira.atlassian.com/browse/CONF-5953

Trusted Application feature fails on Resin 2.1.x

Symptom

As reported in this issue, if theTrusted Application feature is not working and the following is seen in the logs

```
WARN [atlassian.seraph.filter.TrustedApplicationsFilter] Failed to login trusted application:
confluence1234567 due to:
    com.atlassian.security.auth.trustedapps.InvalidCertificateException:
TestTrustedApplicationClient.id
    cause: java.lang.SecurityException: JCE cannot authenticate the provider BC
```

The important bit is: `java.lang.SecurityException:` **JCE cannot authenticate the provider BC**

To diagnose you can [turn on DEBUG logging|link] for the *com.atlassian.seraph* package to see the stack trace in the log. If the following is shown:

```
Caused by: java.lang.SecurityException: JCE cannot authenticate the provider BC
 at javax.crypto.Cipher.getInstance(DashoA12275)
 at javax.crypto.Cipher.getInstance(DashoA12275)
 at
com.atlassian.security.auth.trustedapps.BouncyCastleEncryptionProvider.decodeEncryptedCertificate(BouncyC
... 29 more
Caused by: java.util.jar.JarException: Class is on the bootclasspath
 at javax.crypto.SunJCE_d.a(DashoA12275)
 at javax.crypto.SunJCE_b.b(DashoA12275)
 at javax.crypto.SunJCE_b.a(DashoA12275)
 ... 32 more
```

and the bcprov jar is definitely *not* on the boot class path, the problem is caused by an errant ClassLoader that defines the class without providing a valid ProtectionDomain object. Essentially, security provider classes **must** be loaded using the correct ClassLoader method.

Solution

The solution is to add the Bouncy Castle security provider jar to the application server classpath. For Resin this means to add `bcprov-jdk14-138.jar` to the `server/lib` directory.

---

add your notes

Sun Java App Server (SJAS or Glassfish) Notes

This page has general notes on installing JIRA on Sun's Glassfish / Java App Server, bundled with Java EE. SJAS is currently an **unsupported environment**.

Steeplesoft have also posted a blog on how to Deploy JIRA in Glassfish.

Please use JIRA 3.12 or above, which fixes a problem where JIRA's login "remember me" functionality broke (JRA-13400).

add your notes

Tomcat 5.5 notes

This page has general notes on installing JIRA on Tomcat 5.5.x. It supplements the official Tomcat installation docs.

A user writes:

There is a readme file shipped with JIRA. It said that you need to change port 8080 in order to get Tomcat(say a) to start up. **BUT**, if you have a **CATALINA_HOME** environment variable already set for an existing Tomcat(say b) running on the same server, running startup from the bin directory will start up Tomcat(b). For Tomcat(a) to start one needs to delete the existing **CATALINA_HOME** environment variable as well.

Tomcat 6.0 notes

This page has general notes on installing JIRA on Tomcat 6.0.x. It supplements the official Tomcat installation docs.

Add your notes

Websphere 5.1 Notes

This page has general notes on installing JIRA on Websphere 5.1.x.

Note that from 3.5.1 to 3.6.3+, Issue exports in MS Word format is broken

---

add your notes

Websphere 6.x Notes

This page has general notes on installing JIRA on Websphere 6.x. It supplements the official Websphere 6.x installation docs

Associating new Workflow Scheme Errors

A customer reports of getting the following error when associating a new workflow scheme to a project using DB2

```
Cause:
com.atlassian.jira.exception.DataAccessException: org.ofbiz.core.util.GeneralRuntimeException: Error
getting the next result (Invalid operation: result set closed)

Stack Trace: [hide]

com.atlassian.jira.exception.DataAccessException: org.ofbiz.core.util.GeneralRuntimeException: Error
getting the next result (Invalid operation: result set closed)
at
com.atlassian.jira.web.action.admin.workflow.WorkflowMigrationManager.migrate(WorkflowMigrationManager.ja
com.atlassian.jira.web.action.admin.workflow.scheme.SelectProjectWorkflowSchemeStep2.doExecute(SelectProj
webwork.action.ActionSupport.execute(ActionSupport.java:153)
at com.atlassian.jira.action.JiraActionSupport.execute(JiraActionSupport.java:57)
at webwork.dispatcher.GenericDispatcher.executeAction(GenericDispatcher.java:132)
at com.atlassian.jira.web.dispatcher.JiraServletDispatcher.service(JiraServletDispatcher.java:185)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1282)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1239)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:136)
at com.atlassian.jira.web.filters.AccessLogFilter.doFilter(AccessLogFilter.java:51)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
```

```
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.opensymphony.module.sitemesh.filter.PageFilter.parsePage(PageFilter.java:119)
at com.opensymphony.module.sitemesh.filter.PageFilter.doFilter(PageFilter.java:55)
at
com.atlassian.jira.web.filters.SitemeshExcludePathFilter.doFilter(SitemeshExcludePathFilter.java:38)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.seraph.filter.SecurityFilter.doFilter(SecurityFilter.java:182)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.seraph.filter.LoginFilter.doFilter(LoginFilter.java:181)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.util.profiling.filters.ProfilingFilter.doFilter(ProfilingFilter.java:132)
at com.atlassian.jira.web.filters.ProfilingAndErrorFilter.doFilter(ProfilingAndErrorFilter.java:35)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.jira.web.filters.ActionCleanupDelayFilter.doFilter(ActionCleanupDelayFilter.java:39)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.johnson.filters.JohnsonFilter.doFilter(JohnsonFilter.java:91)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.jira.web.filters.gzip.GzipFilter.doFilter(GzipFilter.java:72)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.atlassian.core.filters.AbstractEncodingFilter.doFilter(AbstractEncodingFilter.java:37)
at com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:142)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:121)
at com.ibm.ws.webcontainer.filter.WebAppFilterChain._doFilter(WebAppFilterChain.java:82)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:670)
at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:2954)
at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:221)
at com.ibm.ws.webcontainer.VirtualHost.handleRequest(VirtualHost.java:210)
at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:1912)
at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:84)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLink.java:472)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewInformation(HttpInboundLink.java:411)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.ready(HttpInboundLink.java:288)
at
com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.sendToDiscriminaters(NewConnectionInitialRea
com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.complete(NewConnectionInitialReadCallback.ja
com.ibm.ws.tcp.channel.impl.WorkQueueManager.requestComplete(WorkQueueManager.java:566)
at com.ibm.ws.tcp.channel.impl.WorkQueueManager.attemptIO(WorkQueueManager.java:619)
at com.ibm.ws.tcp.channel.impl.WorkQueueManager.workerRun(WorkQueueManager.java:952)
at com.ibm.ws.tcp.channel.impl.WorkQueueManager$Worker.run(WorkQueueManager.java:1039)
at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1469)
Caused by: org.ofbiz.core.util.GeneralRuntimeException: Error getting the next result (Invalid
operation: result set closed)
at org.ofbiz.core.entity.EntityListIterator.next(EntityListIterator.java:197)
at
```

```
com.atlassian.jira.web.action.admin.workflow.WorkflowMigrationManager.migrate(WorkflowMigrationManager.ja
56 more
```

This was rectified by setting the JDBC-Provider default setting in WebSphere as follows:

```
JDBC-Provider > DB2 Universal JDBC Driver Provider (XA) > Data sources (Version 4) > <your database
name> > Custom properties
```

and setting the **"resultSetHoldability"** value to 1

http://confluence.atlassian.com/pages/editpage.action?pageId=199337

Database Notes

These pages contain notes on configuring JIRA with various databases. They are supplementary to the JIRA documentation. If you've ever thought "I wish I'd known that when I started", please help others by adding a note to the relevant database page.

Recommended Databases
Please read the Recommended Databases document to see the list of databases that we recommend for use with your JIRA installation.

- JIRA and DB2
- JIRA and Firebird
  - Modifications to entitymodel.xml for Firebird
- JIRA and HSQL
  - Running SQL commands in a HSQL database — On rare occasions, one may wish to run raw SQL queries on a JIRA or Confluence database. This page describes how to obtain a SQL console for hsqldb databases, which JIRA and Confluence Standalone editions use by default.
- JIRA and MS SQL Server
  - Connecting to named instances in SQL Server
  - Error caused by SET NOCOUNT in MS SQL Server
  - MS SQL Server 2000 Startup errors
  - Setting up your JIRA database for MS SQL Server 2005
- JIRA and MYSQL
  - Configuring MySQL 4.1 to store non-ASCII characters
  - Data Access Exception - Errcode - 17
  - JIRA can't connect to MYSQL with Named Pipes enabled
  - MYSQL Administrator and Data Truncation Errors
  - Setting up JIRA Standalone and MySQL on Linux
- JIRA and Oracle
  - Configuring Datasource for Oracle 10g JDBC drivers
  - Restoring data using I-Net (Oranxo) Driver for Oracle
  - Store Workflow on Disk with Oracle 8 — A workaround for the problem of > 4000 character workflows in Oracle 8 is to store these on disk, instead of in the database.
- JIRA and PostgreSQL
  - Setting up JIRA Standalone and PostgreSQL on Linux
- JIRA and Sybase
- Connecting JIRA to a different database than the one provided as default
- Database Disappears
- Incorrect database type specified
- Making databases read-only for upgrade backups
- Restarting from Setup Wizard in JIRA Standalone
- Surviving Connection Closures

JIRA and DB2

This page has general notes on connecting JIRA to DB2. It supplements the official DB2 installation documentation.

JIRA and Firebird

This page has general notes on connecting JIRA to Firebird. It supplements the official Firebird installation documentation.

- Modifications to entitymodel.xml for Firebird

Modifications to entitymodel.xml for Firebird

Update Note
THIS PAGE IS NOT APPLICABLE FOR JIRA 3.7 and up.

JIRA on FireBird

In order to run JIRA with the Firebird DB, the entitymodel.xml needs to be modified to adhere to the following Firebird constraints:

1. **TYPE**, **POSITION** and **PARAMETER** are reserved keywords in Firebird
2. Firebird index keys can not be greater than 250 in size

The attached entitymodel files have been modified to reflect these restrictions:

1. JIRA 3.1.x entitymodel.xml
2. JIRA 3.2.x entitymodel.xml
3. JIRA 3.2.x entitymodel.xml
4. JIRA 3.3.x entitymodel.xml
5. JIRA 3.4.x entitymodel.xml

For other releases, here is a diff of the changes made:

- entitymodel.xml.diff

Performance Note
Due to the index key restriction, the performance of JIRA is greatly reduced as various elements cannot be indexed.

Testing was conducted with Firebird 1.5.2.

JIRA and HSQL

This page has general notes on connecting JIRA to HSQL. It supplements the official HSQL installation documentation.

- Running SQL commands in a HSQL database

Running SQL commands in a HSQL database

On rare occasions, one may wish to run raw SQL queries on a JIRA or Confluence database. This page describes how to obtain a SQL console for hsqldb databases, which JIRA and Confluence Standalone editions use by default.

Locate the hsqldb directory

hsql stores its database in text files on the filesystem. Typically these will be in a `database` directory in the JIRA/Confluence root:

```
[atlassian-jira-professional-3.4.2-standalone ~]$ ls -l database/
total 108
-rw-r--r--    1 jturner  jturner         0 Jul 28 09:12 jiradb.data
-rw-r--r--    1 jturner  jturner       343 Jul 28 09:12 jiradb.properties
-rw-r--r--    1 jturner  jturner     72272 Jul 28 10:02 jiradb.script
[atlassian-jira-professional-3.4.2-standalone ~]$
```

In JIRA, the path is specified in `conf/server.xml` as '${catalina.home}/database/jiradb', where ${catalina.home} will be the directory *atlassian-jira-professional-3.4.2-standalone* in this instance.

Locate the hsqldb jar

The hsqldb binary is usually located in the `common/lib/` directory:

```
[atlassian-jira-professional-3.4.2-standalone ~]$ ls common/lib/hsql*
common/lib/hsqldb-1.7.1-patched.jar
```

Shut down JIRA/Confluence

If you haven't already, shut down any apps using the database.

Run the console

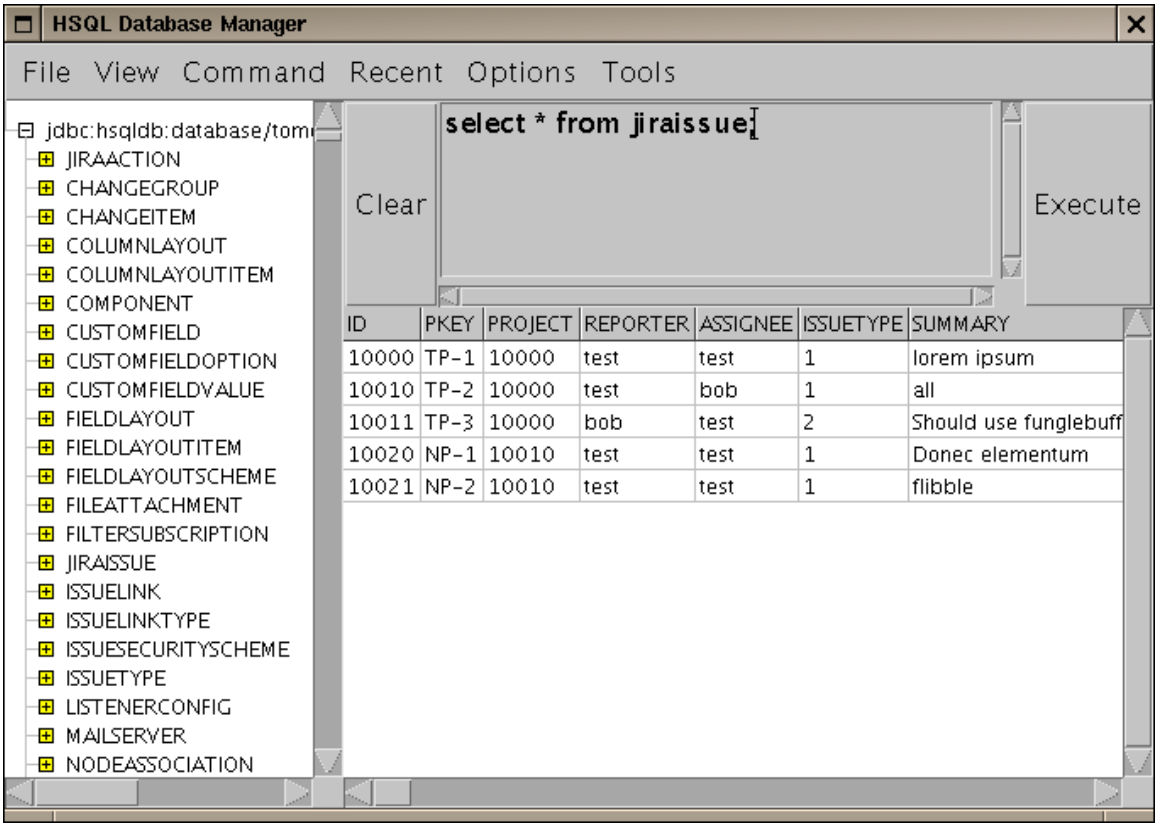
Run the following command (JIRA 3.7+):

```
java -cp common/lib/hsqldb-1.8.0.5.jar org.hsqldb.util.DatabaseManager -user sa -url jdbc:hsqldb:database/jiradb
```

or this for JIRA 3.6.5 and earlier:

```
java -cp common/lib/hsqldb-1.7.1-patched.jar org.hsqldb.util.DatabaseManager -user sa -url jdbc:hsqldb:database/jiradb
```

The hsqldb console should load, listing tables in the database in the left panel. You can run SQL commands in the top panel:



Once you have finished running SQL queries, shut down the console before starting JIRA/Confluence.

JIRA and MS SQL Server

This page has general notes on connecting JIRA to SQL Server. It supplements the official [SQL Server installation documentation].

- Connecting to named instances in SQL Server
- Error caused by SET NOCOUNT in MS SQL Server
- MS SQL Server 2000 Startup errors
- Setting up your JIRA database for MS SQL Server 2005

Connecting to named instances in SQL Server

When using named instances you will need to specify the URL slightly differently in the connection properties.

First off, try:

```
url="jdbc:jtds:<server_type>://<server>[:<port>][/<database>];instance=<instance_name>"
```

This is specified at the JTDS FAQ

If this doesn't work, try dropping the instance name, and changing the port to the port used by the named instance:

```
url="jdbc:jtds:<server_type>://<server>[:<instance_port>][/<database>]"
```

Note. This port is different to the normal SQL Server port as each instance listens on a different port.

Error caused by SET NOCOUNT in MS SQL Server

It is necessary to ensure that the **SET NOCOUNT** option is not set in the SQL Server configuration. For further details on how to verify these settings, please refer to the JIRA Installation documentation.

If this option is set, it can result in the following errors that can be found in the log file:

```
2006-05-03 15:51:24,331 INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Issue Cache
Listener, adding.

2006-05-03 15:51:26,088 WARN [ofbiz.core.entity.SequenceUtil] [SequenceUtil.SequenceBank.fillBank]
first select failed: trying to add row, result set was
empty for sequence: ListenerConfig

2006-05-03 15:51:26,093 WARN [ofbiz.core.entity.SequenceUtil] [SequenceUtil.SequenceBank.fillBank]
first select failed: trying to add row, result set was empty for sequence: ListenerConfig

2006-05-03 15:51:26,094 ERROR [ofbiz.core.entity.SequenceUtil]
[SequenceUtil.SequenceBank.getNextSeqId] Fill bank failed, returning null

2006-05-03 15:51:26,112 ERROR [atlassian.jira.upgrade.ConsistencyCheckImpl] Error adding listener:
org.ofbiz.core.entity.GenericEntityException: while
inserting:
[GenericEntity:ListenerConfig][clazz,com.atlassian.jira.event.listeners.cache.IssueCacheListener][name,Is
Cache Listener][id,null] (SQL
exception while executing the following:INSERT INTO listenerconfig (ID, CLAZZ, listenername) VALUES
(?, ?, ?) (Cannot insert the value NULL into column
'ID', table 'Jira36Test.JiraUser.listenerconfig'; column does not allow nulls. INSERT fails.))

org.ofbiz.core.entity.GenericEntityException: while inserting:
[GenericEntity:ListenerConfig][clazz,com.atlassian.jira.event.listeners.cache.IssueCacheListener][name,Is
Cache Listener][id,null] (SQL Exception while
executing the following:INSERT INTO listenerconfig (ID, CLAZZ, listenername) VALUES (?, ?, ?) (Cannot
insert the value NULL into column 'ID', table 'Jira36Test.JiraUser.listenerconfig'; column does not
allow nulls. INSERT fails.))

    at org.ofbiz.core.entity.GenericDAO.singleInsert(GenericDAO.java:123)
    at org.ofbiz.core.entity.GenericDAO.insert(GenericDAO.java:88)
    at org.ofbiz.core.entity.GenericHelperDAO.create(GenericHelperDAO.java:63)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:470)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:450)
    at org.ofbiz.core.entity.GenericValue.create(GenericValue.java:77)
    at com.atlassian.core.ofbiz.util.EntityUtils.createValue(EntityUtils.java:61)
    at com.atlassian.jira.action.admin.ListenerCreate.execute(ListenerCreate.java:22)
    at webwork.dispatcher.GenericDispatcher.executeAction(GenericDispatcher.java:132)
    at com.atlassian.core.action.DefaultActionDispatcher.execute(DefaultActionDispatcher.java:34)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.ensureSingleListener(ConsistencyCheckImpl.java:669)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkCacheListener(ConsistencyCheckImpl.java:563)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkDataConsistency(ConsistencyCheckImpl.java:306)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkConsistency(ConsistencyCheckImpl.java:295)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.initialise(ConsistencyCheckImpl.java:164)
    at com.atlassian.jira.upgrade.ConsistencyLauncher.contextInitialized(ConsistencyLauncher.java:27)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3692)
    at org.apache.catalina.core.StandardContext.start(StandardContext.java:4127)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:759)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:739)
```

```
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:524)
    at org.apache.catalina.startup.HostConfig.deployDescriptor(HostConfig.java:603)
    at org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:493)
    at org.apache.catalina.startup.HostConfig.check(HostConfig.java:1195)
    at sun.reflect.GeneratedMethodAccessor341.invoke(Unknown Source)

org.ofbiz.core.entity.GenericDataSourceException: SQL Exception while executing the following:INSERT
INTO listenerconfig (ID, CLAZZ, listenername)
VALUES (?, ?, ?) (Cannot insert the value NULL into column 'ID', table
'Jira36Test.JiraUser.listenerconfig'; column does not allow nulls. INSERT fails.)
    at org.ofbiz.core.entity.jdbc.SQLProcessor.executeUpdate(SQLProcessor.java:375)
    at org.ofbiz.core.entity.GenericDAO.singleInsert(GenericDAO.java:115)
    at org.ofbiz.core.entity.GenericDAO.insert(GenericDAO.java:88)
    at org.ofbiz.core.entity.GenericHelperDAO.create(GenericHelperDAO.java:63)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:470)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:450)
    at org.ofbiz.core.entity.GenericValue.create(GenericValue.java:77)
    at com.atlassian.core.ofbiz.util.EntityUtils.createValue(EntityUtils.java:61)
    at com.atlassian.jira.action.admin.ListenerCreate.execute(ListenerCreate.java:22)
    at webwork.dispatcher.GenericDispatcher.executeAction(GenericDispatcher.java:132)
    at com.atlassian.core.action.DefaultActionDispatcher.execute(DefaultActionDispatcher.java:34)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.ensureSingleListener(ConsistencyCheckImpl.java:669)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkCacheListener(ConsistencyCheckImpl.java:563)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkDataConsistency(ConsistencyCheckImpl.java:306)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkConsistency(ConsistencyCheckImpl.java:295)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.initialise(ConsistencyCheckImpl.java:164)
    at com.atlassian.jira.upgrade.ConsistencyLauncher.contextInitialized(ConsistencyLauncher.java:27)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3692)
   at org.apache.catalina.core.StandardContext.start(StandardContext.java:4127)
    at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:759)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:739)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:524)
    at org.apache.catalina.startup.HostConfig.deployDescriptor(HostConfig.java:603)
    at org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:493)
    at org.apache.catalina.startup.HostConfig.check(HostConfig.java:1195)
    at sun.reflect.GeneratedMethodAccessor341.invoke(Unknown Source)

java.sql.SQLException: Cannot insert the value NULL into column 'ID', table
'Jira36Test.JiraUser.listenerconfig'; column does not allow nulls. INSERT fails.
    at net.sourceforge.jtds.jdbc.SQLDiagnostic.addDiagnostic(SQLDiagnostic.java:365)
    at net.sourceforge.jtds.jdbc.TdsCore.tdsErrorToken(TdsCore.java:2781)
    at net.sourceforge.jtds.jdbc.TdsCore.nextToken(TdsCore.java:2224)
    at net.sourceforge.jtds.jdbc.TdsCore.getMoreResults(TdsCore.java:628)
    at net.sourceforge.jtds.jdbc.JtdsStatement.processResults(JtdsStatement.java:525)
    at net.sourceforge.jtds.jdbc.JtdsStatement.executeSQL(JtdsStatement.java:487)
    at net.sourceforge.jtds.jdbc.JtdsPreparedStatement.executeUpdate(JtdsPreparedStatement.java:421)
    at
org.apache.tomcat.dbcp.dbcp.DelegatingPreparedStatement.executeUpdate(DelegatingPreparedStatement.java:10
at org.ofbiz.core.entity.jdbc.SQLProcessor.executeUpdate(SQLProcessor.java:373)
    at org.ofbiz.core.entity.GenericDAO.singleInsert(GenericDAO.java:115)
    at org.ofbiz.core.entity.GenericDAO.insert(GenericDAO.java:88)
    at org.ofbiz.core.entity.GenericHelperDAO.create(GenericHelperDAO.java:63)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:470)
    at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:450)
    at org.ofbiz.core.entity.GenericValue.create(GenericValue.java:77)
    at com.atlassian.core.ofbiz.util.EntityUtils.createValue(EntityUtils.java:61)
    at com.atlassian.jira.action.admin.ListenerCreate.execute(ListenerCreate.java:22)
    at webwork.dispatcher.GenericDispatcher.executeAction(GenericDispatcher.java:132)
    at com.atlassian.core.action.DefaultActionDispatcher.execute(DefaultActionDispatcher.java:34)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.ensureSingleListener(ConsistencyCheckImpl.java:669)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkCacheListener(ConsistencyCheckImpl.java:563)
    at
com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkDataConsistency(ConsistencyCheckImpl.java:306)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.checkConsistency(ConsistencyCheckImpl.java:295)
    at com.atlassian.jira.upgrade.ConsistencyCheckImpl.initialise(ConsistencyCheckImpl.java:164)
```

```
at com.atlassian.jira.upgrade.ConsistencyLauncher.contextInitialized(ConsistencyLauncher.java:27)
at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3692)
at org.apache.catalina.core.StandardContext.start(StandardContext.java:4127)
at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:759)
at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:739)
at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:524)
at org.apache.catalina.startup.HostConfig.deployDescriptor(HostConfig.java:603)
at org.apache.catalina.startup.HostConfig.deployApps(HostConfig.java:493)
```

```
    at org.apache.catalina.startup.HostConfig.check(HostConfig.java:1195)
    at sun.reflect.GeneratedMethodAccessor341.invoke(Unknown Source)
```

---

add your notes

MS SQL Server 2000 Startup errors

Please refer to our Knowledge Base article on the topic.

A user reports getting errors like these every time JIRA starts up:

```
[core.entity.jdbc.DatabaseUtil] Entity "Action" has no table in the database
2003-11-06 09:33:45,265 ERROR [core.entity.jdbc.DatabaseUtil] Could not create table "jiraaction"
2003-11-06 09:33:45,265 ERROR [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the
following:
CREATE TABLE jiraaction (ID NUMERIC NOT NULL, issueid NUMERIC, AUTHOR VARCHAR(255), actiontype
VARCHAR(255), actionlevel VARCHAR(255), actionbody TEXT, CREATED DATETIME, actionnum NUMERIC,
CONSTRAINT PK_jiraaction PRIMARY KEY (ID))
Error was: java.sql.SQLException: There is already an object named 'jiraaction' in the database.
2003-11-06 09:33:45,265 WARN [core.entity.jdbc.DatabaseUtil] Entity "ChangeGroup" has no table in the
database
2003-11-06 09:33:45,265 ERROR [core.entity.jdbc.DatabaseUtil] Could not create table "changegroup"
```

JIRA functions correctly otherwise.

A solution is suggested in this jira-user post:

> *Hi Jason,*
>
> *I have had the same errors when at startup that you are seeing.*
>
> *The problem on my server was that when the user in my database ('JIRA')*
> *created tables they were created as dbo.<tablename> and not*
> *JIRA.<tablename>*
> *so when JIRA attempts to verify a table JIRA.<tablename> exists it fails.*
> *Then it tries to create <tablename>, but it already exists. All the*
> *created*
> *tables are owned by 'dbo' and not 'JIRA'.*
>
> *I am running on Microsoft SQL Server so my fix may not fit exactly but this*
> *is what I had to do:*
> *Create the 'JIRA' user as a regular user of the JIRA database. Add the*
> *JIRA user to the db_owner (database owner) role. (DO NOT change the database*
> *owner to 'JIRA', just add the role!)*
>
> *Of course, you will have to drop your existing database first.*
>
> *Cheers,*
> *Bradley.*

We have also had reports from other users that there are also alternatives to this solution. The `ddl_admin`, `db_datareader` and `db_datawriter` roles could be used instead of `db_owner` role for the `jirauser` account.

Setting up your JIRA database for MS SQL Server 2005

On this page:

- Overview
- Before you start
  - 1. Enable network connectivity for SQL Server
  - 2. Configure SQL Server with the appropriate Authentication Mode
  - 3. Disable the 'SET NOCOUNT' option in SQL Server
- Setting up the JIRA database
  - 1. Create a new database

- 2. Create a new database user
- 3. Create a JIRA database schema

Overview

This page supplements the documentation for Connecting JIRA to SQL Server 2005. It provides detailed instructions on setting up your JIRA database for a **straightforward** integration of JIRA with SQL Server 2005. Unfortunately we do not provide support for advanced database configuration, such as hardening or performance tuning. If you require a more complex solution, refer to MS SQL 2005 Documentation and, if necessary, consult with someone in your organisation who is knowledgeable in the configuration of SQL Server 2005.

Before you start

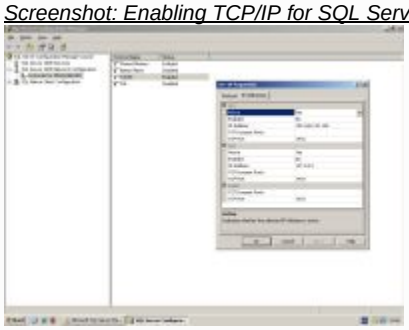
1. Enable network connectivity for SQL Server

Ensure that your instance of SQL Server allows TCP/IP connection and is listing on the default port. Please note that network connectivity is **disabled by default** in some versions of SQL Server (e.g. SQL Server 2005 Express edition). Hence, you will have to enable it, as described below:

To enable TCP/IP for SQL Server,

1. Open the '**SQL Server Configuration Manager**'.
2. Expand '**SQL Server 2005 Network Configuration**' in the console pane.
3. Click '**Protocols for <instance name>**'.
4. The details pane will display (see screenshot below). Right-click '**TCP/IP**' and click '**Enable**'.
5. Click '**SQL Server 2005 Services**' in the console pane.
6. The details pane will display. Right-click '**SQL Server (<instance name>)**' and click '**Restart**' to stop and restart the SQL Server service.

Screenshot: Enabling TCP/IP for SQL Server 2005



2. Configure SQL Server with the appropriate Authentication Mode

Ensure that SQL Server is operating in the appropriate authentication mode. By default, SQL Server operates in '**Windows Authentication Mode** '. However, if your user is not associated with a trusted SQL connection, i.e. '`Microsoft SQL Server, Error: 18452`' is received during JIRA startup, you will need to change the authentication mode to '**Mixed Authentication Mode**'.

Read the Microsoft documentation on authentication modes for instructions on changing the authentication mode.

3. Disable the 'SET NOCOUNT' option in SQL Server

To disable the 'SET NOCOUNT' option in SQL Server,

1. Open the 'SQL Server Management Studio'
2. Navigate to "**Tools' -> 'Options' -> 'Query Execution' -> 'SQL Server' -> 'Advanced'**. The advanced settings for SQL Server will display.
3. Ensure that the '**SET NOCOUNT**' option is **not selected**, as per the screenshot below:

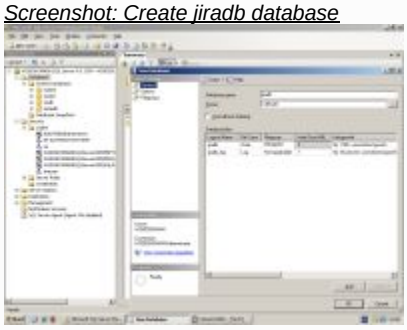
Screenshot: Disabling 'SET NOCOUNT' for SQL Server

Setting up the JIRA database

To set up your JIRA database for SQL Server 2005,

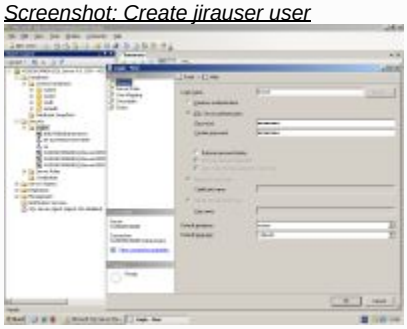
1. Create a new database

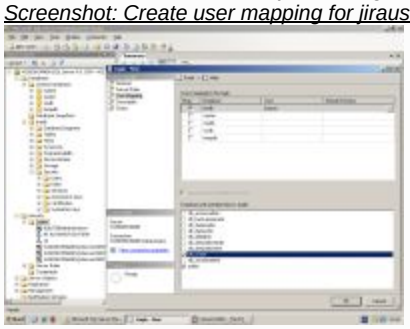
1. Open the 'SQL Server Management Studio'.
2. Connect to the SQL Server that you want to integrate JIRA with. By default this will be '**localhost**'.
3. Navigate to '**<your server name>**' -> '**Databases**' in the left menu of the '**SQL Server Management Studio**'.
4. Right-click '**Databases**' under the server name of your SQL Server and select the '**New Database...**' option from the dropdown menu that appears.
5. The '**New Database**' window will display. Select the '**General**' option in the left menu.
6. The '**General**' page will display (see screenshot below). Enter **jiradb** in the '**Database name**' field.
   Screenshot: Create jiradb database

   
7. Click the '**OK**' button to create the database.

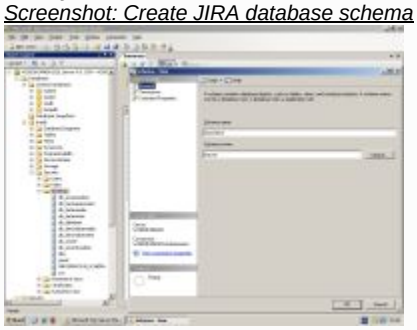
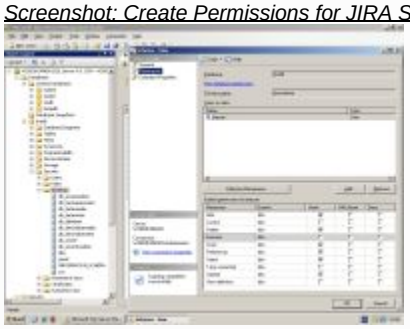
2. Create a new database user

1. Navigate to **'<your server name>' -> 'Security' -> 'Logins'** in the left menu of the 'SQL Server Management Studio'.
2. _Right-click the '**Logins**' folder and select '**New Login**'.
3. The '**Login - New**' window will display. Select the '**General**' option in the left menu.
4. Enter the database user details into the window that displays (see screenshot below), as follows:
    a. Enter '**jirauser**' in the '**Login name**' field.
    b. Select '**SQL Server authentication**'.
    c. Enter '**jirauser**' as the password, and enter '**jirauser**' again in the '**Confirm password**' field.
    d. If you wish to enforce a password policy, check the '**Enforce password policy**' checkbox. However, please be aware that you may need to modify the previously entered password ('jirauser') to meet your password policy rules (e.g. your password policy may require numeric characters in all passwords).
    e. Ensure that the '**Enforce password expiration**' checkbox is unchecked. It will be automatically unchecked and disabled, if you have previously unchecked the '**Enforce password policy**' checkbox.
    f. Ensure that the '**User must change password at next login**' checkbox is unchecked. It will be automatically unchecked and disabled, if you have previously unchecked the '**Enforce password policy**' checkbox.
    Screenshot: Create jirauser user

    
5. Select the '**User Mapping**' option in the left menu.
6. The User Mapping fields for jiradb will display (see screenshot below). Tick the '**jiradb**' checkbox.
7. The '**Database role membership for:jiradb**' panel will display in the bottom panel of the window. Tick the '**db_owner**' checkbox.
8. Click the '**OK**' button to save your changes.
    Screenshot: Create user mapping for jirauser

3. Create a JIRA database schema

1. Navigate to **'<your server name>' -> 'Databases' -> 'jiradb' -> 'Security' -> 'Schemas'** in the left menu of the 'SQL Server Management Studio'.
2. Right-click the '**Schemas**' folder and select '**New Schema**'
3. The 'Schema - New' window will display. Select the '**General**' option in the left menu.
4. The '**General**' page will display (see screenshot below). Fill in the fields, as follows:
    - Enter **jiraschema** in the '**Schema name**' field.
    - Enter **jirauser** in the '**Schema owner**' field.
    Screenshot: Create JIRA database schema
    
5. Select the '**Permissions**' option in the left menu.
6. The '**Permissions**' page will display (see screenshot below). Click the '**Add...**' button.
7. Enter '**jirauser**' in the '**Enter the object names to select (examples):**' field on the pop-up window that displays. Click '**OK**' to save your update and close the pop-up window.
8. Specify the schema permissions in the '**Explicit permission for jirauser**' table on the '**Permissions**' page, as follows:
    - **Alter** — check the '**Grant**' checkbox.
    - **Delete** — check the '**Grant**' checkbox.
    - **Insert** — check the '**Grant**' checkbox.
    - **References** — check the '**Grant**' checkbox.
    - **Select** — check the '**Grant**' checkbox.
    - **Update** — check the '**Grant**' checkbox.
9. Click the '**OK**' button to save your changes.
    Screenshot: Create Permissions for JIRA Schema
    

Congratulations, you have set up a JIRA database for SQL Server 2005. Please refer back to the Connecting JIRA to SQL Server 2005 page to continue integrating SQL Server 2005 with JIRA.

JIRA and MYSQL

This page has general notes on connecting JIRA to MYSQL. It supplements the official MYSQL installation documentation.

- Configuring MySQL 4.1 to store non-ASCII characters
- Data Access Exception - Errcode - 17
- JIRA can't connect to MYSQL with Named Pipes enabled
- MYSQL Administrator and Data Truncation Errors
- Setting up JIRA Standalone and MySQL on Linux

Configuring MySQL 4.1 to store non-ASCII characters

Setting up a MySQL 4.1 database with JIRA to work with non-ASCII (non-English) characters

Please do the following:

1. Create a new MySQL database using the following command:
```
create database jiradb default character set utf8;
```

2. Grant all the required permissions to the JIRA user for the database as described here.
3. Change the JDBC URL in JIRA's datasource definition (conf/server.xml for JIRA Standalone) file to use the new database and be:
   `jdbc:mysql://<your_server>:<port>/jiradb?autoReconnect=true&useUnicode=true&characterEncoding=UTF8`
   Please note the '&amp' XML escape for the ampersands in the url above is needed since it is specified in an xml file.
4. Start JIRA and complete the setup process.

Please ensure that you create a new database using the correct character set and ensure that JIRA creates all its tables on startup without problems. This should allow you to work with all characters supported by Unicode, which covers most characters out there.

Please ensure that you are using the latest MySQL JDBC driver: http://dev.mysql.com/downloads/connector/j/3.1.html

Also please ensure you are using the UTF-8 character encoding in JIRA (Administration -> Global Settings -> General Configuration).

Data Access Exception - Errcode - 17

A user reports of getting this error caused by a MYSQL Bug:

```
com.atlassian.jira.exception.DataAccessException: java.sql.SQLException: Can't create/write to file
'C:\temp2#sql_eb4_0.MYI' (Errcode: 17)
at com.atlassian.jira.upgrade.util.UpgradeUtils.getExactColumnName(UpgradeUtils.java:222)
at
com.atlassian.jira.appconsistency.db.Build178SchemaCheck.isColumnInTable(Build178SchemaCheck.java:81)
at com.atlassian.jira.appconsistency.db.Build178SchemaCheck.check(Build178SchemaCheck.java:71)
at com.atlassian.jira.appconsistency.db.Build178SchemaCheck.isOk(Build178SchemaCheck.java:38)
at com.atlassian.jira.appconsistency.db.DatabaseChecker.checkDatabase(DatabaseChecker.java:108)
at
com.atlassian.jira.appconsistency.db.DatabaseCompatibilityEnforcer.contextInitialized(DatabaseCompatibili
org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3692)
```

where the error code means:

```
C:\>perror 17
OS error code 17: File exists
```

The Workaround:
Disabling their virus checker seemed to resolve the issue of JIRA not coming up. Users should therefore not run "on-access" checking on their Jira servers.

Bug Details:
The bug is described in more detail on the following link MYSQL Bug Forum.

JIRA can't connect to MYSQL with Named Pipes enabled

JIRA can't connect to the database with Named Pipes enabled

```
I've tried a number of things, and it looks like named pipes is the problem. This is a problem with
MySQL, not with JIRA. Essentially I've had to install MySQL with two key things:

    * Go through the Standard Installation route for MySQL, not the Detailed Installation route
    * Enable TCP/IP connections in the MySQL Config Wizard afterwards

After doing this, JIRA now appears to connect to the MySQL and can see the new database
```

Details of what I did to recover MySQL after installing it using named pipes:

MySQL Installation and Config:

- Install MySQL server 5.0 release 5.0.24a (see http://dev.mysql.com/doc/refman/5.0/en/windows-installation.html and http://dev.mysql.com/tech-resources/articles/securing_mysql_windows.html)
- Select typical install
- Configure MySQL with the Configuration Wizard
  o Detailed Configuration
  o Server Machine

    - o Multifunctional Database
    - o Chose C:\<Installation path> for the InnoDB tablespace
    - o Decision Support DSS/OLAP
    - o Disable TCP/IP networking for security and Enable Strict Mode
    - o UTF-8 character set
    - o Install as Windows Service
    - o Include MySQL /bin directory on path - allows mysql* commands to be run directly
    - o Choose root password: ********** Do not allow access from remote machines
- Execute configuration
    - o Config OK - my.ini
    - o Service started - mySQL
    - o Security setting FAILED - error 2017. Can't open named pipe to host: .pipe:mysql(2)
- Create a my.cnf with (client) host=localhost in it, as per http://mysqld.active-venture.com/Windows_vs_Unix.html. Also edit my.ini to have the same line under (client). Don't know if this will work. Named pipes may be a problem.
- Hmm.. There's a suggestion that the Detailed Configuration method just doesn't work (http://forums.mysql.com/read.php?11,80814,93616). If I can't get JIRA to connect to MySQL it might be necessary to re-install the whole thing...
- Create MySQL database and user for JIRA to use. In a command shell run:
    - o mysql -u root
    - o CREATE DATABASE jiradb CHARACTER SET 'utf8';
    - o show databases;
    - o CREATE USER 'jirauser'@'localhost' IDENTIFIED BY '****'; (where **** is jirauser's password)
    - o GRANT ALL PRIVILEGES ON jiradb.* TO 'jirauser'@'localhost';
    - o quit;
    - o mysql -u jirauser -p
    - o <enter password>
    - o show databases;
    - o jiradb is listed as one of the databases
    - o quit;

JIRA configuration to use MySQL:

- Download JDBC driver mysql-connector-java-3.1.12.zip
- Copy the mysql-connector-java-3.1.12-bin.jar file from this zip to C:\Jira\atlassian-jira-professional-3.6.5-standalone\common\lib
- Edit the conf/server.xml file
    - o username and password for the jirauser account set up above
    - o driverClassName="com.mysql.jdbc.Driver"
    - o url="jdbc:mysql://localhost/jiradb?autoReconnect=true&useUnicode=true&characterEncoding=UTF8"
    - o delete the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis parameters
- Edit the entityengine.xml file and change the field-type-name to mysql

Re-start JIRA to use MySQL database

- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\shutdown
    - o Tomcat web-server shuts down
- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\startup
    - o Get error on connection: Unable to establish connection with the database. I suspect this is because the database wasn't set up correctly above and can't open named pipes. This is probably the issue with WinNT-based systems not being able to support named pipes (without modification).
- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\shutdown

Re-installing MySQL without named pipes

- Go to Control Panel->Add/Remove Programs and remove MySQL
- Delete C:\Program Files\MySQL
- Reinstall as above (typical installation)
- Configure and select "Standard Installation"
    - o Install as Windows Service
    - o Add \bin to path
    - o Choose root password **********
- Execute configuration.
    - o Success!
- Run MySQL Config Wizard. Choose options as above.
- Execute configuration
    - o Success!
- Set up MySQL database and jira user as above.
- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\startup
    - o FAILED! Tomcat starts and shuts down immediately. Looking at the logs, it seems that the jirauser account has a & in the password, which stuffs the XML.
- log back in to MySQL as root and run:
    - o DROP USER 'jirauser'@'localhost';
    - o CREATE USER 'jirauser'@'localhost' IDENTIFIED BY '********'; (makeing sure password has no & in it)
    - o GRANT ALL PRIVILEGES ON jiradb.* TO 'jirauser'@'localhost';
    - o quit;
- Edit conf/server.xml to the new password
- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\startup

o No good. Still won't connect.
- Try re-running MySQL config and this time enable TCP/IP connection over port 3306 (so that we don't have to use named pipes)
- run C:\jira\atlassian-jira-professional-3.6.5-standalone\bin\startup
  o Tomcat server starts! Hurray. We appear to connect in some way, although there are lots of exceptions. Maybe these are due to the first start?
- Point web browser at http://localhost:8080/
  o JIRA config screen appears - good
- Go through the JIRA setup and initial configuration steps as above

---

http://confluence.atlassian.com/pages/editpage.action?pageId=133186

MYSQL Administrator and Data Truncation Errors

Hi All,

Due to the release of the 3.7 branch requiring an empty database on startup (see here), a lot of our customers have had issues importing their data into the new install due to encoding inconsistencies between their existing databases and the new ones they've created for 3.7.x.

Errors that users are seeing are Data Truncation errors that look like:

```
org.ofbiz.core.entity.GenericDataSourceException: SQL Exception while executing the following:INSERT
INTO jiraaction (ID, issueid, AUTHOR, actiontype,
actionlevel, rolelevel, actionbody, CREATED, actionnum) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?) (Data
truncation: Data too long for column 'actionbody' at row 1)\
 at org.ofbiz.core.entity.jdbc.SQLProcessor.executeUpdate(SQLProcessor.java:375)
```

We've been told that users using 'MYSQL Administrator' to create their databases lack the ability to specify what encoding type they wish their database to use. Here is the comment we received from a customer:

```
I solved the problem. I used to create the database using the MySQL Adminstrator with the effect, that
I could not define the character-set to use for the database. So I tried to create the database
manually using the command-line tool and bang, the import of the data worked fine.
```

I hope this helps and saves you all some installation hiccups.

-Michelle

Setting up JIRA Standalone and MySQL on Linux

The latest official documentation on configuring JIRA with MySQL can be found here. This is a step-by-step supplement guide to installing the JIRA Standalone distribution with a MySQL database, to replace the default HSQLDB.

Unpack the JIRA Standalone distribution

```
[tmp ~]$ tar zxvf ~/apps/atlassian-jira-professional-3.11-standalone.tar.gz
[tmp ~]$ mv atlassian-jira-professional-3.11-standalone/ jira
[tmp ~]$ cd jira
[jira ~]$
```

Enable MySQL TCP/IP networking

Some Linux distributions (eg. Debian) disable MySQL's TCP/IP networking as a security precaution. You can test that MySQL is listening on the default port (3306) as follows:

```
jturner@teacup:~$ netstat -na | grep 3306
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:48211         127.0.0.1:3306          TIME_WAIT
tcp6       1      0 ::ffff:127.0.0.1:34785  ::ffff:127.0.0.1:3306   CLOSE_WAIT
```

Or if `netstat` isn't available:

```
jturner@teacup:~$ telnet localhost 3306
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
D
5.0.13-rc-Debian_1-lo!X{$:;V#H!ju (press ctrl-] here)
telnet> quit
Connection closed.
```

On Debian, you can enable MySQL TCP connections by editing `/etc/my.cnf`, commenting out the 'skip-networking' flag, and restarting mysqld.

Create MySQL database and user

Create a MySQL user called 'jirauser' and database called 'jiradb':

```
jturner@teacup:~$ mysql --user=root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 559 to server version: 5.0.13-rc-Debian_1-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database jiradb character set utf8;
Query OK, 1 row affected (0.02 sec)

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER,INDEX on
  jiradb.* TO 'jirauser'@'localhost' IDENTIFIED BY 'mypassword';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

The 'IDENTIFIED BY' phrase sets the password for the user (in this case, 'mypassword'). Your hostname may be different; you will find out in the next steps.

Now verify that user 'jirauser' can connect:

```
jturner@teacup:~$ mysql \--user=jirauser \--password=mypassword \--database=jiradb
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 565 to server version: 5.0.13-rc-Debian_1-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

If you get errors like:

```
Access denied for user 'jirauser'@'localhost' (using password: YES)
```

You will need to adjust the 'host' field for the JIRA user record:

```
jturner@teacup:~$ mysql --user=root -p mysql
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 655 to server version: 5.0.13-rc-Debian_1-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select user, host from user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| debian-sys-maint | localhost |
| jirauser         | localhost |
| root             | localhost |
| root             | teacup    |
+------------------+-----------+
4 rows in set (0.00 sec)

mysql> update user set host='localhost.localdomain' where user='jirauser';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> flush privileges;
Query OK, 0 rows affected (0.03 sec)
```

See also Atlassian's MySQL Tips.

If problems persist, see the MySQL Causes of Access Denied Errors page.

For more general information, see Adding New User Accounts to MySQL.

Copy the JDBC driver

```
[jira ~]$ cp ../mysql-connector-java-3.1.12-bin.jar common/lib/
[jira ~]$
```

Customise conf/server.xml

Customise conf/server.xml with MySQL details:

```
[jira ~]$ cp conf/server.xml /tmp
[jira ~]$ vim conf/server.xml
[jira ~]$ diff -u /tmp/server.xml conf/server.xml
--- /tmp/server.xml     2007-02-16 17:09:52.000000000 +1100
+++ conf/server.xml     2007-02-16 17:10:55.000000000 +1100
@@ -11,12 +11,10 @@

         <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
           <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
-            username="sa"
-            password=""
-            driverClassName="org.hsqldb.jdbcDriver"
-            url="jdbc:hsqldb:${catalina.home}/database/jiradb"
-            minEvictableIdleTimeMillis="4000"
-            timeBetweenEvictionRunsMillis="5000"
+            username="jirauser"
+            password="mypassword"
+            driverClassName="com.mysql.jdbc.Driver"
+            url="jdbc:mysql://localhost/jiradb?autoReconnect=true&amp;useUnicode=true
&amp;characterEncoding=UTF8"
+            maxActive="20"
             validationQuery="select 1" />

           <!-- NOTE: When a database server reboots or their is a network failure all the connections
in the
```

The `validationQuery` parameter is required to prevent database connections from dropping out. This has been noted to occur when MySQL is used together with some Tomcat versions. JIRA 3.13 Standalone ships with one of the affected Tomcat versions (5.5.26). See JRA-15731 for further details.

Customise entityengine.xml

```
[jira ~]$ cp atlassian-jira/WEB-INF/classes/entityengine.xml /tmp
[jira ~]$ vim atlassian-jira/WEB-INF/classes/entityengine.xml
[jira ~]$ diff \-u /tmp/entityengine.xml atlassian-jira/WEB-INF/classes/entityengine.xml
--- /tmp/entityengine.xml       2007-02-16 17:11:49.000000000 \+1100
\++\+ atlassian-jira//WEB-INF/classes/entityengine.xml    2007-02-16 17:12:04.000000000 \+1100
@@ \-97,8 \+97,7 @@

PLEASE DO NOT set the use-foreign-key\* values to "true" as JIRA does not currently support this.
\-->
- <datasource name="defaultDS" field-type-name="hsql"
- schema-name="PUBLIC"
+    <datasource name="defaultDS" field-type-name="mysql"
helper-class="org.ofbiz.core.entity.GenericHelperDAO"
check-on-start="true"
use-foreign-keys="false"
```

Start JIRA

```
[jira ~]$ ./bin/startup.sh
....
```

Check for errors

```
Check the logs for any errors:16/02/2007 17:13:43 org.apache.coyote.http11.Http11BaseProtocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
16/02/2007 17:13:43 org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 1181 ms
16/02/2007 17:13:43 org.apache.catalina.realm.JAASRealm setContainer
INFO: Set JAAS app name Catalina
16/02/2007 17:13:43 org.apache.catalina.core.StandardService start
INFO: Starting service Catalina
16/02/2007 17:13:43 org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/5.5.20
16/02/2007 17:13:43 org.apache.catalina.core.StandardHost start
INFO: XML validation disabled
2007-02-16 17:13:48,209 main [core.entity.jdbc.DatabaseUtil] Entity "Action" has no table in the
database
2007-02-16 17:13:48,497 main [core.entity.jdbc.DatabaseUtil] Entity "ChangeGroup" has no table in the
database
2007-02-16 17:13:48,671 main [core.entity.jdbc.DatabaseUtil] Entity "ChangeItem" has no table in the
database
2007-02-16 17:13:49,110 main [core.entity.jdbc.DatabaseUtil] Entity "ColumnLayout" has no table in the
database
2007-02-16 17:13:49,195 main [core.entity.jdbc.DatabaseUtil] Entity "ColumnLayoutItem" has no table in
the database
....
2007-02-16 17:13:50,529 main [core.entity.jdbc.DatabaseUtil] Entity "WorkflowScheme" has no table in
the database
2007-02-16 17:13:50,535 main [core.entity.jdbc.DatabaseUtil] Entity "WorkflowSchemeEntity" has no
table in the database
2007-02-16 17:14:03,413 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Starting JIRA
Enterprise Edition, Version: 3.X-#XXX
2007-02-16 17:14:03,413 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Configured to use
database: mysql
2007-02-16 17:14:03,421 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Issue
Cache Listener, adding.
2007-02-16 17:14:04,532 main WARN [ofbiz.core.entity.SequenceUtil]
[SequenceUtil.SequenceBank.fillBank] first select failed: trying to add row, result set was empty for
sequence: ListenerConfig
2007-02-16 17:14:04,666 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Mail
Listener, adding.
2007-02-16 17:14:04,790 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Mail
Queue Service, adding.
2007-02-16 17:14:04,805 main WARN [ofbiz.core.entity.SequenceUtil]
[SequenceUtil.SequenceBank.fillBank] first select failed: trying to add row, result set was empty for
sequence: ServiceConfig
2007-02-16 17:14:05,034 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Input Language has not
been set.  Setting to 'English'
2007-02-16 17:14:05,038 main WARN [ofbiz.core.entity.SequenceUtil]
[SequenceUtil.SequenceBank.fillBank] first select failed: trying to add row, result set was empty for
sequence: OSPropertyEntry
2007-02-16 17:14:05,124 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl]

\*********************************************************************************************\* J
3.X build: XXX (Enterprise Edition) started. You can now access JIRA through your web browser.
\*********************************************************************************************\*
profiling] Using parameter [jira_profile]
[Filter: profiling] defaulting to off [autostart=false]
[Filter: profiling] Turning filter off [jira_profile=off]
2007-02-16 17:14:07,726 main [webwork.dispatcher.ServletDispatcher] Unable to find
'webwork.multipart.saveDir' property setting. Defaulting to javax.servlet.context.tempdir
16/02/2007 17:14:08 org.apache.coyote.http11.Http11BaseProtocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
16/02/2007 17:14:09 org.apache.catalina.storeconfig.StoreLoader load
INFO: Find registry server-registry.xml at classpath resource
16/02/2007 17:14:09 org.apache.catalina.startup.Catalina start
INFO: Server startup in 25881 ms
```

Again, if you see an 'Access denied' error:

```
Access denied for user 'jirauser'@'localhost.localdomain' (using password: YES)
```

Then you need to adjust your /etc/hosts so that 'localhost' comes before 'localhost.localdomain', and restart MySQL. This is a MySQL bug fixed in 5.0.11.

Run the Setup Wizard

Point a browser at http://localhost:8080/, and set up JIRA, as described in the Setup Wizard.

Troubleshooting

Q: I get the following error message in MySQL, "`Attempted reconnect 3 times. Giving up.`" What should I do?
A:

```
jdbc:mysql://localhost/test?autoReconnect=true connection error :
Server connection failure during transaction.
Attempted reconnect 3 times. Giving up.
```

MySQL error message

To troubleshoot your MySQL connection, please follow the steps below:
1. Enter the following command to connect to MySQL:

```
mysql -p -u [dbuser] -h 127.0.0.1 [dbname]
```

For example,

```
mysql -p -u mydbuser -h 127.0.0.1 test
```

2. If you cannot connect to MySQL after entering your password, login to your mysql with the root account:

```
mysql -p -u root
```

And enter following command:

```
mysql> GRANT ALL PRIVILEGES ON <dbname>.* to <user>@127.0.0.1 identified by '<password>';
mysql> FLUSH PRIVILEGES;
```

where,
`<dbname>` is your database name,
`<user>` is your database user name,
`<password>` is your database password,

Don't forget the last command: '`FLUSH PRIVILEGES`'

3. If you still cannot connect, please check that your MySQL is listening on the default port of `3306` and bind in your IP, `127.0.0.1` by running either of the following commands:

```
netstat -a |grep mysql
```

or,

```
netstat -a |grep 3306
```

If MySQL is listening, you should see the following message:

```
tcp 0 0 *:mysql *:* LISTEN
```

Alternatively, you also could check if your MySQL is listening on the default port by running this command:

```
telnet 127.0.0.1 3306
```

JIRA and Oracle

This page has general notes on connecting JIRA to Oracle. It supplements the official Oracle installation documentation.

- Configuring Datasource for Oracle 10g JDBC drivers
- Restoring data using I-Net (Oranxo) Driver for Oracle
- Store Workflow on Disk with Oracle 8

Configuring Datasource for Oracle 10g JDBC drivers

When using JIRA with Oracle, the Oracle 10g JDBC driver needs to have the **SetBigStringTryClob** property set to true to store text of unlimited size in the database. If this property is not set, you will have problems modifying JIRA workflows and storing large (over 32k) text strings.

The **SetBigStringTryClob** property needs to be set in the application server, where the database connection is defined (the 'datasource' definition). The definition depends on the application server that you are using. Please refer to one of the sections below that is applicable to your application server to determine what to add to the datasource definition.

The same thing applies to I-Net's JDBC driver, except the property is called **streamstolob**.

Tomcat 4 and 5.0

Add the section shown in bold to where JIRA is configured (usually **conf/server.xml**):

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"/>
    <ResourceParams name="jdbc/JiraDS">
      <parameter>
        <name>driverClassName</name>
        <value>oracle.jdbc.driver.OracleDriver</value>
      </parameter>
      <parameter>
        <name>url</name>
        <value>jdbc:oracle:thin:@<database host machine>:<port>:<SID></value>
      </parameter>
      <parameter>
        <name>username</name>
        <value>...</value>
      </parameter>
      <parameter>
        <name>password</name>
        <value>...</value>
      </parameter>

      <parameter>
         <name>connectionProperties</name>
         <value>SetBigStringTryClob=true</value>
      </parameter>

      <parameter>
        <name>factory</name>
        <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
    </ResourceParams>
```

Tomcat 5.5 (JIRA Standalone)

Modify the section where JIRA is configured, usually **conf/Catalina/localhost/jira.xml** or **conf/server.xml** (in JIRA Standalone):

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.11.war">
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
    username="...."
    password="...."
    driverClassName="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@<database host machine>:<port>:<SID>"
    connectionProperties="SetBigStringTryClob=true" />
  <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
    factory="org.objectweb.jotm.UserTransactionFactory"
    jotm.timeout="60"/>
  <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>
</Context>
```

Orion / OC4J (Oracle iAS)

Orion and OC4J are not capable of passing arbitrary properties directly to the JDBC driver. These app servers can pass properties to the underlying Data Source if it has appropriate "setter" methods for them. Unfortunately the Data Sources that Oracle ships with its JDBC drivers do not have an appropriate "setter" method for the **SetBigStringTryClob** property.

To get around this problem please use the **com.atlassian.oracle.OracleConnectionPoolProxyDataSource** which is available from the **atlassian-oracle-orion-compat** jar file. To use this Data Source please follow these instructions:

1. Download the *latest* `atlassian-oracle-orion-compat` from our repository. (For example, the atlassian-oracle-orion-compat-1.0.jar file).
2. Put the downloaded jar file into the same directory as the ojdbc14.jar file. For Orion, this is the lib directory. For OC4J the directory is called applib.
3. Edit the `config/data-sources.xml` file and specify the data source as follows:

```
<data-source
class="com.atlassian.oracle.OracleConnectionPoolProxyDataSource"
name="<name>"
location="<location>"
xa-location="<xa-location>"
ejb-location="<ejb-location>"
url="<url>"
username="<login>"
password="<password>"
inactivity-timeout="30"
>
<property name="setBigStringTryClob" value="true" />
</data-source>
```

Note the small '**s**' in **setBigStringTryClob** here -- this fix *will not work* otherwise. Also note that this small 's' *only* applies to Orion/OC4J. All other app servers need a big 'S'.

OC4J 10.1.3.x

OC4J 10.1.3.x no longer uses the `<data-source>` element, and we are not aware of another way of passing through the **setBigStringTryClob** property. See JRA-12564 for details.

This means that you cannot store text strings longer than 32K in Oracle for OC4J 10.1.3.x.

Other app servers

Consult the relevant app server documentation to find how to add the property.

Restoring data using I-Net (Oranxo) Driver for Oracle

When restoring data into an Oracle 9 database using the I-Net Oranxo Driver a user was seeing this error message in their logs:

```
org.ofbiz.core.entity.GenericDataSourceException: SQL Exception while executing the following:INSERT
INTO jiraaction (ID, issueid, AUTHOR, actiontype, actionlevel, actionbody, CREATED, actionnum) VALUES
(?, ?, ?, ?, ?, ?, ?, ?) ([OraDriver] #7 Unexpected end of inputstream in header.)
 at org.ofbiz.core.entity.jdbc.SQLProcessor.executeUpdate(SQLProcessor.java:375)
 at org.ofbiz.core.entity.GenericDAO.singleInsert(GenericDAO.java:115)
 at org.ofbiz.core.entity.GenericDAO.insert(GenericDAO.java:88)
 at org.ofbiz.core.entity.GenericHelperDAO.create(GenericHelperDAO.java:63)
 at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:470)
 at org.ofbiz.core.entity.GenericDelegator.create(GenericDelegator.java:450)
 at org.ofbiz.core.entity.GenericValue.create(GenericValue.java:77)
 at com.atlassian.jira.action.admin.ImportParser$1.run(ImportParser.java:191)
 at EDU.oswego.cs.dl.util.concurrent.PooledExecutor$Worker.run(PooledExecutor.java:751)
 at java.lang.Thread.run(Thread.java:595)
com.inet.ora.Ora3SQLException: [OraDriver] #7 Unexpected end of inputstream in header.
 at com.inet.ora.Ora3Factory.createSQLException(Unknown Source)
 at com.inet.ora.Ora3Factory.createSQLException(Unknown Source)
 at com.inet.ora.OraConnection.c(Unknown Source)
 at com.inet.ora.OraConnection.a(Unknown Source)
 at com.inet.ora.OraPreparedStatement.a(Unknown Source)
 at com.inet.ora.OraPreparedStatement.a(Unknown Source)
 at com.inet.ora.OraPreparedStatement.executeUpdate(Unknown Source)
```

Fix

This error was fixed by changing the **Set Clob** entry in the `server.xml` if using Standalone or the `jira.xml` file if using the WebApp edition of JIRA.

So instead of using:

```
connectionProperties="SetBigStringTryClob=true"
```

for this driver it needs to be replaced with:

```
connectionProperties="streamstolob=true"
```

As per this document on Oracle JDBC Manual

Store Workflow on Disk with Oracle 8

Atlassian support for Oracle 8 officially ends with the JIRA 3.6 release. Oracle 8 users are advised to upgrade to avoid the problem described on this page.

Applies to:	Oracle 8 users

Oracle has a 4000 character limitation on its VARCHAR2 field type. This causes problems for JIRA, which uses VARCHAR2 to store comments and 'workflows'. Whenever a comment or workflow exceeds 4000 characters (very easy in the case of workflows), JIRA breaks.

Oracle have a workaround for this problem in their 10g JDBC driver, which can be used with Oracle 9 and 10. Use of this workaround has been incorporated into the documentation.

This doesn't help Oracle 8 users. A workaround for the problem of > 4000 character workflows in Oracle 8 is to store these on disk, instead of in the database. This can be done as follows:

1. Run JIRA Standalone (with hsqldb database) to construct the workflow, and then:
2. Export the created workflow as XML, and save this to disk, eg custom-workflow.xml
3. In the JIRA instance that will use Oracle, edit WEB-INF/classes/workflows.xml and add a line:

```
<workflow name="custom" type="resource" location="custom-workflow.xml"/>
```

Where 'custom' is the workflow name.

4. Copy custom-workflow.xml to WEB-INF/classes/
5. Restart JIRA. The 'custom' workflow should appear in the list of available workflows.

JIRA and PostgreSQL

This page has general notes on connecting JIRA to Postgres. It supplements the official Postgres installation documentation.

- Setting up JIRA Standalone and PostgreSQL on Linux

Setting up JIRA Standalone and PostgreSQL on Linux

This is a step-by-step guide to getting JIRA running with PostgreSQL, on Ubuntu. The information should be helpful for other Linux distributions too.

Sun Java

Make sure that you have got Sun's Java Development Kit (JDK) installed. Ubuntu and Debian come with a GCJ variant which *will not work*. See Installing Java on Ubuntu or Debian.

Creating a jira system account

It's a good idea to create a dedicated user to run JIRA (as with all server software):

```
jturner:~$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

Set up a PostgreSQL user

PostgreSQL is very easy to set up on Ubuntu:

```
jturner:~$ sudo apt-get install postgresql-8.2 postgresql-client-8.2
Reading package lists... Done
Building dependency tree
....
 * Starting PostgreSQL 8.2 database server             [ OK ]
```

Now we create a `jira` PostgreSQL user for the `jira` user to connect as:

```
jturner:~$ sudo su - postgres
postgres:~$ createuser -P jira
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) n
CREATE ROLE
```

We can now connect as our `jira` user and create a database. It's a good idea to version your databases so upgrading is safer, so here I've added `_312` (indicating JIRA 3.12) to the database name:

```
postgres:~$ logout
jturner:~$ sudo su - jira
jira:~$ createdb jira_312
CREATE DATABASE
```

Configuring JIRA Standalone

Download JIRA Standalone and unpack it:

```
jturner:~$ sudo su - jira
jira:~$ tar zxvf /home/jturner/Desktop/atlassian-jira-enterprise-3.12-standalone.tar.gz > /dev/null
jira:~$ ln -s atlassian-jira-enterprise-3.12-standalone/ jira-3.12
jira:~$ cd jira-3.12
jira:~/jira-3.12$
```

Now download the PostgreSQL JDBC driver from the website and place it in JIRA's `common/lib` directory.. As I installed PostgreSQL 8.2 and have JDK 1.6, I got the JDBC4 version:

jira:~/jira-3.12$ cd common/lib/
jira:~/jira-3.12/common/lib$ wget -q 'http://jdbc.postgresql.org/download/postgresql-8.2-507.jdbc4.jar'
jira:~/jira-3.12/common/lib$ ls -l postgresql-8.2-507.jdbc4.jar
~~rw-rr-~~ 1 jira jira 457436 2007-12-02 19:42 postgresql-8.2-507.jdbc4.jar
jira:~/jira-3.12/common/lib$

We can now configure the JDBC datasource:

```
jira:~/jira-3.12/common/lib$ cd ../../conf
jira:~/jira-3.12/conf$ cp server.xml server.xml.orig
jira:~/jira-3.12/conf$ vi server.xml
jira:~/jira-3.12/conf$ diff -u server.xml.orig server.xml
--- server.xml.orig     2007-12-16 10:50:06.000000000 +1100
+++ server.xml  2007-12-16 10:51:03.000000000 +1100
@@ -11,12 +11,10 @@

         <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
           <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
-            username="sa"
-            password=""
-            driverClassName="org.hsqldb.jdbcDriver"
-            url="jdbc:hsqldb:${catalina.home}/database/jiradb"
-            minEvictableIdleTimeMillis="4000"
-            timeBetweenEvictionRunsMillis="5000"
+            username="jira"
+            password="jira"
+            driverClassName="org.postgresql.Driver"
+            url="jdbc:postgresql://localhost/jira_312"
             maxActive="20" />

           <!-- NOTE: When a database server reboots or their is a network failure all the connections
in the
```

Configure JIRA to know what type of database it's connecting to:

```
jira:~/jira-3.12/conf$ cd ../atlassian-jira/WEB-INF/classes/
jira:~/jira-3.12/atlassian-jira/WEB-INF/classes$ cp entityengine.xml entityengine.xml.orig
jira:~/jira-3.12/atlassian-jira/WEB-INF/classes$ vim entityengine.xml
jira:~/jira-3.12/atlassian-jira/WEB-INF/classes$ diff -u entityengine.xml.orig entityengine.xml
--- entityengine.xml.orig       2007-12-16 10:52:25.000000000 +1100
+++ entityengine.xml    2007-12-16 10:53:00.000000000 +1100
@@ -97,8 +97,8 @@

         PLEASE DO NOT set the use-foreign-key* values to "true" as JIRA does not currently support
this.
       -->
-    <datasource name="defaultDS" field-type-name="hsql"
-      schema-name="PUBLIC"
+    <datasource name="defaultDS" field-type-name="postgres72"
+      schema-name="public"
       helper-class="org.ofbiz.core.entity.GenericHelperDAO"
       check-on-start="true"
       use-foreign-keys="false"
jira:~/jira-3.12/atlassian-jira/WEB-INF/classes$
```

Starting JIRA

Everything is now configured, and we can start JIRA.

```
jira:~/jira-3.12$ ./bin/startup.sh
If you encounter issues starting up JIRA Standalone Edition, please see the Troubleshooting guide at
http://confluence.atlassian.com/display/JIRA/Installation+Troubleshooting+Guide
Using CATALINA_BASE:   /usr/local/jira/jira-3.12
Using CATALINA_HOME:   /usr/local/jira/jira-3.12
Using CATALINA_TMPDIR: /usr/local/jira/jira-3.12/temp
Using JRE_HOME:       /usr/lib/jvm/java-6-sun
jira:~/jira-3.12$
```

No JAVA_HOME?

If you get the error:

```
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program
```

You'll need to install the Sun JVM and set JAVA_HOME.

To see whether JIRA is starting properly, you should 'tail' the logs with:

```
jira:~/jira-3.12$ tail -f logs/catalina.out
2007-12-16 11:51:22,912 main [core.entity.jdbc.DatabaseUtil] Entity "Version" has no table in the
database
2007-12-16 11:51:22,919 main [core.entity.jdbc.DatabaseUtil] Entity "VersionControl" has no table in
the database
2007-12-16 11:51:22,923 main [core.entity.jdbc.DatabaseUtil] Entity "Workflow" has no table in the
database
2007-12-16 11:51:22,931 main [core.entity.jdbc.DatabaseUtil] Entity "WorkflowScheme" has no table in
the database
2007-12-16 11:51:22,936 main [core.entity.jdbc.DatabaseUtil] Entity "WorkflowSchemeEntity" has no
table in the database
2007-12-16 11:51:22,943 main [core.entity.jdbc.DatabaseUtil] Entity "Worklog" has no table in the
database
2007-12-16 11:51:25,653 main ERROR [jira.ext.fisheye.FishEyeConfigImpl] Invalid FishEye configuration:
FishEye Url must be specified.
2007-12-16 11:51:25,667 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Starting JIRA
Enterprise Edition, Version: 3.12-#298
2007-12-16 11:51:25,667 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Configured to use
database: postgres72
2007-12-16 11:51:25,669 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Issue
Cache Listener, adding.
2007-12-16 11:51:26,032 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Mail
Listener, adding.
2007-12-16 11:51:26,038 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Could not find Mail
Queue Service, adding.
2007-12-16 11:51:26,053 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl] Input Language has not
been set.  Setting to 'English'
2007-12-16 11:51:26,071 main INFO [atlassian.jira.upgrade.ConsistencyCheckImpl]

*********************************************************************************************
JIRA 3.12 build: 298 (Enterprise Edition) started. You can now access JIRA through your web browser.
*********************************************************************************************

[Filter: profiling] Using parameter [jira_profile]
[Filter: profiling] defaulting to off [autostart=false]
[Filter: profiling] Turning filter off [jira_profile=off]
2007-12-16 11:51:26,487 main [webwork.dispatcher.ServletDispatcher] Unable to find
'webwork.multipart.saveDir' property setting. Defaulting to javax.servlet.context.tempdir
16/12/2007 11:51:26 org.apache.coyote.http11.Http11BaseProtocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
16/12/2007 11:51:26 org.apache.catalina.storeconfig.StoreLoader load
INFO: Find registry server-registry.xml at classpath resource
16/12/2007 11:51:26 org.apache.catalina.startup.Catalina start
INFO: Server startup in 5761 ms
```

The `Entity "..." has no table in the database` messages indicate that JIRA has found an empty database and is creating the necessary tables. Ie. the connection to PostgreSQL is working.

Congratulations, you have got JIRA installed. You can see it running at http://localhost:8080, where you can follow the setup wizard to complete the configuration.

JIRA and Sybase

This page has general notes on connecting JIRA to Sybase. It supplements the official Sybase installation documentation.

Connecting JIRA to a different database than the one provided as default

Please see Connecting JIRA to a Database and Switching Databases.

Database Disappears

This problem has been fixed in JIRA 3.3 and above

Symptoms

Occasionally people start JIRA standalone up to find that their database has disappeared, and they are asked to go through the setup wizard again.

Solution

The problem is that the path to the HSQL database (that ships with JIRA standalone) is relative to where you started the application.

So this command:

```
c:\jira\bin\> startup.bat
```

Will create a database in a different directory to:

```
c:\jira\> bin\startup.bat
```

The solution is to shut down JIRA and start JIRA up from the normal location.

Explanation

HSQL's database location is specified at a relative location. You can see this in `server.xml` in Tomcat:

```
<parameter>
<name>url</name>
<value>jdbc:hsqldb:../database/tomcatdb</value
</parameter>
```

As JIRA does not know where it will be installed, we cannot specify an absolute directory. In Tomcat 4, there is no way to get the TOMCAT_HOME variable from inside the JVM either.

This should be fixed when we upgrade JIRA standalone to Tomcat 5.

Incorrect database type specified

Background

JIRA needs to know what kind of database it will be using, in order to generate database tables of the correct data types, and to generate correctly formatted SQL. The database type is specified in **atlassian-jira/WEB-INF/classes/entityengine.xml** (JIRA Standalone), or **edit-webapp/WEB-INF/classes/entityengine.xml** (JIRA WAR/Webapp):

```
...
    <datasource name="defaultDS" field-type-name="hsql"
        helper-class="org.ofbiz.core.entity.GenericHelperDAO"
        check-on-start="true"
        use-foreign-keys="false"
        use-foreign-key-indices="false"
        ...
```

In this example, JIRA expects to use HSQLDB (the default for JIRA Standalone).

If you've got it wrong ...

If you forgot to edit the `entityengine.xml` file (see the documentation), then follow these steps to recover:

1. Fix the type in entityengine.xml

- If you are using **JIRA Standalone**, edit **atlassian-jira/WEB-INF/classes/entityengine.xml**, and correct the database type (valid types are listed in entityengine.xml). See the Using External Databases with JIRA Standalone guide to check if you haven't made other mistakes.
- If you have deployed JIRA as a Webapp into your own app server, you need to:
    - Edit **edit-webapp/WEB-INF/classes/entityengine.xml** and fix the type, as above.
    - Run build.bat or build.sh in the root, to rebuild the JIRA webapp.
    - Deploy the rebuilt webapp into your app server.

2. Fix the database

Is this the first time you have run JIRA?

If so, the **database has been created incorrectly**. Specifically, table columns have been created with incorrect data types, and you will see warnings like these in the logs:

```
2005-02-10 12:24:33,307 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE jiraaction (ID BIGINT NOT NULL, issueid BIGINT, AUTHOR VARCHAR, actiontype VARCHAR,
actionlevel
  VARCHAR, actionbody VARCHAR, CREATED TIMESTAMP, actionnum BIGINT, CONSTRAINT PK_jiraaction PRIMARY
KEY (ID))
Error was: java.sql.SQLException: You have an error in your SQL syntax.  Check the manual that
corresponds
  to your MySQL server version for the right syntax to use near ' actiontype VARCHAR, actionlevel
VARCHAR, actionbody VARCHAR, C
2005-02-10 12:24:33,308 [core.entity.jdbc.DatabaseUtil] Entity "ChangeGroup" has no table in the
database
2005-02-10 12:24:33,309 [core.entity.jdbc.DatabaseUtil] Could not create table "changegroup"
2005-02-10 12:24:33,309 [core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:
CREATE TABLE changegroup (ID BIGINT NOT NULL, issueid BIGINT, AUTHOR VARCHAR, CREATED TIMESTAMP,
CONSTRAINT
  PK_changegroup PRIMARY KEY (ID))
Error was: java.sql.SQLException: You have an error in your SQL syntax.  Check the manual that
corresponds to
  your MySQL server version for the right syntax to use near ' CREATED TIMESTAMP, CONSTRAINT
PK_changegroup PRIMARY KEY (ID))
2005-02-10 12:24:33,310 [core.entity.jdbc.DatabaseUtil] Entity "ChangeItem" has no table in the
database
2005-02-10 12:24:33,310 [core.entity.jdbc.DatabaseUtil] Could not create table "changeitem"
```

The solution is to drop (delete) and recreate the database. When next restarted with the correct data types, JIRA will recreate the tables correctly.

Upgrading JIRA?

This situation is potentially problematic, because the newer version of JIRA may have added tables or columns with incorrect data types to your existing database schema.

The safest solution is to start a new database, and import an XML backup made before the upgrade.

If for some reason, you cannot import an XML backup (eg. your upgraded instance has been in production for a few days and contains new data), it is generally possible to patch the database by hand with SQL '`alter table`' statements. Please review the log files for information on what types JIRA expects, and what is actually present. JIRA will print this information every time it starts up. If in doubt, attach the logs and other relevant information to a support request on our support system.

Other situations

If this is not the first time JIRA has loaded, and you are not upgrading, you probably do not need to fix the data. After fixing the `entityengine.xml` file, restart and check the logs for errors. If there are none, the database is fine.

Need help?

Please create a support request and attach the startup logs, your current `entityengine.xml` file, and any other information relevant.

Making databases read-only for upgrade backups

For medium to large databases, backing up JIRA data can take a long time, during which it's possible for users and JIRA services to alter the database, possibly resulting in inconsistent data in your backup.

For automated daily backups, there is no solution to this, and it's a good reason to prefer native database backup utilities, as the documentation suggests.

For upgrading, though, you need to generate an XML backup, and this problem needs to be solved to ensure a consistent backup.

One way to ensure a consistent backup is to shut down JIRA, start it on a private port and do the backup there, knowing that there are no existing or new users doing things. This causes downtime though.

A simpler approach is to make JIRA read-only for the duration of the backup, just before you take it down for the upgrade. The process would be:

- Set a banner warning users that JIRA is going read-only before an upgrade.
- In the database, remove write permissions from the database user JIRA uses.

- Generate an XML backup.
- Shut down JIRA.
- Proceed with the upgrade (start the new JIRA pointing at a new empty database, and import the XML backup).

Users are still able to search and view issues, but creating, editing or commenting on an issue will fail with an error. This is admittedly a bit ugly, but setting a banner explaining the situation ameliorates this.

The process for making a database read-only for a particular user is very database-specific.

PostgreSQL

Log in to the JIRA database with `psql`, and run:

```
REVOKE INSERT, DELETE, UPDATE on project,jiraaction,jiraissue from jira;
```

where `jira` is the Postgres user JIRA uses.

For instance, on a linux system this might look as follows:

```
jturner@psyche:~$ sudo su - postgres
Password:
postgres@psyche:~$ psql jiradb_391
Welcome to psql 8.2.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

jiradb_391=# \du
                             List of roles
 Role name | Superuser | Create role | Create DB | Connections | Member of
-----------+-----------+-------------+-----------+-------------+-----------
 jira      | no        | no          | no        | no limit    |
 jturner   | yes       | yes         | yes       | no limit    |
 postgres  | yes       | yes         | yes       | no limit    |
(3 rows)

jiradb_391=# REVOKE INSERT, DELETE, UPDATE on project,jiraaction,jiraissue from jira;
REVOKE
jiradb_391=#
```

MySQL

This is very similar to PostgreSQL, except that specifying the user has to include a hostname component. First find the relevant username and host:

```
jturner@psyche:~$ mysql -u root -p mysql
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 303
Server version: 5.0.38-Ubuntu_0ubuntu1-log Ubuntu 7.04 distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select user,host from user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| bugzilla         | %         |
| debian-sys-maint | %         |
| jirauser         | %         |
| root             | %         |
| bugzilla         | localhost |
| debian-sys-maint | localhost |
| jira             | localhost |
| root             | teacup    |
+------------------+-----------+
8 rows in set (0.00 sec)
```

The relevant user is 'jira', valid from host 'localhost'. Now run:

```
mysql> REVOKE INSERT,DELETE,UPDATE on jiradb_391.* from 'jira'@'localhost';
Query OK, 0 rows affected (0.01 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.02 sec)
```

Unlike PostgreSQL, you will need to restart JIRA for this change to take effect.

Other databases.

To help grow this page, please add a description in a comment if you have successfully done this on another database.

Restarting from Setup Wizard in JIRA Standalone

If you ever want to start again from the Setup Wizard in JIRA Standalone, this can be done as follows:

1. Stop JIRA (run bin\shutdown.bat)
2. Delete the _database_ directory in the JIRA root.
3. Start JIRA, and point your browser at any JIRA page.

JIRA will detect that no database is present, and will take you through the setup wizard again.

Surviving Connection Closures

When a database server reboots, or there is a network failure, all the connections in the connection pool are broken and this normally requires a Application Server reboot.

However, the Commons DBCP (Database Connection Pool) which is used by the Tomcat application server (and hence JIRA Standalone) can validate connections before issuing them by running a simple SQL query, and if a broken connection is detected, a new one is created to replace it. To do this, you will need to set the "validationQuery" option on the database connection pool.

Performance Considerations
Please note, that setting the `validationQuery` option on the database connection pool will have a performance impact. The overall decrease in performance should be minimal, as the query itself is quick to run. In addition, the query will only execute when you make a connection. Thus, if you keep the connection for the duration of a request, the query will only occur once per request.
You may wish to assess the performance impact of this change before implementing it, if you are running a large JIRA instance.

Determining the Validation Query

Each database has slightly different SQL syntax. The Validation Query should be as simple as possible, as this is run every time a connection is retrieved from the pool.
Some examples are:

Database	Validation Query
MySQL	Select 1
MS SQL Server	Select 1
Oracle	select 'validationQuery' from dual

Setting the validationQuery parameter

In your application server, where the JDBC DataSource is configured, a parameter needs to be added to tell the Connection Pool to use a validation query (determined above) to validate connections.

JIRA Standalone (after version 3.2) or JIRA EAR/WAR on Tomcat 5.5 and Tomcat 6.0

If you are using JIRA Standalone, edit `conf/server.xml`
If you are using JIRA EAR/WAR on Tomcat 5.5 or Tomcat 6.0, edit `conf/Catalina/localhost/jira.xml`

Locate the section where the '`jdbc/JiraDS`' DataSource is set up, and add the following:

- If you are using **MySQL** or **MS SQL Server**, add `validationQuery="select 1"`
- If you are using **Oracle**, add `validationQuery="select 'validationQuery' from dual"`

For example (for MySQL):

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/jiradb?useUnicode=true&amp;characterEncoding=UTF8"
    username="jirauser"
    password="jirapassword"
    maxActive="20"
    validationQuery="select 1" />
```

Tomcat 4 and 5.0:

Edit `conf/server.xml` (Tomcat 4) or `conf/Catalina/localhost/jira.xml` (Tomcat 5.0), locate the section where the '`jdbc/JiraDS`' DataSource is set up, and add the following:

- If you are using **MySQL** or **MS SQL Server**, add

```
<parameter>
<name>validationQuery</name>
<value>select 1</value>
</parameter>
```

- If you are using **Oracle**, add

```
<parameter>
<name>validationQuery</name>
<value>select 'validationQuery' from dual</value>
</parameter>
```

For example:

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/JiraDS">
<parameter>
<name>driverClassName</name>
<value>oracle.jdbc.driver.OracleDriver</value>
</parameter>
<parameter>
<name>url</name>
<value>jdbc:oracle:thin:@<database host machine>:<port>:<SID></value>
</parameter>
<parameter>
<name>username</name>
<value>...</value>
</parameter>
<parameter>
<name>password</name>
<value>...</value>
</parameter>

<parameter>
<name>validationQuery</name>
<value>select 1</value>
</parameter>

<parameter>
<name>factory</name>
<value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
</parameter>
</ResourceParams>
```

Orion / OC4J

For Orion/OC4J, edit `config/data-sources.xml`, and add the property as a nested tag:

```
<data-source
class="<datasource driver class>"
name="<name>"
location="<location>"
xa-location="<xa-location>"
ejb-location="<ejb-location>"
url="<url>"
connection-driver="<driver>"
username="<login>"
password="<password>"
inactivity-timeout="30"
>
<property name="validationQuery" value="Select 1" />
</data-source>
```

Other app servers

Consult the relevant JIRA app server guide and the app server documentation to find how to add the property.

Results

You should now be able to survive a complete loss of all connections and be able to recover without rebooting the your App Server.

Configure JIRA as service on Mac OS X

If you want to run JIRA as a server on OSX, you will need to configure it to load as a userdaemon. OSX has migrated configuration scripts from services such as cron, rc, or init.d to the launchd utility. There are some good introductory and in-depth explanations of it's function on the web. You can find out more about launchd here:

http://developer.apple.com/macosx/launchd.html

It's easier to use Lingon (http://lingon.sourceforge.net/) to define your plist xml definitions for import into launchd; although because launchd does not permit forking of processes you will need to call $TOMCAT_HOME/bin/catalina.sh directly.

Example definition of com.atlassian.jira.plist:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
         <key>Label</key>
         <string>com.atlassian.jira</string>
         <key>ProgramArguments</key>
         <array>

<string>/usr/local/apache-tomcat-5.5.23/bin/catalina.sh</string>
                 <string>run</string>
         </array>
         <key>RunAtLoad</key>
         <true/>
         <key>ServiceDescription</key>
         <string>JIRA autoloaded as a service</string>
         <key>UserName</key>
         <string>pw</string>
</dict>
</plist>
```

I'd advise adding the <UserName> tag to tell OSX which user to run Tomcat under, eg running Tomcat under root is not recommended. JAVA_HOME will be inherited as an environment variable for whichever user you define in the XML definition - so configure it for that user's .profile in their home directory.

If you wanted to run JIRA as a WAR web-archive, and use OSX's factory install of Tomcat, please see the JIRA WAR installation instructions.

Configuring IIS with Tomcat

It is possible to set this up rather painlessly and the main documentation that we provide covers most use cases. But sometimes there are a few IIS configurations that differ from the default.

You are not using the "Default Website" in IIS

If you are in this boat, you will need to mimic what the Jakarta ISAPI installer created for you in the default website.

There will need to be two virtual directories:

One named 'jira'

Follow these steps to setup the 'jira' virtual directory

1. Right-click on the website, go to New > Virtual Directory ...
2. The alias needs to be called **jira**
3. The path can point to any location, a temp directory, or perhaps your jira install location
4. Check the "Execute (Such as ISAPI application of CGI)", then next and you are now finished.

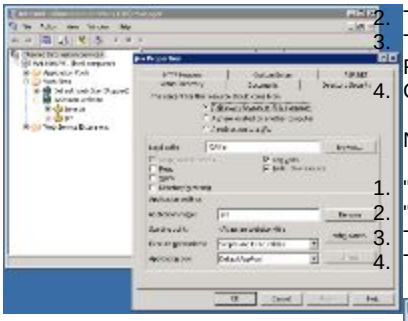
Now in the properties for the 'jira' virtual directory confirm:

1. The Execute Permissions is set to "Scripts and Executables" in the "Execute Permissions" section.

One named 'jakarta'

Follow these steps to setup the 'jakarta' virtual directory

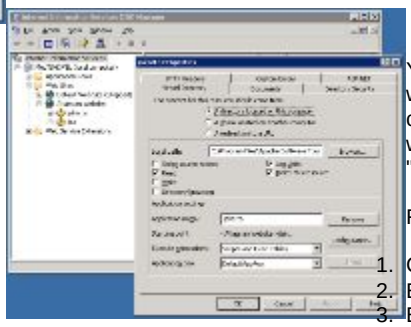
1. Right-click on the website, go to New > Virtual Directory ...



2. The alias needs to be called **jakarta**
3. The path needs to point to the bin directory of the Jakarta Isapi Redirector, ie: C:\Program Files\Apache Software Foundation\Jakarta Isapi Redirector\bin
4. Check the "Execute (Such as ISAPI application of CGI)", then next and you are now finished.

Now in the properties for the 'jakarta' virtual directory confirm:

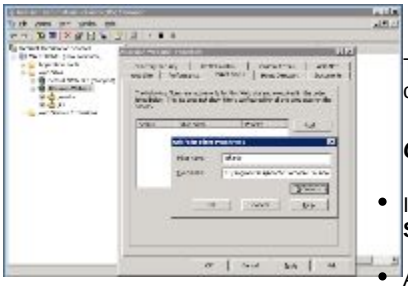
1. "Script source access" is checked
2. "Read" access is checked
3. The Execute Permissions is set to "Scripts and Executables"
4. The "Local Path" points to the bin directory of the Jakarta Isapi Redirector



You will also need to make sure that the non-default website has the Jakarta Redirector installed. This can be done by right-clicking on the non-default website, clicking 'properties' and then clicking on the "ISAPI Filters" tab.

From here you will need too:

1. Click the "Add..." button
2. Enter a filter name: **jakarta**
3. Browse to the "isapi_redirect.dll" file located here: C:\Program Files\Apache Software Foundation\Jakarta Isapi Redirector\bin
4. Click OK, Apply and then OK.



The final step is to restart the IIS Server, this can be done by opening the services.msc and clicking restart on "World Wide Web Publishing"

Gotcha's

- If you are using IIS 6.0 did you remember to add the *Jakarta Isapi Redirector* to the **Web Service Extension's** and set the extension status to *allow* ?
- Also for IIS 6.0 did you remember to add the *Jakarta Isapi Redirector* to the **ISAPI Filters** for the website?

- Is Tomcat listening on port 8009? Try the following from the command prompt to make sure:

```
netstat -na | findstr 8009
```

- Have you given JIRA a context in Tomcat's **server.xml**?

```
<Context path="/jira" docBase="${catalina.home}/atlassian-jira" reloadable="false">
```

And does it match the virtual directory and value in your **uriworkermap.properties** file?

```
/jira/*=wlb
```

Deploying JIRA in a clustered environment

> ***Does JIRA support clustering?***

No, though users can vote towards the JIRA clustering feature request if this is a requirement for them.

If you are deploying JIRA to a clustered environment, please ensure that JIRA Webapp is deployed to only **one** node in the cluster. Most clustered applications support this feature (e.g. WebLogic). Also ensure that your load balancer(s) are configured to direct all requests for JIRA to the correct node in the cluster.

If you wish to run JIRA across multiple nodes you will need a license for each node. You can find our discounting policy here.

If you are interested in clustering for hardware failover, you can configure a distributed database, load balancer front-end and two separate

servers. The primary JIRA instance runs on the main server, while the second server contains an unstarted, free development license instance of JIRA. The load balancer directs 100% of requests to the primary as long as it is responsive, but if the primary server goes down, the load balancer starts the backup instance and directs all requests there until manually reset.

> ***Although Atlassian does not support JIRA clustering there are some alternatives available:***

1. WAN Disco
2. Scarlet

Installation Troubleshooting Guide

This troubleshooting guide lists some of the common installation problems people run into

> ⚠ If you have a question that is not answered here, please see our support page for information on how to seek help.

Issues

- My JIRA instance starts up with strange errors, what could be wrong?

My JIRA instance starts up with strange errors, what could be wrong?

If you're using the Windows XP, you may have extracted JIRA with the built-in unzip tool. This built-in unzip tool is broken - it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like 7-zip or WinZIP to unpack JIRA.

If you're using Solaris, it also suffers from similar problems. You will need to use GNU tar to handle the long filenames.

Other users have reported similar problems using Midnight Commander.

Installing a LDAP server on Debian Linux for use with JIRA

This page contains an example of how to install LDAP on Linux. It's assumed that you are working towards LDAP authentication in JIRA or Confluence.

Install LDAP

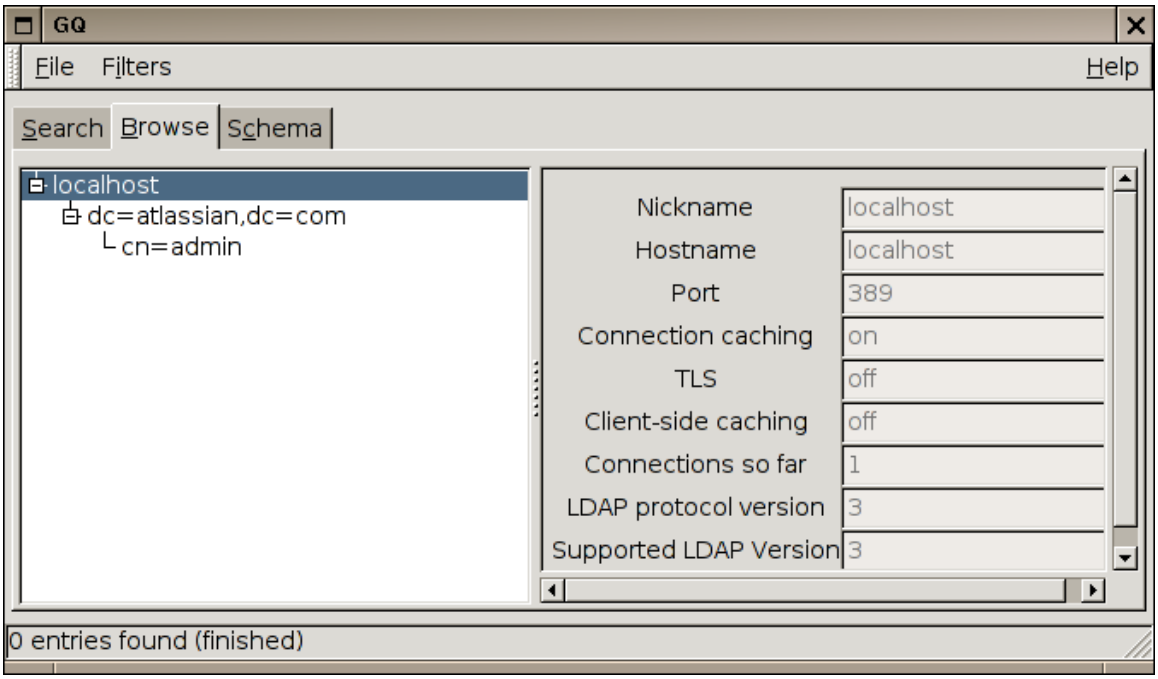
On Debian, an LDAP server can be installed with:

```
apt-get install slapd ldap-utils
```

Entering the following details when prompted (customize for your organization):

Domain name	atlassian.com
Organization name	Atlassian
Admin password	secret
LDAP v2 protocol	no

At this point, you might as well install a graphical LDAP browser, like 'gq'. Connecting anonymously, you'll see there is one entry, `cn=admin,dc=atlassian,dc=com`, created.



Create a schema

Rather than try to devise my own LDAP schema, I used the 'migrationtools' package to create a schema, and import system users from `/etc/passwd`:

```
apt-get install migrationtools
```

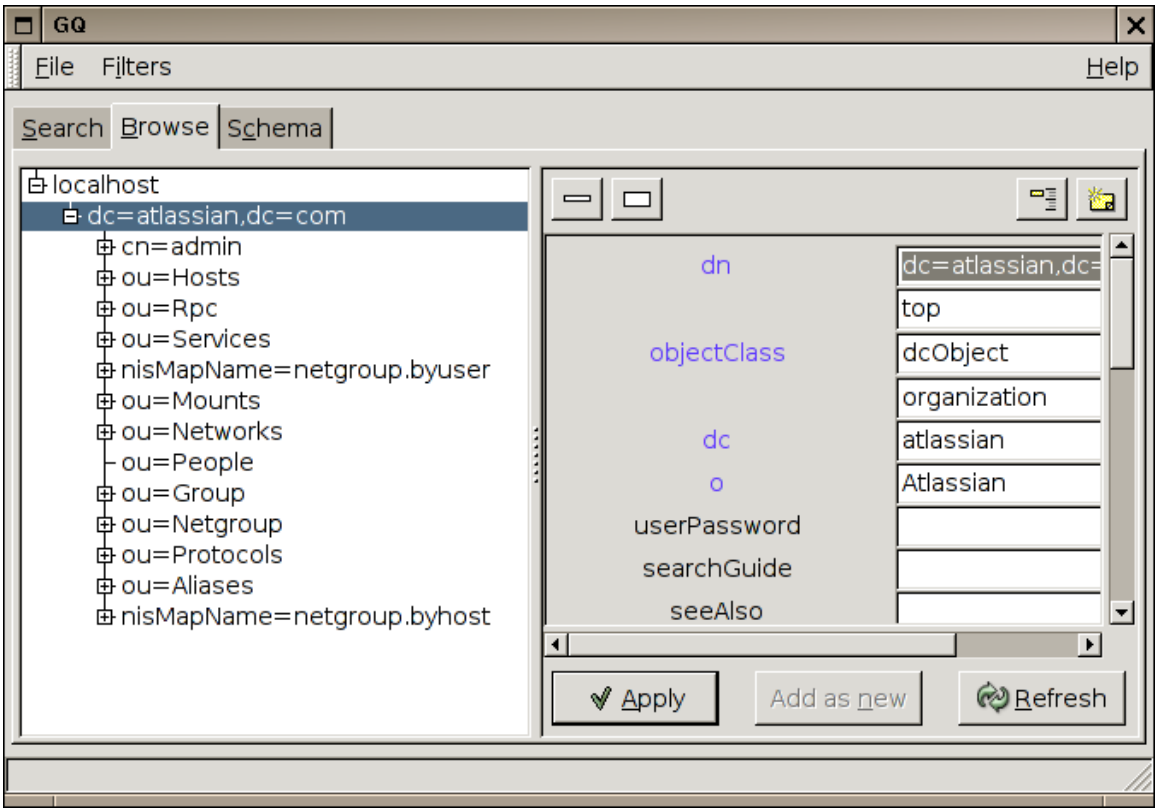
1. Edit `/etc/migrationtools/migrate_common.ph`, and make the following changes:

```
@@ -68,10 +68,10 @@
}

    1. Default DNS domain
       -$DEFAULT_MAIL_DOMAIN = "padl.com";
       +$DEFAULT_MAIL_DOMAIN = "atlassian.com";

    1. Default base
       -$DEFAULT_BASE = "dc=padl,dc=com";
       +$DEFAULT_BASE = "dc=atlassian,dc=com";

    1. Turn this on for inetLocalMailReceipient
    2. sendmail support; add the following to
       @@ -93,8 +93,8 @@
       #$USE_UTF8 = 1;

    1. Uncomment these to avoid Debian managed system users and groups
       -#$IGNORE_UID_BELOW = 1000;
       -#$IGNORE_GID_BELOW = 100;
       +$IGNORE_UID_BELOW = 1000;
       +$IGNORE_GID_BELOW = 100;

    1. And here's the opposite for completeness
       #$IGNORE_UID_ABOVE = 9999;
```

1. Run /usr/share/migrationtools/migrate_all_online

```
teacup:/usr/share/migrationtools# ./migrate_all_online.sh
Enter the X.500 naming context you wish to import into: [dc=padl,dc=com] dc=atlassian,dc=com
Enter the hostname of your LDAP server Integrating JIRA with LDAP: localhost
Enter the manager DN: [cn=admin,dc=atlassian,dc=com]:
Enter the credentials to bind with:
Do you wish to generate a DUAConfigProfile [yes|no]? no

Importing into dc=atlassian,dc=com...

Creating naming context entries...
Migrating aliases...
Migrating groups...
Migrating hosts...
Migrating networks...
Migrating users...
Migrating protocols...
Migrating rpcs...
Migrating services...
Migrating netgroups...
Migrating netgroups (by user)...
Migrating netgroups (by host)...
Importing into LDAP...
adding new entry "ou=Hosts,dc=atlassian,dc=com"

adding new entry "ou=Rpc,dc=atlassian,dc=com"

adding new entry "ou=Services,dc=atlassian,dc=com"

adding new entry "nisMapName=netgroup.byuser,dc=atlassian,dc=com"

adding new entry "ou=Mounts,dc=atlassian,dc=com"

adding new entry "ou=Networks,dc=atlassian,dc=com"

adding new entry "ou=People,dc=atlassian,dc=com"

adding new entry "ou=Group,dc=atlassian,dc=com"

adding new entry "ou=Netgroup,dc=atlassian,dc=com"

adding new entry "ou=Protocols,dc=atlassian,dc=com"

adding new entry "ou=Aliases,dc=atlassian,dc=com"

adding new entry "nisMapName=netgroup.byhost,dc=atlassian,dc=com"

adding new entry "cn=postmaster,ou=Aliases,dc=atlassian,dc=com"
ldapadd: update failed: cn=postmaster,ou=Aliases,dc=atlassian,dc=com
ldap_add: Undefined attribute type (17)
additional info: rfc822MailMember: attribute type undefined

/usr/bin/ldapadd: returned non-zero exit status
```

At this point, you should be able to browse the updated schema in a LDAP browser:



Add users

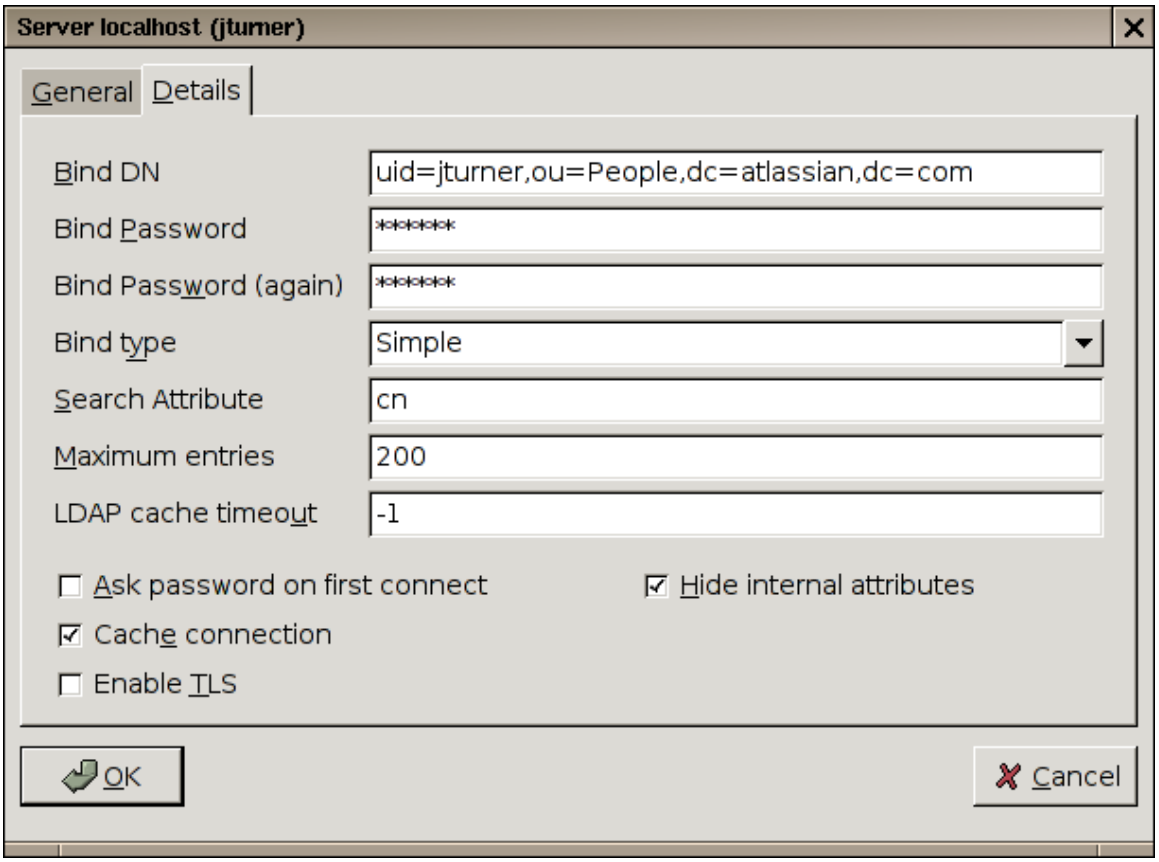
Still in the migrationtools directory, run:

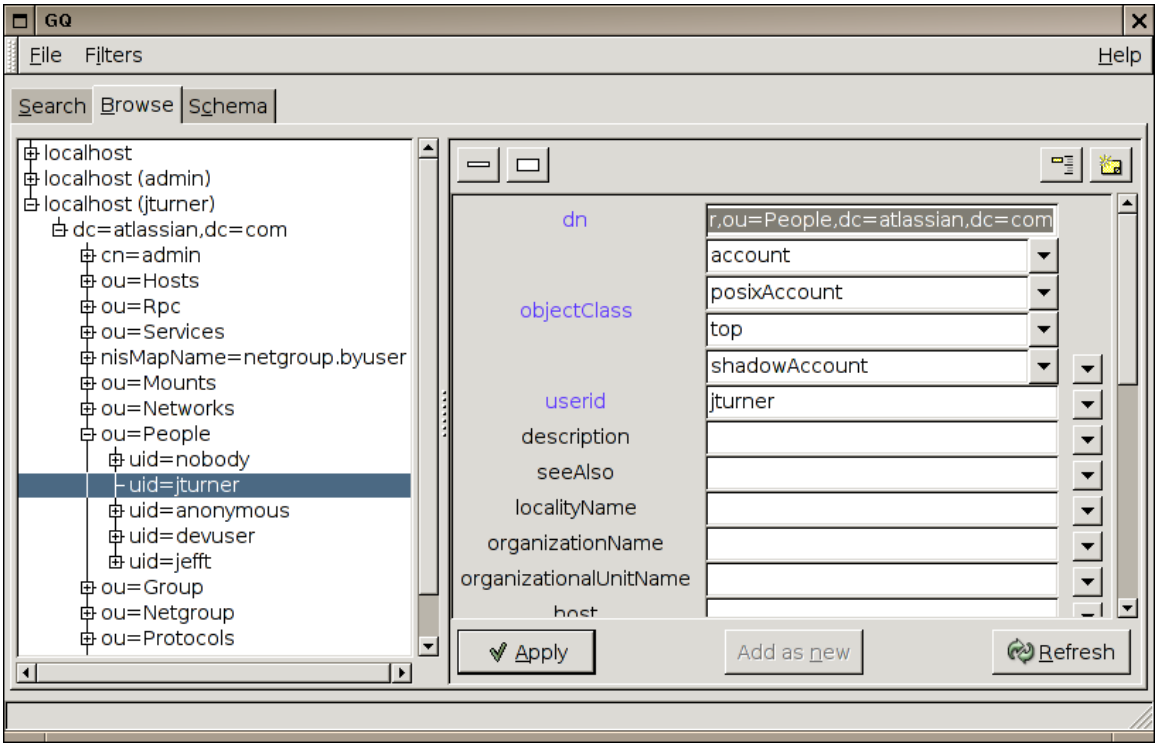
```
teacup:/usr/share/migrationtools# ./migrate_passwd.pl /etc/passwd | ldapadd -x -D "cn=admin,dc=atlassian,dc=com" -W
Enter LDAP Password:
adding new entry "uid=nobody,ou=People,dc=atlassian,dc=com"

adding new entry "uid=jturner,ou=People,dc=atlassian,dc=com"

adding new entry "uid=anonymous,ou=People,dc=atlassian,dc=com"

adding new entry "uid=devuser,ou=People,dc=atlassian,dc=com"

adding new entry "uid=jefft,ou=People,dc=atlassian,dc=com"
```

This creates users, but doesn't set passwords. We must do this manually:

```
teacup:/usr/share/migrationtools# ldappasswd -x -v -S -W -D "cn=admin,dc=atlassian,dc=com"
"uid=jturner,ou=People,dc=atlassian,dc=com"
New password:
Re-enter new password:
Enter LDAP Password:
ldap_initialize( <DEFAULT> )
Result: Success (0)
```

You should now be able to connect anonymously, or as an authenticated user:





Installing Java on Ubuntu or Debian

Some linux distributions (notably Debian and Ubuntu) come with a free version of Java called GIJ (from the GCJ project) pre-installed:

```
jturner:~$ java -version
java version "1.4.2"
gij (GNU libgcj) version 4.1.2 (Ubuntu 4.1.2-0ubuntu5)

Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
jturner:~$
```

Unfortunately GCJ is incomplete, and unable to run Atlassian Java applications without problems.

The solution is to install Sun's version of Java:

```
jturner:~$ sudo apt-get install sun-java6-jdk
Password:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  java-common libltdl3 odbcinst1debian1 sun-java6-bin sun-java6-jre unixodbc
Suggested packages:
  equivs sun-java6-demo sun-java6-doc sun-java6-source sun-java6-plugin ia32-sun-java6-plugin
sun-java6-fonts
  ttf-sazanami-gothic ttf-sazanami-mincho libmyodbc odbc-postgresql libct1
Recommended packages:
  gsfonts-x11
The following NEW packages will be installed:
  java-common libltdl3 odbcinst1debian1 sun-java6-bin sun-java6-jdk sun-java6-jre unixodbc
0 upgraded, 7 newly installed, 0 to remove and 3 not upgraded.
Need to get 42.6MB of archives.
After unpacking 127MB of additional disk space will be used.
Do you want to continue [Y/n]? y

....

Setting up java-common (0.25ubuntu2) ...

Setting up libltdl3 (1.5.22-4) ...

Setting up odbcinst1debian1 (2.2.11-13) ...

Setting up unixodbc (2.2.11-13) ...

Setting up sun-java6-bin (6-00-2ubuntu2) ...

Setting up sun-java6-jre (6-00-2ubuntu2) ...

Setting up sun-java6-jdk (6-00-2ubuntu2) ...

jturner:~$
```

After doing this, make sure the correct version of Java is in use by running `java -version`:

```
jturner:~$ java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build 1.6.0-b105)
Java HotSpot(TM) Server VM (build 1.6.0-b105, mixed mode)
jturner:~$
```

If the GCJ Java is still being used, you will need to explicitly set Sun's Java as the default:

```
jturner:~$ sudo update-alternatives --config java

There are 2 alternatives which provide `java'.

  Selection    Alternative
-----------------------------------------------
*         1    /usr/bin/gij-wrapper-4.1
 +        2    /usr/lib/jvm/java-6-sun/jre/bin/java

Press enter to keep the default[*], or type selection number: 2
Using `/usr/lib/jvm/java-6-sun/jre/bin/java' to provide `java'.
jturner:~$
```

Setting JAVA_HOME

Some programs like Tomcat (bundled with most Atlassian products) need a JAVA_HOME variable set, so they know where Java is installed. This can be set system-wide in `/etc/profile`:

```
jturner:~$ sudo su -
Password:
root:~# cat >> /etc/profile
JAVA_HOME=/usr/lib/jvm/java-6-sun
export JAVA_HOME
root:~#
```

The variable will be set for new terminals:

```
jturner:~$ echo $JAVA_HOME
/usr/lib/jvm/java-6-sun
jturner:~$
```

java.lang.NoClassDefFoundError

If you get the following error when starting JIRA:

> *"java.lang.NoClassDefFoundError:com/atlassian/jira/issue/search/parameters/lucene/SingleFieldMultiValueLuceneParameter"*

this means that Windows XP's unzip is broken. See the Installation Guide for how to avoid this.

JVM and Appserver configuration info

- Causes of OutOfMemoryErrors — When memory problems do occur, the following checklist can help you identify the cause.
- Issues with tomcat 4.x mail and activation jars
- Java crashes
- Setting up a database connection in Websphere 5.1 — This page will help you set up a database connection (DataSource) in Websphere 5.1, through the Websphere admin console.

Causes of OutOfMemoryErrors

To our knowledge, JIRA does not have any memory leaks. We know of various public high-usage JIRA instances (eg. 40k issues, 100+ new issues/day, 22 pages/min in 750Mb of memory) that run for months without problems. When memory problems do occur, the following checklist can help you identify the cause.

Too little memory allocated?

Check the System Info page (see Increasing JIRA memory) after a period of sustained JIRA usage to determine how much memory is allocated.

Windows service, -Xms otherwise)

- [x] Restart JIRA
- [x] Go to Admin -> System Info, and ensure that Total Memory is the minimum you set.

Too much memory allocated?

When increasing Java's memory allocation with -Xmx, please ensure that your system actually has the allocated amount of memory free. For example, if you have a server with 1Gb of RAM, most of it is probably taken up by the operating system, database and whatnot. Setting -Xmx1Gb to a Java process would be a *very bad idea*. Java would claim most of this memory from swap (disk), which would dramatically slow down everything on the server. If the system ran out of swap, you would get OutOfMemoryErrors.

If the server does not have much memory free, it is better to set -Xmx conservatively (eg. -Xmx256m), and only increase -Xmx when you actually see OutOfMemoryErrors. Java's memory management will work to keep within the limit, which is better than going into swap.

Task List

- [ ] **On Windows, ctrl-alt-del, and check the amount of memory marked "Available": !winmem.png|thumbnail!**
- [ ] **On Unix, `cat /proc/meminfo` or use `top` to determine free memory.**
- [ ] **If JIRA is running, check there is spare available memory.**
- [ ] **If raising a support request, please let us know the total system memory and (if on linux) the `/proc/meminfo` output.**

Bugs in older JIRA versions

Please make sure you are using the **latest version of JIRA**. There are often memory leaks fixed in JIRA. Here are some recent ones:

Errors were reported by the JIRA trusted connection.

- APP_UNKNOWN; Unknown Application: {0}; ["confluence:4557196"]

JIRA Issues (17 issues)

Key	Summary	Updated	Status
JRA-18202	Add Google Collections to the webapp classpath to workaround FinalizableReferenceQueue memory leak	Aug 06, 2009	Resolved
JRA-18129	Memory Leak in SAL 2.0.10	Jul 29, 2009	Resolved
JRA-18116	Memory Leak in Apache Shindig	Aug 10, 2009	Resolved
JRA-17390	Memory Leak in Felix framework BundleProtectionDomain	May 22, 2009	Resolved
JRA-16765	Re-enable bundled plugins in setenv	May 11, 2009	Resolved
JRA-16750	Fix any memory leaks in JIRA mainly caused by restoring data from XML and refreshing all singleton objects	May 05, 2009	Resolved
JRA-16742	SOAP search methods are unbounded - this can lead to xml-rpc generating huge xml responses causing memory problems	Apr 14, 2009	Resolved
JRA-15898	too many commit	Nov 05, 2008	Resolved

JRA-15489	Tomcat Manager not unloading classes leading to Permgen errors	Aug 27, 2008	Resolved
JRA-15460	Cannot create index directory on reindexing jira	Aug 26, 2008	Resolved
JRA-15059	One/TwoDimensionalTermHitCollectors use StatsJiraLuceneFieldCache with no cacheing	Jul 15, 2008	Open
JRA-14053	MappedSortComparator needs to reduce its memory footprint	Nov 28, 2007	Closed
JRA-13042	OutOfMemoryError in Events and Issue Status admin pages when lots of issue types and workflows	Jul 11, 2007	Resolved
JRA-12665	CustomFields using the DocumentSortComparatorSource may cause a memory leak when sorting	Apr 02, 2008	Resolved
JRA-12549	JIRA leaks instances of the VelocityEngine in several places	May 14, 2007	Resolved
JRA-12411	OutOfMemoryError during reindex all (due to EagerLoadingOfbizCustomFieldPersister's caching of custom field values)	Dec 12, 2007	Resolved
JRA-10828	SOAP getProjects call can blow up with an OutOfMemoryError	Oct 14, 2008	Resolved

Too many webapps (out of PermGen space)

People running multiple JSP-based web applications (eg. JIRA and Confluence) in one Java server are likely to see this error:

```
java.lang.OutOfMemoryError: PermGen space
```

Java reserves a fixed 64Mb block for loading class files, and with more than one webapp this is often exceeded. You can fix this by setting the `-XX:MaxPermSize=128m` property. See the Increasing JIRA memory page for details.

Tomcat memory leak

Tomcat caches JSP content. If JIRA is generating huge responses (eg. multi-megabyte Excel or RSS views), then these cached responses will quickly fill up memory and result in OutOfMemoryErrors.

In Tomcat 5.5.15+ there is a workaround – set the `org.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true` property (see how). For earlier Tomcat versions, *including* that used in JIRA Standalone 3.6.x and earlier, there is no workaround. Please upgrade Tomcat, or switch to another app server.

Task List

- [x] Ensure you are using Tomcat 5.5.15 or above.
- [x] On Unix, run {ps -ef | grep java} and make sure the LIMIT_BUFFER property is set.

Other webapps

We strongly recommend running JIRA in its own JVM (app server instance), so that web applications cannot affect each other, and each can be restarted/upgraded separately. Usually this is achieved by running app servers behind Apache or IIS.

If you are getting OutOfMemoryErrors, separating the webapps should be your first action. It is virtually impossible to work out retroactively which webapp is consuming all the memory.

Task List

- [ ] **Check which webapps are running (eg. look in /webapps in Tomcat, and/or check the logs for indications of**

what is running.

- [ ] **If raising a support request, please attach all the log files (eg. `logs/* in Tomcat`).**

Plugins

Plugins are a frequent cause of memory problems. If you have any third-party plugins in use, try disabling them temporarily. The same applies to Atlassian plugins such as the toolkit, charting and calendar plugins.

Task List

- [ ] **Get a directory listing of the `WEB-INF/lib` directory, and check for `*-plugin*.jar` files.**
- [ ] **Disable the plugin in the Administration page and remove the jar file from the `WEB-INF/lib` directory.**
- [ ] **If raising a support request, please include this directory listing in the issue.**

Millions of `notificationinstance` records

In order to correctly 'thread' email notifications in mail browsers, JIRA tracks the `Message-Id` header of mails it sends. In heavily used systems, the `notificationinstance` table can become huge, with millions of records. This can cause OutOfMemoryErrors in the JDBC driver when it is asked to generate an XML export of the data (see JRA-11725)

Task List

- [ ] **Run the SQL `select count(*) from notificationinstance;`. If you have over (say) 500,000 records, delete the old ones with {{delete from notificationinstance where id < }}{pick an id halfway}.**

Services (custom, CVS, etc)

Occasionally people write their own services, which can cause memory problem if (as is often the case) they iterate over large numbers of issues. If you have any custom services, please try disabling them for a while to eliminate them as a cause of problems.

The CVS service *sometimes* causes memory problems, if used with a huge CVS repository (in this case, simply increase the allocated memory).

A symptom of a CVS (or general services-related) problem is that JIRA will run out of memory just minutes after startup.

Task List

- [ ] **Go to Admin -> Services**
- [ ] **Check for any services other than the usual (backup, mail queue}.**
- [ ] **If raising a support request, please cut & paste your services list into the issue.**

JIRA backup service with large numbers of issues.

Do you have hundreds of thousands of issues? Is JIRA's built-in backup service running frequently? If so, please switch to a native backup tool and disable the JIRA backup service, which **will** be taking a lot of CPU and memory to generate backups that are unreliable anyway (due to lack of locking). See the JIRA backups documentation for details.

Task List

- [ ] **Check the total issue count in Admin -> System Info**
- [ ] **Go to Admin -> Services**
- [ ] **Check if a backup service is configured and note its frequency.**

JIRA mail misconfiguration causing comment loops.

Does a user have an e-mail address that is the same as one of the mail accounts in your mail handler services? This can cause a comment loop where notifications are sent out and appended to the issue which then triggers another notification and so forth. If a user then views that issue, it could consume a lot of memory. You can query your database using this query that will show you issues with more than 50 comments. It could be normal for issues that have 50 comments, you want to spot for any irregular pattern in the comments themselves such as repeating notifications.

```
SELECT count(*) as commentcount, issueid from jiraaction group by issueid having commentcount > 50
order by commentcount desc
```

The SOAP `getProjects` request

The SOAP `getProjects` call loads a huge object graph, particularly when there are many users in JIRA, and thus can cause OutOfMemoryErrors. Please always use getProjectsNoSchemes instead.

Task List

- [ ] **Ensure no locally run SOAP clients use `getProjects`.**
- [ ] **As below - enable and check access logs.**

Eclipse Mylyn plugin

If your developers use the Eclipse Mylyn plugin, make sure they are using the latest version. The Mylyn bundled with Eclipse 3.3 (2.0.0.v20070627-1400) uses the `getProjects` method, causing problems as described above.

Task List

- [ ] **As below - enable access logging and ensure the latest Mylyn plugin is used.**

Huge XML/RSS or SOAP requests

This applies particularly to publicly visible JIRAs. Sometimes a crawler can slow down JIRA by making multiple huge requests. Every now and then someone misconfigures their RSS reader to request XML for every issue in the system, and sets it running once a minute. Similarly, people sometimes write SOAP clients without consideration of the performance impact, and set it running automatically. JIRA might survive these (although be oddly slow), but then run out of memory when a legitimate user's large Excel view pushes it over the limit.

The best way to diagnose unusual requests is to enable Tomcat access logging (on by default in JIRA Standalone), and look for requests that take a long time.

In JIRA 3.10 there is a `jira.search.views.max.limit` property you can set in `WEB-INF/classes/jira-application.properties`, which is a hard limit on the number of search results returned. It is a good idea to enable this for sites subject to crawler traffic.

Task List

- [ ] **Turn on access logging to see if SOAP requests are being made.**
- [ ] **Check your access logs for long-running or repeated requests.**

Unusual JIRA usage

Every now and then someone reports memory problems, and after much investigation we discover they have 3,000 custom fields, or are parsing 100Mb emails, or have in some other way used JIRA in unexpected ways. Please be aware of where your JIRA installation deviates from typical usage.

Task List

- [ ] **If raising a support request, cut & paste the System Info output, which include basic usage stats.**
- [ ] **Better yet, please attach a JIRA backup of your data (optionally anonymized) so we can replicate the problem.**
- [ ] **Turn on access logging to see how JIRA is being used. If submitting a support request, please submit this log too.**

Memory diagnostics

If you have been through the list above, there are a few further diagnostics which may provide clues.

Getting memory dumps

By far the most powerful and effective way of identifying memory problems is to have JIRA dump the contents of its memory on exit (when exiting due to an OutOfMemoryError hang). These run with no noticeable performance impact. This can be done in one of two ways:

- On Sun's JDK 1.5.0_07 and above, or 1.4.2_12 and above, set the `-XX:+HeapDumpOnOutOfMemoryError` option. If JIRA runs out of memory, it will create a `jira_pid*.hprof` file containing the memory dump in the directory you started JIRA from.
- On other platforms, you can use the yourkit profiler agent. Yourkit can take memory snapshots when when the JVM exits, or when an OutOfMemoryError is imminent (eg. 95% memory used), or when manually triggered. The agent part of Yourkit is freely redistributable. For more information, see Profiling Memory and CPU usage with YourKit.

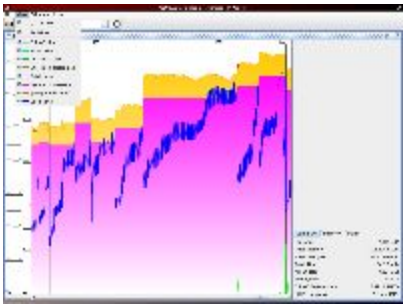
Please **reduce** your maximum heap size (-Xmx) to 750m or so, so that the generated heap dump is of manageable size. You can turn -Xmx up once a heap dump has been taken.

Enable gc logging

Garbage collection logging looks like this:

```
0.000: [GC [PSYoungGen: 3072K->501K(3584K)] 3072K->609K(4992K), 0.0054580 secs]
0.785: [GC [PSYoungGen: 3573K->503K(3584K)] 3681K->883K(4992K), 0.0050140 secs]
1.211: [GC [PSYoungGen: 3575K->511K(3584K)] 3955K->1196K(4992K), 0.0043800 secs]
1.734: [GC [PSYoungGen: 3583K->496K(3584K)] 4268K->1450K(4992K), 0.0045770 secs]
2.437: [GC [PSYoungGen: 3568K->499K(3520K)] 4522K->1770K(4928K), 0.0042520 secs]
2.442: [Full GC [PSYoungGen: 499K->181K(3520K)] [PSOldGen: 1270K->1407K(4224K)]
    1770K->1589K(7744K) [PSPermGen: 6658K->6658K(16384K)], 0.0480810 secs]
3.046: [GC [PSYoungGen: 3008K->535K(3968K)] 4415K->1943K(8192K), 0.0103590 secs]
3.466: [GC [PSYoungGen: 3543K->874K(3968K)] 4951K->2282K(8192K), 0.0051330 secs]
3.856: [GC [PSYoungGen: 3882K->1011K(5248K)] 5290K->2507K(9472K), 0.0094050 secs]
```

This can be parsed with tools like gcviewer to get an overall picture of memory use:



To enable gc logging, start JIRA with the option `-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -Xloggc:gc.log`. Replace `gc.log` with an absolute path to a `gc.log` file.

For example, with a Windows service, run:

```
tomcat5 //US//JIRA ++JvmOptions="-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc
-Xloggc:c:\jira\logs\gc.log"
```

or in `bin/setenv.sh`, set:

```
export CATALINA_OPTS="$CATALINA_OPTS -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc
-Xloggc:${CATALINA_BASE}/logs/gc.log"
```

If you modify `bin/setenv.sh`, you will need to restart JIRA for the changes to take effect.

Access logs

It is important to know what requests are being made, so unusual usage can be identified. For instance, perhaps someone has configured their RSS reader to request a 10Mb RSS file once a minute, and this is killing JIRA.

If you are using Tomcat, access logging can be enabled by adding the following to `conf/server.xml`, below the `</Host>` tag:

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
          pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b %T %S %D" resolveHosts="false" />
```

The %S logs the session ID, allowing requests from distinct users to be grouped. The %D logs the request time in milliseconds. Logs will appear in `logs/access_log.<date>`, and look like this:

```
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /secure/Dashboard.jspa HTTP/1.1" 200 15287 2.835
A2CF5618100BFC43A867261F9054FCB0 2835
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /styles/combined-printable.css HTTP/1.1" 200 111 0.030
A2CF5618100BFC43A867261F9054FCB0 30
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /styles/combined.css HTTP/1.1" 200 38142 0.136
A2CF5618100BFC43A867261F9054FCB0 136
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /styles/global.css HTTP/1.1" 200 548 0.046
A2CF5618100BFC43A867261F9054FCB0 46
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /includes/js/combined-javascript.js HTTP/1.1" 200
65508 0.281 A2CF5618100BFC43A867261F9054FCB0 281
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /includes/js/calendar/calendar.js HTTP/1.1" 200 49414
0.004 A2CF5618100BFC43A867261F9054FCB0 4
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /includes/js/calendar/lang/calendar-en.js HTTP/1.1"
200 3600 0.000 A2CF5618100BFC43A867261F9054FCB0 0
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /includes/js/calendar/calendar-setup.js HTTP/1.1" 200
8851 0.002 A2CF5618100BFC43A867261F9054FCB0 2
127.0.0.1 - - [23/Nov/2006:18:37:48 +1000] "GET /includes/js/cookieUtil.js HTTP/1.1" 200 1506 0.001
A2CF5618100BFC43A867261F9054FCB0 1
```

Alternatively, or if you are not using Tomcat or can't modify the app server config, JIRA has a built-in user access logging which can be enabled from the admin section, and produces terser logs like:

```
2006-09-27 10:35:50,561 INFO [jira.web.filters.AccessLogFilter] bob
http://localhost:8080/secure/IssueNavigator.jspa 102065-4979 1266
2006-09-27 10:35:58,002 INFO [jira.web.filters.AccessLogFilter] bob
http://localhost:8080/secure/IssueNavigator.jspa 102806-4402 1035
2006-09-27 10:36:05,774 INFO [jira.web.filters.AccessLogFilter] bob http://localhost:8080/browse/EAO-2
97058+3717 1730
```

Thread dumps

If JIRA has hung with an OutOfMemoryError, the currently running threads often point to the culprit. Please take a thread dump of the JVM, and send us the logs containing it.

References

Monitoring and Managing Java SE 6 Platform Applications

Issues with tomcat 4.x mail and activation jars

Issue using mail in Jira on Tomcat 4.x

There is an issue with Jira standalone prior to and including 3.2.2 and with deploying the Jira .war into tomcat 4.x. The problem is that Jira requires newer versions of the javax.mail (mail.jar) and the javax.activation (activation.jar) packages than are included in tomcat 4.x.

The solution to the problem is to remove the mail.jar and activation.jar from the TOMCAT_HOME/common/lib directory and replace it with the jars that Jira ships with. To do this you must:

1. Remove the TOMCAT_HOME/common/lib/mail.jar
2. Remove the TOMCAT_HOME/common/lib/activation.jar
3. Move JIRA_WAR_LOCATION/WEB-INF/lib/activation-1.0.2.jar to TOMCAT_HOME/common/lib/activation-1.0.2.jar
4. Move JIRA_WAR_LOCATION/WEB-INF/lib/javamail-1.3.2.jar to TOMCAT_HOME/common/lib/javamail-1.3.2.jar

This should make the mail enabled elements within Jira work properly.

All is not lost

Tomcat 5.0.x and 5.5.x no longer ship the mail and activation jars, so this is not a problem with later versions of tomcat. Jira 3.3 Standalone will ship with Tomcat 5.5.

Java crashes

Symptoms

If you:

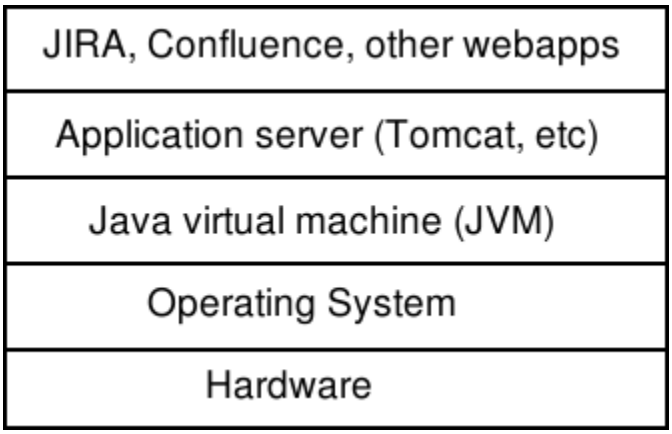
- discover JIRA (and other webapps) completely unavailable
- the Java process isn't even running, and you're sure no-one manually shut it down
- you find files with names like `hs_err_pid20929.log` being created in your app server's `bin` directory (wherever you start it from), containing text like:

```
#
# An unexpected error has been detected by HotSpot Virtual Machine:
#
#  SIGSEGV (0xb) at pc=0xfe9bb960, pid=20929, tid=17
#
# Java VM: Java HotSpot(TM) Server VM (1.5.0_01-b08 mixed mode)
# Problematic frame:
# V  [libjvm.so+0x1bb960]
#

---------------  T H R E A D  ---------------

Current thread (0x01a770e0):  JavaThread "JiraQuartzScheduler_Worker-1" [_thread_in_vm, id=17]

siginfo:si_signo=11, si_errno=0, si_code=1, si_addr=0x00000000

Registers:
 O0=0xf5999882 O1=0xf5999882 O2=0x00000000 O3=0x00000000
 O4=0x00000000 O5=0x00000001 O6=0xc24ff0b0 O7=0x00008000
 G1=0xfe9bb80c G2=0xf5999a48 G3=0x0a67677d G4=0xf5999882
 G5=0xc24ff380 G6=0x00000000 G7=0xfdbc3800 Y=0x00000000
 PC=0xfe9bb960 nPC=0xfe9bb964
....
```

this indicates Java is crashing.

What's happening

Web applications like JIRA and Confluence run in a **Java virtual machine** (JVM). The virtual machine is responsible for emulating a CPU, managing memory and devices, just like the operating system does for native applications (MS Office, web browsers etc).



A Java virtual machine crash is analogous to getting a Windows Blue Screen of Death when using MS Office or a web browser.

So..

That's right, we're **abdicating responsibility**. Java crashes are problems in the virtual machine, not our software. Not even the most wildly broken pure Java software should be able to crash the JVM.

That said, JVM bugs causing crashes often appear when the JVM is running out of memory. The `hs_err_pid*.log` file should list the state of memory at the time of the crash. If the JVM was low on memory, then it's a good idea to read the Causes of OutOfMemoryErrors page and follow the suggestions there.

What should I do?

- Download the latest JDK and use it. Crashes are often caused by JDK bugs that get fixed over time.
- Ensure that your operating system is one of the Supported System Configurations, and (especially for Solaris) ensure that the operating system has all current required patches applied.
- As a last resort, you could try a non-Sun JVM, from vendors such as IBM, BEA (JRockit) or Apache (Harmony).

And of course, you can always let us know (include the hs_err_pid) file), and we can provide advise taking into account the specifics of your

situation.

Setting up a database connection in Websphere 5.1

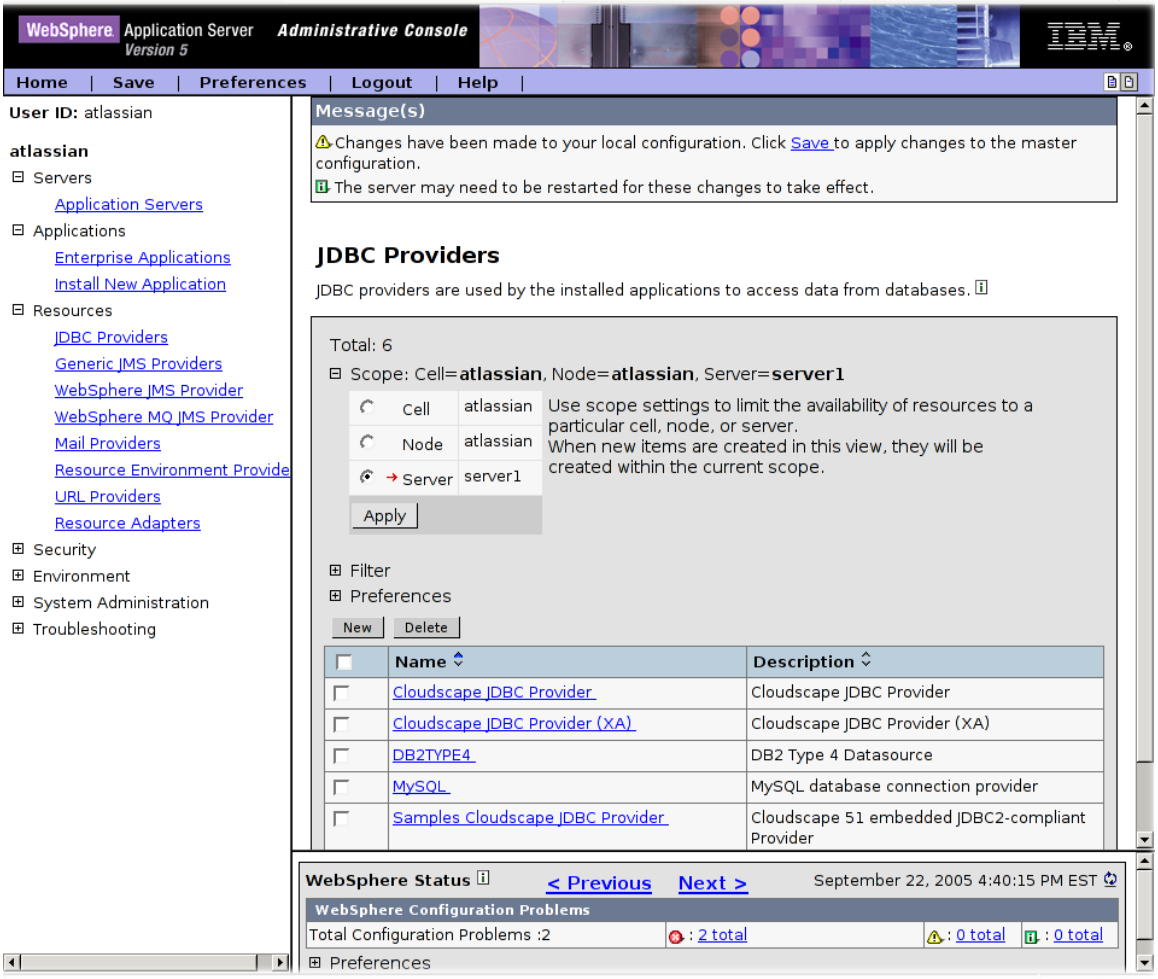
This page will help you set up a database connection (DataSource) in Websphere 5.1, through the Websphere admin console.

Locate the Websphere admin console.

Go to the Websphere admin webapp, which will be accessible on a URL like http://yourserver:9090/admin
http://yourserver:9090/admin

Create a JDBC Provider

Click on Resources -> JDBC Providers, and create a new JDBC Provider for the database you wish to deploy to. Here we are using MySQL:

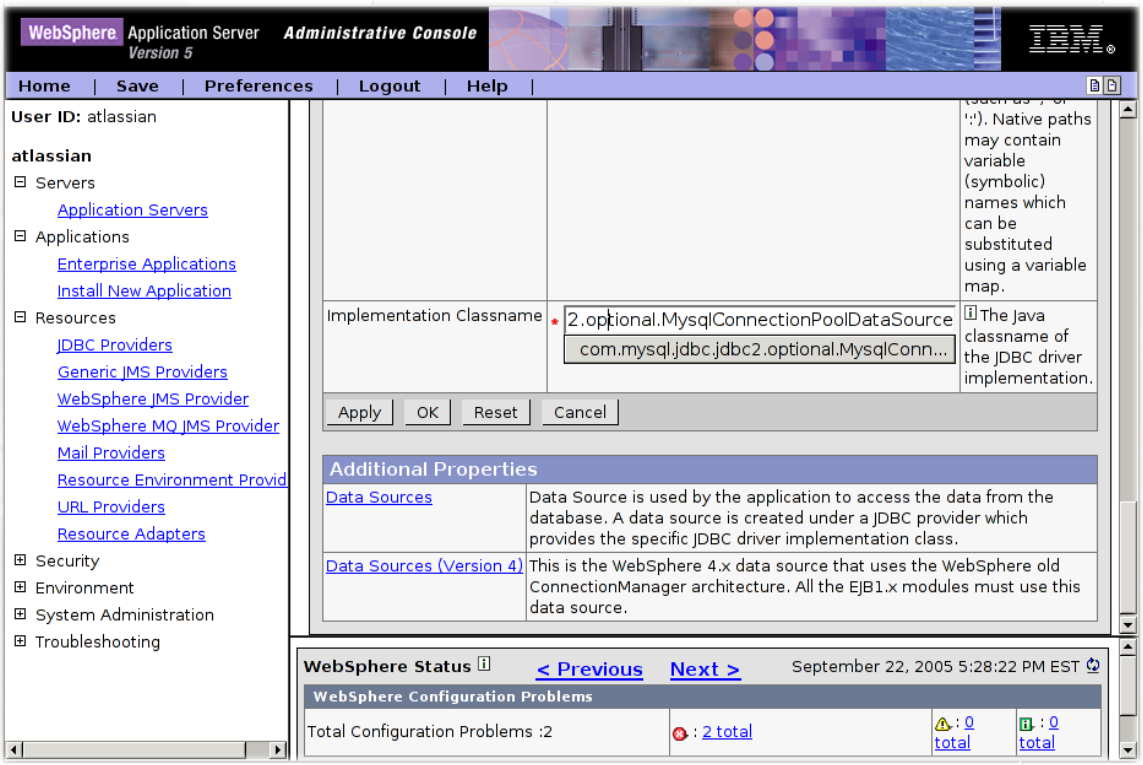


When configuring the JDBC Provider, set the **Classpath** field to include the JDBC driver jar.

In the **Implementation Classname** field, you need to enter the name of a DataSource implementation classname, *not* the JDBC driver classname. For instance, the MySQL JDBC driver class is 'com.mysql.jdbc.Driver',but here we need to enter '**com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource**'.

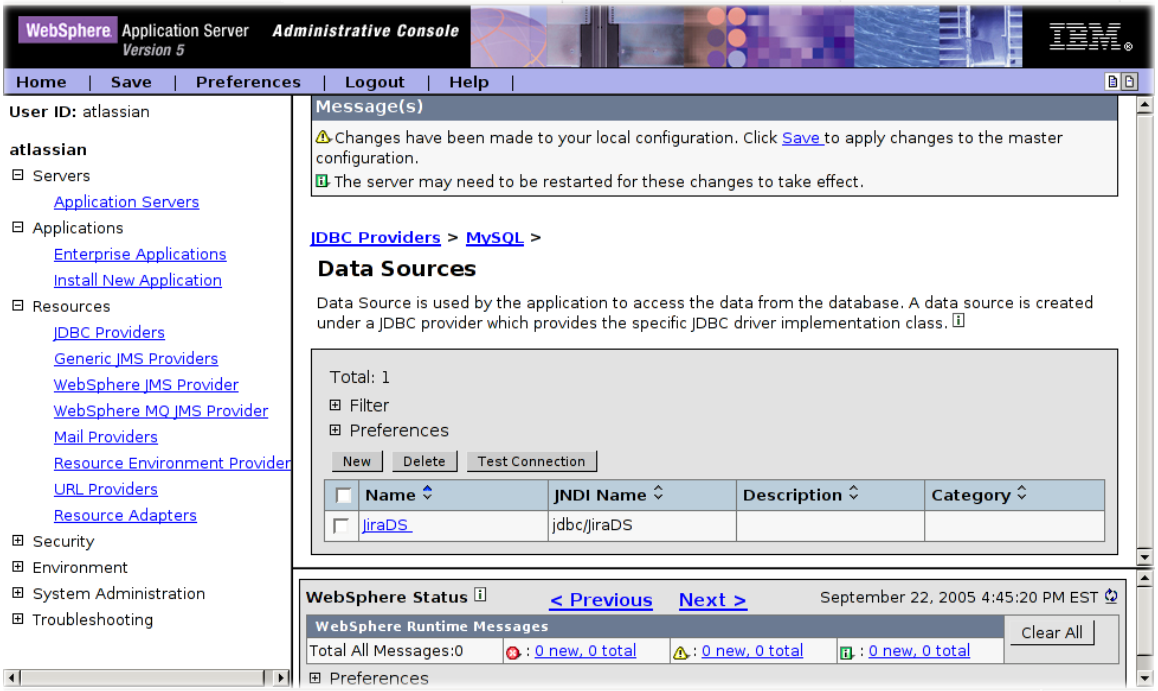
How do you know what class name to enter here? 'Consult your vendor documentation' - or take a guess by looking at the contents of the JDBC driver jar:

```
jturner@teacup:~/mysql$ jar tvf mysql.jar | grep DataSource
1216 Thu Jun 23 00:27:38 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlConnectionPoolDataSource.class
5785 Thu Jun 23 00:27:40 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlDataSource.class
2927 Thu Jun 23 00:27:40 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlDataSourceFactory.class
jturner@teacup:~/mysql$
```



Create a DataSource

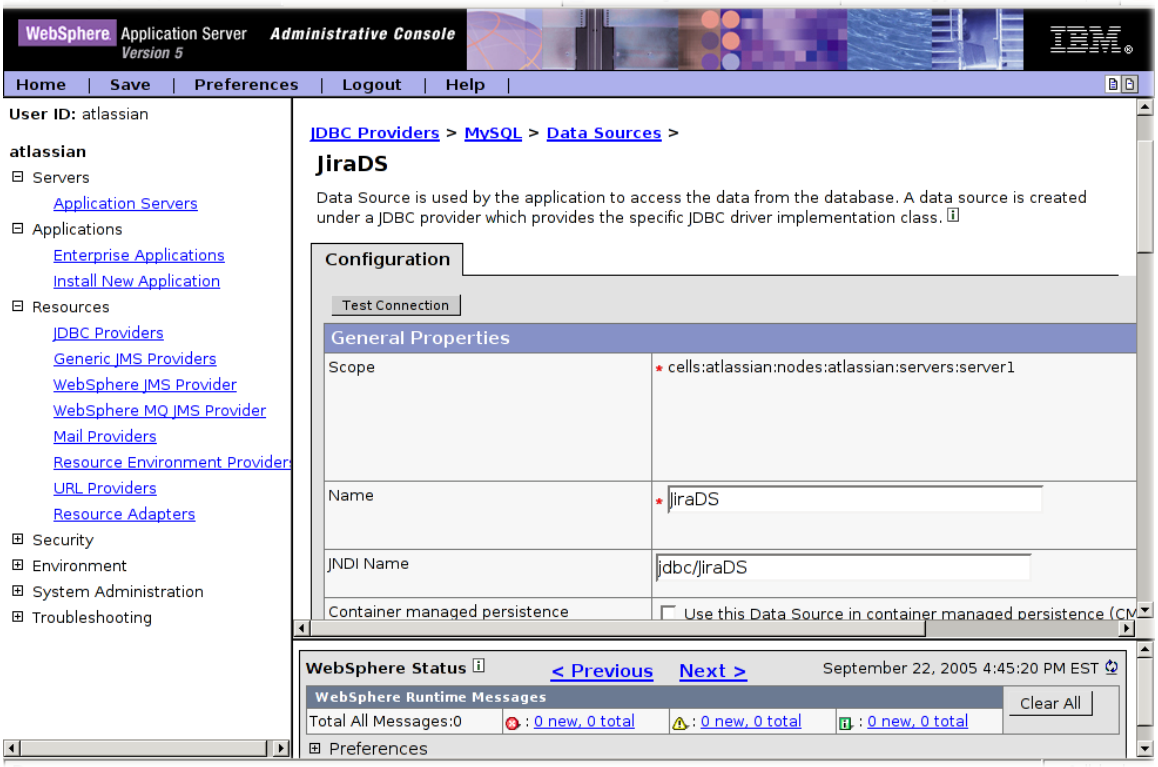
Now click on your newly created JDBC provider, scroll down to the bottom of the right panel, and add a DataSource (the name doesn't matter):



The important thing to get right here is the **JNDI Name**. This must be the same as that specified in your webapp. For JIRA, that means it has to correspond to:

```
<jndi-jdbc jndi-server-name="default" jndi-name="jdbc/JiraDS"/>
```

in WEB-INF/classes/entityengine.xml.



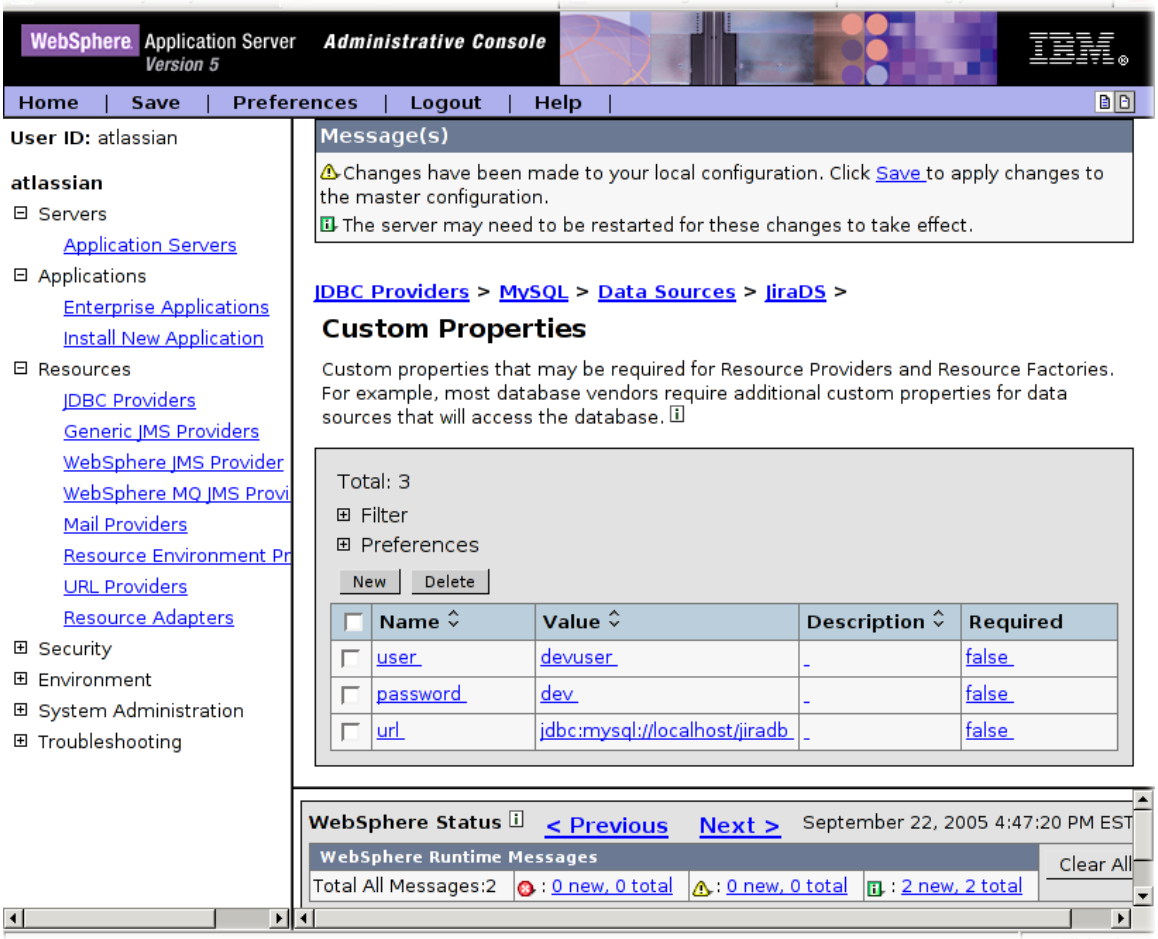
Configure the DataSource

Now scroll to the bottom of the Data Source right panel, and click 'Custom Properties'. Oddly enough, this is where you configure your DataSource with the username, password and database URL to use.

As for the property names, you can simply guess ('user', 'username', etc), or you can introspect the class you specified earlier as follows:

```
jturner@teacup:~/mysql$ export CLASSPATH=mysql.jar
jturner@teacup:~/mysql$ jar tvf mysql.jar \| grep DataSource
1216 Thu Jun 23 00:27:38 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlConnectionPoolDataSource.class
5785 Thu Jun 23 00:27:40 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlDataSource.class
2927 Thu Jun 23 00:27:40 EST 2005 com/mysql/jdbc/jdbc2/optional/MysqlDataSourceFactory.class
jturner@teacup:~/mysql$ javap com.mysql.jdbc.jdbc2.optional.MysqlDataSource
Compiled from "MysqlDataSource.java"
public class com.mysql.jdbc.jdbc2.optional.MysqlDataSource extends com.mysql.jdbc.ConnectionProperties
implements javax.sql.DataSource,javax.naming.Referenceable,java.io.Serializable{
protected static com.mysql.jdbc.Driver mysqlDriver;
protected java.io.PrintWriter logWriter;
protected java.lang.String databaseName;
protected java.lang.String encoding;
protected java.lang.String hostName;
protected java.lang.String password;
protected java.lang.String profileSql;
protected java.lang.String url;
protected java.lang.String user;
protected boolean explicitUrl;
protected int port;
public com.mysql.jdbc.jdbc2.optional.MysqlDataSource();
public java.sql.Connection getConnection()       throws java.sql.SQLException;
public java.sql.Connection getConnection(java.lang.String, java.lang.String)       throws
java.sql.SQLException;
public void setDatabaseName(java.lang.String);
public java.lang.String getDatabaseName();
public void setLogWriter(java.io.PrintWriter)       throws java.sql.SQLException;
public java.io.PrintWriter getLogWriter();
public void setLoginTimeout(int)       throws java.sql.SQLException;
public int getLoginTimeout();
public void setPassword(java.lang.String);
public void setPort(int);
public int getPort();
public void setPortNumber(int);
public int getPortNumber();
public void setPropertiesViaRef(javax.naming.Reference)       throws java.sql.SQLException;
public javax.naming.Reference getReference()       throws javax.naming.NamingException;
public void setServerName(java.lang.String);
public java.lang.String getServerName();
public void setURL(java.lang.String);
public java.lang.String getURL();
public void setUrl(java.lang.String);
public java.lang.String getUrl();
public void setUser(java.lang.String);
public java.lang.String getUser();
protected java.sql.Connection getConnection(java.util.Properties)       throws java.sql.SQLException;
static {};
}
```

Here we can see that the DataSource class has 'setUser', 'setPassword' and 'setURL' methods, so we can infer the existence of 'user', 'password' and 'url' properties:



At this point you can test the connection (the 'Test Connection') button, and if it all works, save the changes.

Known Java Issues

Before installing Java, please take a look at this page for any existing issues.

Sun JDK Issues

For Jira we recommend that you use the latest version of Java. Below are the issues that we have found with earlier versions:

JDK 1.4

- Jira Standalone releases 3.3 and later includes Apache Tomcat 5.5.9. This release of Tomcat is intended for use with JDK 1.5. In order to use JDK 1.4 you will need to download the "jakarta-tomcat-5.5.9-compat" package from the Apache website.
- If you are using FreeBSD, you'll need JDK 1.4.2-p5 or above.

JDK 1.5

- Due to a JVM bug, we recommend that you do not use JDK 1.5.0_06_b05 or JDK 1.5.0_05 (see JRA-9198 for further details). If you are using one of these versions, please upgrade to a later version.
- Older JIRA releases (JIRA 3.1.1 and earlier) do not work 100% with JDK 1.5 due to a JDK bug that affects filter subscriptions. The problem has been worked around in JIRA 3.2 and above.

Older JDK releases

Jira Standalone releases for Jira 3.3 and later no longer support JDK 1.3.

Other Issues

Development

- To build JIRA from source, you will need JDK 1.4 or above

Linux users

- On recent X.org-based distros (eg. FC4+) to avoid getting errors like the following, you will need to install the **xorg-x11-deprecated-libs** package (Fedora) or equivalent (check Google).

```
java.lang.UnsatisfiedLinkError: /opt/j2sdk1.4.2_11/jre/lib/i386/libawt.so: libXp.so.6: cannot
open shared object file: No such file or directory
```

LicenseFactory error after upgrading JIRA

If, after upgrading JIRA, you get an error containing 'com/atlassian/jira/license/LicenseFactory', it means that your application server is using old cached JSPs from the previous JIRA version. Please delete the directory where the app server keeps these (the **work/** directory in Tomcat; the **application-deployments/** directory in Orion; etc), and restart.

Licensing

Please see http://www.atlassian.com/about/licensing/faq.jsp

Logging request headers

If you are having trouble authenticating to JIRA or a web application, it can be useful to log the details of all HTTP request headers that are being sent to the web application. If your application server is Tomcat, you can do this with the **Request Dumper Valve**.

Add the following entry to the `<Engine>` section of your Tomcat `conf/server.xml` file:

```
<Valve className="org.apache.catalina.valves.RequestDumperValve"/>
```

Then restart JIRA.

You will get **lots** of entries like the following in your `logs/catalina.out` log file:

```
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=user-agent=Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.6)
Gecko/20071008 Ubuntu/7.10 (gutsy) Firefox/2.0.0.6
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:
header=accept=text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*
16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=accept-language=en-us,en;q=0.5
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=accept-encoding=gzip,deflate
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=accept-charset=ISO-8859-1,utf-8;q=0.7,*;q=0.7
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=keep-alive=300
12/11/2007 16:27:06 org.apache.catalina.valves.RequestDumperValve invoke
INFO:             header=connection=keep-alive
```

Solaris ClassNotFoundException

After unpacking the WAR on Solaris, JIRA fails to start with a ClassNotFoundException once deployed. How is this fixed?

On Solaris, the default **tar** utility should be avoided as it cannot handle long filenames. **GNU tar** should be used instead in order to handle long filenames found within the JIRA distribution, it can usually be found at:

```
/usr/sfw/bin/
```

Stacktrace example:

```
2006-11-15 15:43:27,539 ERROR [ContainerBase.[Catalina].[localhost].[/]] Error configuring application
listener of class com.atlassian.jira.upgrade.ConsistencyLauncher
java.lang.ClassNotFoundException: com.atlassian.jira.upgrade.ConsistencyLauncher
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1332)
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1181)
            at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3617)
            at org.apache.catalina.core.StandardContext.start(StandardContext.java:4104)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardHost.start(StandardHost.java:718)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardEngine.start(StandardEngine.java:442)
            at org.apache.catalina.core.StandardService.start(StandardService.java:450)
            at org.apache.catalina.core.StandardServer.start(StandardServer.java:683)
            at org.apache.catalina.startup.Catalina.start(Catalina.java:537)
            at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
            at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
            at java.lang.reflect.Method.invoke(Method.java:324)
            at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
            at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)
2006-11-15 15:43:27,582 ERROR [ContainerBase.[Catalina].[localhost].[/]] Error configuring application
listener of class com.atlassian.jira.upgrade.UpgradeLauncher
java.lang.ClassNotFoundException: com.atlassian.jira.upgrade.UpgradeLauncher
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1332)
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1181)
            at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3617)
            at org.apache.catalina.core.StandardContext.start(StandardContext.java:4104)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardHost.start(StandardHost.java:718)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardEngine.start(StandardEngine.java:442)
            at org.apache.catalina.core.StandardService.start(StandardService.java:450)
            at org.apache.catalina.core.StandardServer.start(StandardServer.java:683)
            at org.apache.catalina.startup.Catalina.start(Catalina.java:537)
            at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
            at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
            at java.lang.reflect.Method.invoke(Method.java:324)
            at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
            at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)
2006-11-15 15:43:27,592 ERROR [ContainerBase.[Catalina].[localhost].[/]] Error configuring application
listener of class com.atlassian.jira.scheduler.JiraSchedulerLauncher
java.lang.ClassNotFoundException: com.atlassian.jira.scheduler.JiraSchedulerLauncher
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1332)
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1181)
            at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3617)
            at org.apache.catalina.core.StandardContext.start(StandardContext.java:4104)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardHost.start(StandardHost.java:718)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardEngine.start(StandardEngine.java:442)
            at org.apache.catalina.core.StandardService.start(StandardService.java:450)
            at org.apache.catalina.core.StandardServer.start(StandardServer.java:683)
            at org.apache.catalina.startup.Catalina.start(Catalina.java:537)
            at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
            at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
            at java.lang.reflect.Method.invoke(Method.java:324)
            at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
            at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)
2006-11-15 15:43:27,602 ERROR [ContainerBase.[Catalina].[localhost].[/]] Error configuring application
listener of class com.atlassian.jira.soap.axis.JiraAxisHttpListener
```

```
java.lang.ClassNotFoundException: com.atlassian.jira.soap.axis.JiraAxisHttpListener
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1332)
            at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1181)
            at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3617)
            at org.apache.catalina.core.StandardContext.start(StandardContext.java:4104)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardHost.start(StandardHost.java:718)
            at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
            at org.apache.catalina.core.StandardEngine.start(StandardEngine.java:442)
            at org.apache.catalina.core.StandardService.start(StandardService.java:450)
            at org.apache.catalina.core.StandardServer.start(StandardServer.java:683)
            at org.apache.catalina.startup.Catalina.start(Catalina.java:537)
            at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
            at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
            at java.lang.reflect.Method.invoke(Method.java:324)
            at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
            at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)
```

```
2006-11-15 15:43:27,606 ERROR [ContainerBase.[Catalina].[localhost].[/]] Skipped installing
application listeners due to previous error(s)
```

Transaction Isolation Change exceptions with JBoss

```
2007-09-26 08:14:31,033 INFO  [STDOUT] 08:14:31,030 ERROR
[SimpleWorkflowManager] An exception occurred
org.ofbiz.core.entity.GenericTransactionException: Error occurred
while starting transaction. (Cannot change transaction isolation level in the middle of a
transaction.)
       at org.ofbiz.core.entity.TransactionUtil.beginLocalTransaction(TransactionUtil.java:276)
       at com.atlassian.core.ofbiz.util.CoreTransactionUtil.begin(CoreTransactionUtil.java:31)
       at
com.atlassian.jira.workflow.SimpleWorkflowManager.doWorkflowAction(SimpleWorkflowManager.java:251)
       at
com.atlassian.jira.workflow.WorkflowTransitionUtilImpl.progress(WorkflowTransitionUtilImpl.java:264)
       at
com.atlassian.jira.web.action.workflow.SimpleWorkflowAction.doExecute(SimpleWorkflowAction.java:34)
       at webwork.action.ActionSupport.execute(ActionSupport.java:153)
       ...
caused by: org.postgresql.util.PSQLException: Cannot change transaction isolation level in the middle
of a transaction.
       at
org.postgresql.jdbc2.AbstractJdbc2Connection.setTransactionIsolation(AbstractJdbc2Connection.java:765)
       at
org.jboss.resource.adapter.jdbc.BaseWrapperManagedConnection.setJdbcTransactionIsolation(BaseWrapperManag
at
org.jboss.resource.adapter.jdbc.WrappedConnection.setTransactionIsolation(WrappedConnection.java:390)
       at org.ofbiz.core.entity.TransactionUtil.beginLocalTransaction(TransactionUtil.java:262)
       at com.atlassian.core.ofbiz.util.CoreTransactionUtil.begin(CoreTransactionUtil.java:31)
```

After much pain it appears that the datasource configuration had the following:

```
<new-connection-sql>select 1</new-connection-sql>
<check-valid-connection-sql>select 1</check-valid-connection-sql>
```

Which for some reason seems to be being run within the same transaction as the client code - "fixing" the isolation level.

I tried to raise a bug at jira.jboss.org or comment on this one (that was the only one I could find that seemed to be similar) but after a long and drawn out fight with the jboss.com registration system was unable to log in to it.

Status: has not been reproduced and tested internally but has fixed problem for two customers.
Noticed with:
JIRA 3.8.1 Enterprise: compiled as an ear with plugins for SVN and SVN-commit
JBoss 4.0.5 GA installed with JEMS for EJB3
PostgreSQL 8.1.8
References: this forum post - this JBoss bug
Support Ref: JSP-14364

Windows cannot find -Xms128m

When running startup.bat I get an error message: "Windows cannot find '-Xms128m'"

This error message means that the Java JDK (Java Development Kit) is not installed or the **JAVA_HOME** environment variable has not been set correctly. Please refer to the Java installation guide.

Performance FAQ

JIRA Administrators FAQ

Performance FAQ

- Finding out how many requests a web application currently has
- Indexing in JIRA
  - Thread Dump in JIRA Log File
    - Sample Thread Dump
- Issue Caching Problems in early JIRA versions
- Is your JIRA Running Slowly
  - Testing Database Access Speed
  - Testing Disk Access Speed
  - Windows Performance Manager
- Profiling Memory and CPU usage with YourKit
  - Setting Up YourKit Agent Parameter
  - Troubleshooting Profiling Memory and CPU usage with YourKit
- Slow JIRA Operations

Finding out how many requests a web application currently has

If you want to get an indication of how many requests your web application (e.g. JIRA, Confluence) currently has, and it is running on Unix, run the following command:

```
netstat -a -n | grep :<port number> | grep tcp | grep ESTABLISHED | wc -l
```

For example, if the application server has JIRA (or Confluence) running on port 8080:

```
netstat -a -n | grep :8080 | grep tcp | grep ESTABLISHED | wc -l
```

This will return the number of established TCP/IP connections to that port.

Indexing in JIRA

Overview

Indexing in JIRA is performed by Lucene a high-performance, full-featured text search engine library.

What you need to know

JIRA will perform partial re-indexes of the Lucene index after every issue change, this is handled by the Issue Index Listener viewable in Administration > System > Listeners.

It is also possible to run a complete re-index on demand by running a manual re-index from Administration > System > Indexing. From this location you can also specify a new location for JIRA's index files.

Indexing recommendations

We really only have one major recommendation when it comes to indexing:

Golden Rule
The file path that you specify for JIRA's indexes must be located on a fast local disk.

If JIRA is accessing its indexes across a network drive, Samba or NFS mount you are going to run into indexing exceptions, this would be a likely exception:

```
2006-03-28 13:06:05,540 ERROR [jira.issue.index.DefaultIndexManager] Giving up reindex - waited 10
seconds
com.atlassian.jira.issue.index.IndexException
...
```

So you should never run JIRA with this type of configuration.

Indexing errors

Locking Exception

This can occur when another process has a lock on the index, possible causes of this are:

- There is a lock file left on disk from a previous unclean restart.
- An old JIRA instance still has threads running; possibly from an earlier JIRA instance that was not entirely shut down when JIRA was last restarted.
- A process from the current JIRA instance that is taking an unusually long time (10+seconds) finishing.

If your logs mention a .lock file the safest solution is too:

1. Stop JIRA
2. Delete the .lock files
3. Restart JIRA
4. Perform a full re-index.

These lock files (if you are running JIRA standalone) will be in Tomcat's temp directory. If you need some extra information finding these lock files it may help to have a quick read of the F.A.Q. article on the Lucene wiki.

Timeout Exceptions

Increase the index wait time

If you are seeing exceptions similar to the example above there may be some contention for the lucene indexes. The first option is to increase the allowable timeout for obtaining a lock.

This is done by modifying the jira-application.properties file, specifically the value:

- jira.index.lock.waittime=10000 (please note this time is in milliseconds)
  More information on modifying values in this properties file is available here:
  Advanced JIRA configuration with jira-application.properties

Take a thread dump

JIRA v3.7 and later will automatically perform do a thread dump into the log file in the event of index lock timeouts. If the above does not work and you are still seeing these exceptions it is time to create a support request and attach a thread dump when this error is occurring. Hopefully this thread dump will show what JIRA is currently waiting on to grab the lock on the indexing files.

Poor Indexing Performance

If indexing performance is poor, the typical result is slow response times and index timeouts. Index timeouts will result in potentially out-of-date issue results from searches and in issue navigator and portlets.

Note that poor indexing performance is typically the result of poor disk performance. Note also that an overwhelmingly common cause of poor disk performance on MS Windows is anti-virus software. Understandably, many customers find it difficult to understand how many anti-virus products can continue to have a serious negative impact on disk performance *even when the antivirus software is disabled*. Nevertheless, it is sometimes necessary to completely uninstall anti-virus software in order to correctly assess the impact it has on disk performance and JIRA indexing. Many anti-virus products can exclude designated directories from close, performance degrading monitoring. The JIRA index directory would be an ideal candidate for such an exclusion.

You can use our disk access speed utility to determine if disk performance is a problem for you. Compare the results you get to our published **Good** results in that page.

Thread Dump in JIRA Log File

Symptom

- Slow response times and/or out-of-date issue results from searches and in issue navigator and portlets.
- The JIRA log file (e.g. `catalina.out` for Tomcat) contains output similar to the following:

```
2007-05-10 09:39:53,580 TP-Processor68 ERROR [jira.issue.index.DefaultIndexManager] Lock attempt timed
out - waited 30000 milliseconds
com.atlassian.jira.issue.index.IndexException: Lock timout - waited 30000 milliseconds
    at com.atlassian.jira.issue.index.DefaultIndexManager.getIndexLock(DefaultIndexManager.java:451)
    at com.atlassian.jira.issue.index.DefaultIndexManager.reIndexIssues(DefaultIndexManager.java:279)
    at com.atlassian.jira.issue.index.DefaultIndexManager.reIndexIssues(DefaultIndexManager.java:244)
  .
  .
  .
```

followed by, at the end of the stack trace:

```
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] ************
Start Thread Dump 1178807993608 *******************
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]  -- Memory
Details --
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Total Memory =
2023MB
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Used Memory =
1360MB
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Free Memory =
663MB
2007-05-10 09:39:53,608 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]  --- --- --- ---
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Thread=
TP-Processor15 daemon prio=5id=60 RUNNABLE
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.net.SocketInputStream.socketRead0(Native Method)
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.net.SocketInputStream.read(SocketInputStream.java:129)
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.io.BufferedInputStream.fill(BufferedInputStream.java:218)
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.io.BufferedInputStream.read1(BufferedInputStream.java:256)
2007-05-10 09:39:53,609 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.io.BufferedInputStream.read(BufferedInputStream.java:313)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.jk.common.ChannelSocket.read(ChannelSocket.java:620)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.jk.common.ChannelSocket.receive(ChannelSocket.java:558)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.jk.common.ChannelSocket.processConnection(ChannelSocket.java:685)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.jk.common.ChannelSocket$SocketConnection.runIt(ChannelSocket.java:889)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:684)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.lang.Thread.run(Thread.java:595)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]  --- --- --- ---
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Thread=
http-8588-Processor5 daemon prio=5id=22 WAITING
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.lang.Object.wait(Native Method)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.lang.Object.wait(Object.java:474)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.lang.Thread.run(Thread.java:595)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]  --- --- --- ---
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper] Thread=
TP-Processor45 daemon prio=5id=96 RUNNABLE
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.net.SocketInputStream.socketRead0(Native Method)
2007-05-10 09:39:53,610 TP-Processor68 ERROR [jdk.utilities.threaddump.ThreadDumper]
java.net.SocketInputStream.read(SocketInputStream.java:129)
  .
  .
  .
```

What is going on?

A Lucene index lock timeout has occurred. JIRA v3.7 and later automatically performs a thread dump (into the log file) in the event of index lock timeouts.

The thread dump is only generated if JIRA is running with JDK 1.5 or later. JDK 1.4 and earlier do not support programmatically generating a thread dump.

See Indexing in JIRA for a description of Lucene indexing, index locks, and index lock timeout.

What can you do?

Again, read Indexing in JIRA for common causes of poor indexing performance. If you still need help after that, create a support request. We are here to help. Make sure you include the log file.

If you like getting your hands dirty...

If you are from a software development background, and you would like to know more about what is going on, read on...

If you are keen, but you are not familiar with how to interpret Java stack traces and thread dumps, this article will help.

It is useful to know which thread had the lock when the timeout occurred. Look for a thread in a runnable state that contains Atlassian specific code. For example, the following output indicates that a Bulk Edit operation had the lock when the timeout occurred:

```
Thread= TP-Processor57 daemon prio=5id=108 RUNNABLE
 java.io.RandomAccessFile.writeBytes(Native Method)
 java.io.RandomAccessFile.write(RandomAccessFile.java:456)
 org.apache.lucene.store.FSIndexOutput.flushBuffer(FSDirectory.java:498)
 org.apache.lucene.store.BufferedIndexOutput.flush(BufferedIndexOutput.java:84)
 org.apache.lucene.store.BufferedIndexOutput.writeBytes(BufferedIndexOutput.java:74)
 org.apache.lucene.index.CompoundFileWriter.copyFile(CompoundFileWriter.java:211)
 org.apache.lucene.index.CompoundFileWriter.close(CompoundFileWriter.java:168)
 org.apache.lucene.index.SegmentMerger.createCompoundFile(SegmentMerger.java:145)
 org.apache.lucene.index.IndexWriter.mergeSegments(IndexWriter.java:706)
 org.apache.lucene.index.IndexWriter.mergeSegments(IndexWriter.java:658)
 org.apache.lucene.index.IndexWriter.optimize(IndexWriter.java:517)
 com.atlassian.bonnie.ConcurrentLuceneConnection$2.perform(ConcurrentLuceneConnection.java:120)
 com.atlassian.bonnie.ConcurrentLuceneConnection.withWriter(ConcurrentLuceneConnection.java:296)
 com.atlassian.bonnie.ConcurrentLuceneConnection.optimize(ConcurrentLuceneConnection.java:116)
 com.atlassian.jira.issue.index.SingleThreadedIssueIndexer.optimize(SingleThreadedIssueIndexer.java:75)
com.atlassian.jira.issue.index.DefaultIndexManager.optimize0(DefaultIndexManager.java:393)
 com.atlassian.jira.issue.index.DefaultIndexManager.optimizeIfNecessary(DefaultIndexManager.java:332)
 com.atlassian.jira.issue.index.DefaultIndexManager.reIndexIssues(DefaultIndexManager.java:288)
 com.atlassian.jira.issue.index.DefaultIndexManager.reIndexIssues(DefaultIndexManager.java:244)
 com.atlassian.jira.issue.index.DefaultIndexManager.reIndex(DefaultIndexManager.java:264)
 com.atlassian.jira.event.listeners.search.IssueIndexListener.reIndex(IssueIndexListener.java:114)
 com.atlassian.jira.event.listeners.search.IssueIndexListener.reIndex(IssueIndexListener.java:124)
 com.atlassian.jira.event.listeners.search.IssueIndexListener.issueUpdated(IssueIndexListener.java:55)
 com.atlassian.jira.event.issue.AbstractIssueEventListener.workflowEvent(AbstractIssueEventListener.java:
com.atlassian.jira.event.issue.IssueEventDispatcher.dispatchEvent(IssueEventDispatcher.java:107)
 com.atlassian.jira.event.issue.IssueEventDispatcher.dispatchEvent(IssueEventDispatcher.java:88)
 com.atlassian.jira.issue.util.DefaultIssueUpdater.doUpdate(DefaultIssueUpdater.java:63)
 com.atlassian.jira.action.issue.AbstractIssueUpdateAction.doUpdate(AbstractIssueUpdateAction.java:137)
com.atlassian.jira.action.issue.AbstractIssueUpdateAction.doUpdate(AbstractIssueUpdateAction.java:127)
 com.atlassian.jira.action.issue.IssueUpdate.doExecute(IssueUpdate.java:75)
 webwork.action.ActionSupport.execute(ActionSupport.java:153)
 com.atlassian.jira.action.JiraActionSupport.execute(JiraActionSupport.java:54)
 com.atlassian.jira.action.JiraNonWebActionSupport.execute(JiraNonWebActionSupport.java:27)
 webwork.dispatcher.GenericDispatcher.executeAction(GenericDispatcher.java:132)
 com.atlassian.core.action.DefaultActionDispatcher.execute(DefaultActionDispatcher.java:34)
 com.atlassian.jira.bulkedit.operation.BulkEditOperation.perform(BulkEditOperation.java:199)
 com.atlassian.jira.web.action.issue.bulkedit.BulkEdit.doPerform(BulkEdit.java:164)
        .
        .
        .
```

The actual root problem is probably specific to your issue. If you are keen, this might give you enough information to diagnose what is going on. Remember though, Atlassian support is ready to help you.

Related

Logging a thread dump also contains a thread dump analyzer TDA 1.0 Final

Sample Thread Dump

```
Full thread dump Java HotSpot(TM) Client VM (1.4.2_06-b03 mixed mode):

"http-8080-Monitor" prio=1 tid=0x0851cdf0 nid=0x38e in Object.wait() [5a40f000..5a40f854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d3710> (a org.apache.tomcat.util.threads.ThreadPool$MonitorRunnable)
        at org.apache.tomcat.util.threads.ThreadPool$MonitorRunnable.run(ThreadPool.java:559)
        - locked <0x449d3710> (a org.apache.tomcat.util.threads.ThreadPool$MonitorRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor25" daemon prio=1 tid=0x0851c298 nid=0x38e runnable [5a289000..5a289854]
        at java.net.PlainSocketImpl.socketAccept(Native Method)
        at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:353)
        - locked <0x45d47238> (a java.net.PlainSocketImpl)
        at java.net.ServerSocket.implAccept(ServerSocket.java:448)
        at java.net.ServerSocket.accept(ServerSocket.java:419)
        at
org.apache.tomcat.util.net.DefaultServerSocketFactory.acceptSocket(DefaultServerSocketFactory.java:60)
        at org.apache.tomcat.util.net.PoolTcpEndpoint.acceptSocket(PoolTcpEndpoint.java:407)
        at
org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt(LeaderFollowerWorkerThread.java:70)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:684)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor24" daemon prio=1 tid=0x0851b740 nid=0x38e in Object.wait() [5a208000..5a208854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d2fc0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d2fc0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor23" daemon prio=1 tid=0x080cb6b8 nid=0x38e in Object.wait() [5a187000..5a187854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d2c18> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d2c18> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor22" daemon prio=1 tid=0x080cab78 nid=0x38e in Object.wait() [5a106000..5a106854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d2870> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d2870> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor21" daemon prio=1 tid=0x080ca1f8 nid=0x38e in Object.wait() [5a025000..5a025854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d23b8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d23b8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor20" daemon prio=1 tid=0x080c9878 nid=0x38e in Object.wait() [59fa4000..59fa4854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d2010> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d2010> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor19" daemon prio=1 tid=0x08668058 nid=0x38e in Object.wait() [59f23000..59f23854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d1c68> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d1c68> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)
```

```
"http-8080-Processor18" daemon prio=1 tid=0x086676f0 nid=0x38e in Object.wait() [59ea2000..59ea2854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d17f8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d17f8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor17" daemon prio=1 tid=0x08666d88 nid=0x38e in Object.wait() [59e21000..59e21854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d1450> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d1450> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor16" daemon prio=1 tid=0x08666420 nid=0x38e in Object.wait() [59da0000..59da0854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d10a8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d10a8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor15" daemon prio=1 tid=0x086659a8 nid=0x38e in Object.wait() [59d1f000..59d1f854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d0d00> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d0d00> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor14" daemon prio=1 tid=0x08790380 nid=0x38e in Object.wait() [59c9e000..59c9e854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d0958> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d0958> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor13" daemon prio=1 tid=0x0878fa18 nid=0x38e in Object.wait() [59c1d000..59c1d854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d05b0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d05b0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor12" daemon prio=1 tid=0x0878f0b0 nid=0x38e in Object.wait() [59b9c000..59b9c854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449d0208> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449d0208> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor11" daemon prio=1 tid=0x084ec3f0 nid=0x38e in Object.wait() [59b1b000..59b1b854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cfe60> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cfe60> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor10" daemon prio=1 tid=0x084ec288 nid=0x38e in Object.wait() [59a9a000..59a9a854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cfab8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
```

```
        - locked <0x449cfab8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor9" daemon prio=1 tid=0x084eb908 nid=0x38e in Object.wait() [59a19000..59a19854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cf6a8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cf6a8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor8" daemon prio=1 tid=0x084eaf88 nid=0x38e in Object.wait() [59998000..59998854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cf300> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cf300> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor7" daemon prio=1 tid=0x084ea608 nid=0x38e in Object.wait() [59917000..59917854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cef58> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cef58> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor6" daemon prio=1 tid=0x08817b48 nid=0x38e in Object.wait() [59896000..59896854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cebb0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cebb0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor5" daemon prio=1 tid=0x088171c8 nid=0x38e in Object.wait() [59815000..59815854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449ce778> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449ce778> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor4" daemon prio=1 tid=0x083ae308 nid=0x38e in Object.wait() [59794000..59794854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449ce3d0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449ce3d0> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor3" daemon prio=1 tid=0x083add88 nid=0x38e in Object.wait() [59713000..59713854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449ce028> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449ce028> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor2" daemon prio=1 tid=0x083ad408 nid=0x38e in Object.wait() [59692000..59692854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cdc80> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cdc80> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"http-8080-Processor1" daemon prio=1 tid=0x0883e478 nid=0x38e in Object.wait() [59611000..59611854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x449cd8d8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
```

```
        at java.lang.Object.wait(Object.java:429)
        at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:656)
        - locked <0x449cd8d8> (a org.apache.tomcat.util.threads.ThreadPool$ControlRunnable)
        at java.lang.Thread.run(Thread.java:534)

"ContainerBackgroundProcessor[StandardEngine[Catalina]]" daemon prio=1 tid=0x08445388 nid=0x38e
waiting on condition [59590000..59590854]
        at java.lang.Thread.sleep(Native Method)
        at
org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessor.run(ContainerBase.java:1546)
        at java.lang.Thread.run(Thread.java:534)

"Thread-4" prio=1 tid=0x0879d888 nid=0x38e waiting on condition [5aa74000..5aa74854]
        at java.lang.Thread.sleep(Native Method)
        at com.atlassian.jira.upgrade.ConnectionKeeper.run(ConnectionKeeper.java:63)

"JonasClock" daemon prio=1 tid=0x0872f2f8 nid=0x38e waiting on condition [5a9f3000..5a9f3854]
        at java.lang.Thread.sleep(Native Method)
        at org.objectweb.jonas_timer.TimerManager.clock(TimerManager.java:142)
        at org.objectweb.jonas_timer.Clock.run(TimerManager.java:46)

"JonasBatch" daemon prio=1 tid=0x0872f020 nid=0x38e in Object.wait() [5a66c000..5a66c854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x461c0780> (a java.util.Vector)
        at java.lang.Object.wait(Object.java:429)
        at org.objectweb.jonas_timer.TimerManager.batch(TimerManager.java:193)
        - locked <0x461c0780> (a java.util.Vector)
        at org.objectweb.jonas_timer.Batch.run(TimerManager.java:65)

"JiraQuartzScheduler_QuartzSchedulerThread" prio=1 tid=0x08662610 nid=0x38e in Object.wait()
[5a972000..5a972854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x46080b18> (a java.lang.Object)
        at org.quartz.core.QuartzSchedulerThread.run(QuartzSchedulerThread.java:217)
        - locked <0x46080b18> (a java.lang.Object)

"JiraQuartzScheduler_Worker-3" prio=1 tid=0x086246a8 nid=0x38e in Object.wait() [5a8f1000..5a8f1854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.getNextRunnable(SimpleThreadPool.java:423)
        - locked <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.access$000(SimpleThreadPool.java:53)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:514)

"JiraQuartzScheduler_Worker-2" prio=1 tid=0x08626918 nid=0x38e in Object.wait() [5a870000..5a870854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.getNextRunnable(SimpleThreadPool.java:423)
        - locked <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.access$000(SimpleThreadPool.java:53)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:514)

"JiraQuartzScheduler_Worker-1" prio=1 tid=0x086263b0 nid=0x38e in Object.wait() [5a7ef000..5a7ef854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.getNextRunnable(SimpleThreadPool.java:423)
        - locked <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.access$000(SimpleThreadPool.java:53)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:514)

"JiraQuartzScheduler_Worker-0" prio=1 tid=0x08623f28 nid=0x38e in Object.wait() [5a76e000..5a76e854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.getNextRunnable(SimpleThreadPool.java:423)
        - locked <0x46080b78> (a java.lang.Object)
        at org.quartz.simpl.SimpleThreadPool.access$000(SimpleThreadPool.java:53)
        at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:514)

"Thread-3" prio=1 tid=0x08622a28 nid=0x38e waiting on condition [5a6ed000..5a6ed854]
        at java.lang.Thread.sleep(Native Method)
```

```
        at org.hsqldb.Log.run(Unknown Source)
        at java.lang.Thread.run(Thread.java:534)

"Thread-1" daemon prio=1 tid=0x08617f88 nid=0x38e waiting on condition [5a5eb000..5a5eb854]
        at java.lang.Thread.sleep(Native Method)
        at org.apache.tomcat.dbcp.pool.impl.GenericObjectPool$Evictor.run(GenericObjectPool.java:1080)
        at java.lang.Thread.run(Thread.java:534)

"Signal Dispatcher" daemon prio=1 tid=0x080aefe8 nid=0x38e waiting on condition [0..0]

"Finalizer" daemon prio=1 tid=0x0809a750 nid=0x38e in Object.wait() [58b1b000..58b1b854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x45c10138> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:111)
        - locked <0x45c10138> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:127)
        at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:159)

"Reference Handler" daemon prio=1 tid=0x08099ba8 nid=0x38e in Object.wait() [58a9a000..58a9a854]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x45c101a0> (a java.lang.ref.Reference$Lock)
        at java.lang.Object.wait(Object.java:429)
        at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:115)
        - locked <0x45c101a0> (a java.lang.ref.Reference$Lock)

"main" prio=1 tid=0x0805b3b0 nid=0x38e runnable [bfffc000..bfffcec8]
        at java.net.PlainSocketImpl.socketAccept(Native Method)
        at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:353)
        - locked <0x44f575b8> (a java.net.PlainSocketImpl)
        at java.net.ServerSocket.implAccept(ServerSocket.java:448)
        at java.net.ServerSocket.accept(ServerSocket.java:419)
        at org.apache.catalina.core.StandardServer.await(StandardServer.java:362)
        at org.apache.catalina.startup.Catalina.await(Catalina.java:600)
        at org.apache.catalina.startup.Catalina.start(Catalina.java:560)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:324)
        at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
        at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)

"VM Thread" prio=1 tid=0x08098948 nid=0x38e runnable
```

```
"VM Periodic Task Thread" prio=1 tid=0x080b17f8 nid=0x38e waiting on condition
"Suspend Checker Thread" prio=1 tid=0x080ae660 nid=0x38e runnable
```

Issue Caching Problems in early JIRA versions

Early JIRA versions (confirmed JIRA 2.5.x and prior) suffer from occasional NPEs with stacktraces that look something like this:

---

com.ibm.ws.webcontainer.webapp.WebAppErrorReport: null<p><small><small><pre>**java.lang.NullPointerException**

at **java.util.LinkedList.remove**(LinkedList.java(Compiled Code))

at org.ofbiz.core.util.UtilCache.get(UtilCache.java(Compiled Code))

at **com.atlassian.jira.issue.cache.IssueCache.getIssue**(IssueCache.java(Compiled Code))

at com.atlassian.jira.issue.cache.IssueCache.getIssue(IssueCache.java(Compiled Code))

at com.atlassian.jira.issue.cache.IssueCache.getIssue(IssueCache.java(Compiled Code))

...

This will occur somewhat randomly while **Cache Issues**, accessible in the General Configuration, is set to ON.

The performance problem that lead to issue caching being necessary has been fixed in later versions, so the best option is to upgrade. Otherwise, you can disable issue caching at the cost of performance.

Is your JIRA Running Slowly

If JIRA is not responding as quickly as a previous version or is taking a while to perform certain tasks, please follow the instructions on this page.

- Tune JIRA
- Gather information
    - Profiling
    - Thread Dumps
    - What is the machine doing?
    - Garbage Collection
    - Database Connection Pooling
- Get Help

Tune JIRA

There are quite a few performance tips that can help JIRA to run quicker and may in fact solve your issues.

Gather information

If tuning JIRA did not help, you will need to find exactly what is slow and what is causing it to run slow.

Profiling

Lets find out where JIRA is spending its time.
JIRA comes with its own built-in Profiling Tool. It allows us to see how long JIRA spends doing individual requests and where that time is being taken.
The tool logs a tree of the time spent in certain methods with the output looking like:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

Thread Dumps

If JIRA is taking a long time to execute a specific operation (E.g. Creating an issue), it can be useful to get a complete thread dump of the JVM to find out exactly what is being executed.
For example, if *Create Issue* is taking a long time, execute the create issue and while you are waiting for it to return, execute a thread dump every few seconds. (for a 20 second operation, 5 thread dumps would be useful).

What is the machine doing?

It is usually useful to find out how you machine is coping at the time of the slowness. In particular, the CPU usage and the memory swapping.
In *nix these can be done by using the command:

```
vmstat 1 > vmstat.log
```

This will put all the results into a file called `vmstat.log`.
Make sure you turn this off after, otherwise the file will consume your hard disk.

On Windows, you can use a third party tool or try and get a rough idea from the System Properties window. It is also possible to generate a log of system resources and their use with the Windows Performance Manager.

Garbage Collection

Verbose garbage collection will generate log statements that indicate when java is collecting garbage, how long it takes, and how much memory has been freed. You can enable verbose garbage collection by specifying the -verbose:gc jvm option. To have the garbage collection logs written to a file, use -Xloggc:<filename>

Database Connection Pooling

It is possible that your instance of JIRA maybe running low, or out of connections, this can cause JIRA to hang or perform very slowly. The best way to see this type of behaviour is via a thread dump, below is a sample snippet of a thread dump showing this problem:

```
"Thread-4" prio=6 tid=0x16d85988 nid=0xe20 in Object.wait() [0x17d9f000..0x17d9fce8]
        at java.lang.Object.wait(Native Method)
        - waiting on <0x04139d18> (a org.apache.tomcat.dbcp.pool.impl.GenericObjectPool)
        at java.lang.Object.wait(Object.java:474)
        at org.apache.tomcat.dbcp.pool.impl.GenericObjectPool.borrowObject(GenericObjectPool.java:748)
        - locked <0x04139d18> (a org.apache.tomcat.dbcp.pool.impl.GenericObjectPool)
        at org.apache.tomcat.dbcp.dbcp.PoolingDataSource.getConnection(PoolingDataSource.java:95)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.getConnection(BasicDataSource.java:540)
        at org.ofbiz.core.entity.transaction.JNDIFactory.getJndiConnection(JNDIFactory.java:168)
        at org.ofbiz.core.entity.transaction.JNDIFactory.getConnection(JNDIFactory.java:140)
        at org.ofbiz.core.entity.TransactionFactory.getConnection(TransactionFactory.java:99)
        at org.ofbiz.core.entity.ConnectionFactory.getConnection(ConnectionFactory.java:53)
        at com.atlassian.jira.upgrade.ConnectionKeeper.openConnections(ConnectionKeeper.java:106)
        at com.atlassian.jira.upgrade.ConnectionKeeper.run(ConnectionKeeper.java:87)
```

The lines that indicate this problem are the following:

```
- locked <0x04139d18> (a org.apache.tomcat.dbcp.pool.impl.GenericObjectPool)
        at org.apache.tomcat.dbcp.dbcp.PoolingDataSource.getConnection(PoolingDataSource.java:95)
```

To help alleviate this you can increase your connection pool size. Information on performing this is application server specific, but if you are running Apache Tomcat or JIRA standalone (which uses Apache Tomcat), you can follow the following documentation to adjust the connection

pool size.

Get Help

We are the first to admit we are not experts in everything and, more often than not, there are other customers who have similar set ups and face similar issues. You should ask about your issues on the user forums as you are more than likely to get a quick and knowledgable response. If you still have not got it resolved (even though Atlassians lurk on the forums continuously), please create a Support Request attaching all the information you gathered above with a detailed description of what operation is slow and your environment:

1. Number and speed of CPU's.
2. Is JIRA on the same machine as the Database?
3. What are the `ping` times from the App server to the Database (if it is on a seperate machine)
4. Is the inedx location on the same machine?
5. How much data do you have? (This can be found from `ADMINISTRATION -> System -> System Info`)
6. Cut and copy the contents of the Environment data from `ADMINISTRATION -> System -> System Info`
7. Attach the log file (location of this can be found in `ADMINISTRATION -> System -> System Info`)
8. What actions are slow to perform? Is it a specific page?
9. Do you have any 3rd party plugins installed?

Testing Database Access Speed

We have a utility to test database access speed. Use it if your JIRA installation is running slowly, and you suspect the culprit is database access speed (or you want to rule it out).

You run it on the machine that JIRA is running on. You need to know the path to your JDBC driver jar file and to the atlassian-log-analysis-0.1.1.jar file (once you have downloaded it).

Please note!

This tool only measures retrieval latency of key JIRA database objects. It does not simulate real life activities such as permission checking that may significantly increase processing time. Thus it is not recommended to use this tool as an overall instance performance estimator.

You will need JDK 1.5 or later.

```
java -cp your/path/to/atlassian-log-analysis-0.1.1.jar:/path/to/your/jdbc-driver.jar \
        com.atlassian.util.benchmark.JIRASQLPerformance \
        username \
        password \
        jdbc-url \
        driver-classname \
        > out.txt
```

Note that the use of "\" at the end of the line is the way you can break up a long command line across lines.

You will need to fill in the arguments above to the tool. Unfortunately we have not built a user interface for this tool. Note the `> out.txt` makes the output redirect to a file called `out.txt` - doing this helps remove the time to write to the console from the time measured to the database.

Here is the exact command line I used to run the tool using an MySQL database on my machine:

```
java -cp
./atlassian-log-analysis-0.1.1.jar:/home/idaniel/tools/jdbc_drivers/mysql-connector-java-3.1.12-bin.jar
\
        com.atlassian.util.benchmark.JIRASQLPerformance \
        jirauser \
        jirauser \

'jdbc:mysql://localhost:3306/icfi?autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=UTF8' \
        com.mysql.jdbc.Driver \
        > out.txt
```

The program executes a set of queries 1000 times and then writes a summary (at the bottom of the file).
As a means of comparison, here are the summary results for my machine against an MySQL database on my machine.

I am running a Xeon 2.80GHz with 1.5 GBytes of memory.

```
TOTALS
----    ----    ----    ----    ----
stat    mean    median  min max
----    ----    ----    ----    ----
retrieve-issue  5,338,000   979,000 213,000 46,007,000
get-issue   174,775 93,000  62,000  11,621,000
retrieve-workflow   5,117,153   607,000 341,000 47,738,000
get-workflow    98,996  64,000  40,000  2,962,000
retrieve-custom-field-value 601,093 495,000 316,000 23,082,000
get-custom-field-value  91,246  52,000  37,000  3,453,000
----    ----    ----    ----    ----
All times are in nanoseconds.
```

See Also

- Indexing in JIRA
- Is your JIRA Running Slowly
- Testing Disk Access Speed

Testing Disk Access Speed

We have a utility that you can run to get disk access times. Use it if JIRA or Confluence is running slowly and you suspect that disk access speed is the culprit (or you want to rule it out).

Run it from the machine that JIRA or Confluence is running on, from the directory you have downloaded support-tools.jar to.

```
java -Djava.io.tmpdir=<index directory> -jar support-tools.jar
```

This should give you a result like:

Good

Linux Workstation A

My local machine, which is not slow:

```
TOTALS
----    ----    ----    ----    ----
stat    avg     median  min     max
----    ----    ----    ----    ----
open    75,830  62,000  44,000  2,869,000
r/w     46,407  40,000  35,000  907,000
close   5,751   5,000   4,000   336,000 to
delete  118,942 81,000  65,000  22,864,000
----    ----    ----    ----    ----
All times are in nanoseconds.
```

Linux Workstation B

```
TOTALS
----    ----    ----    ----    ----
stat    avg median  min max
----    ----    ----    ----    ----
open    32,254  29,916  18,610  692,447
r/w 23,676  20,309  17,888  411,218
close   4,857   4,374   3,617   221,829
delete  36,232  34,753  22,000  425,200
----    ----    ----    ----    ----
All times are in nanoseconds.
```

Bad

```
TOTALS
----    ----    ----    ----    ----
stat    avg     median  min     max
----    ----    ----    ----    ----
open    275,886 267,184 246,412 2,544,101
r/w     92,956  81,538  70,727  7,358,896
close   3,194,958       3,177,771       3,076,064       4,331,872
delete  542,650 530,271 514,226 1,771,416
----    ----    ----    ----    ----
All times are in nanoseconds.
```

This site had REALLY fast disks..... with a virus scanner turned on...

Why would disk access be slow?

Some things that can cause slow disk access are:

1. virus checkers
2. remote disks or shared drives
3. synchronization to other machines over a slow network

VMWare

Because we get asked how JIRA performs when running in a VMWare image, here are some results for VMWare. These images were both running on Linux Workstation B.

Windows in VMWare

```
TOTALS
---- ---- ---- ---- ----
stat avg median min max
---- ---- ---- ---- ----
open 593,183 548,114 426,591 8,659,481
r/w 173,372 159,517 10,895 1,954,718
close 174,506 141,079 29,333 17,045,183
delete 405,493 372,673 14,248 7,665,499
---- ---- ---- ---- ----
All times are in nanoseconds.
```

Linux in VMWare

```
TOTALS
----    ----    ----    ----    ----
stat    avg     median  min     max
----    ----    ----    ----    ----
open    84,324  46,000  35,000  16,700,000
r/w     124,550 50,000  41,000  61,272,000
close   17,936  16,000  14,000  961,000
delete  64,528  49,000  36,000  3,603,000
----    ----    ----    ----    ----
All times are in nanoseconds.
```

This shows that, at least in our testing, you should not run JIRA in VMWare, as disk access, is much slower than on a native operating system installation.

See Also

- Indexing in JIRA
- Is your JIRA Running Slowly

Windows Performance Manager

This document is a guide on how to capture server performance information on systems running the Windows operating system. More information on generating similar logging information within *nix systems is available here.

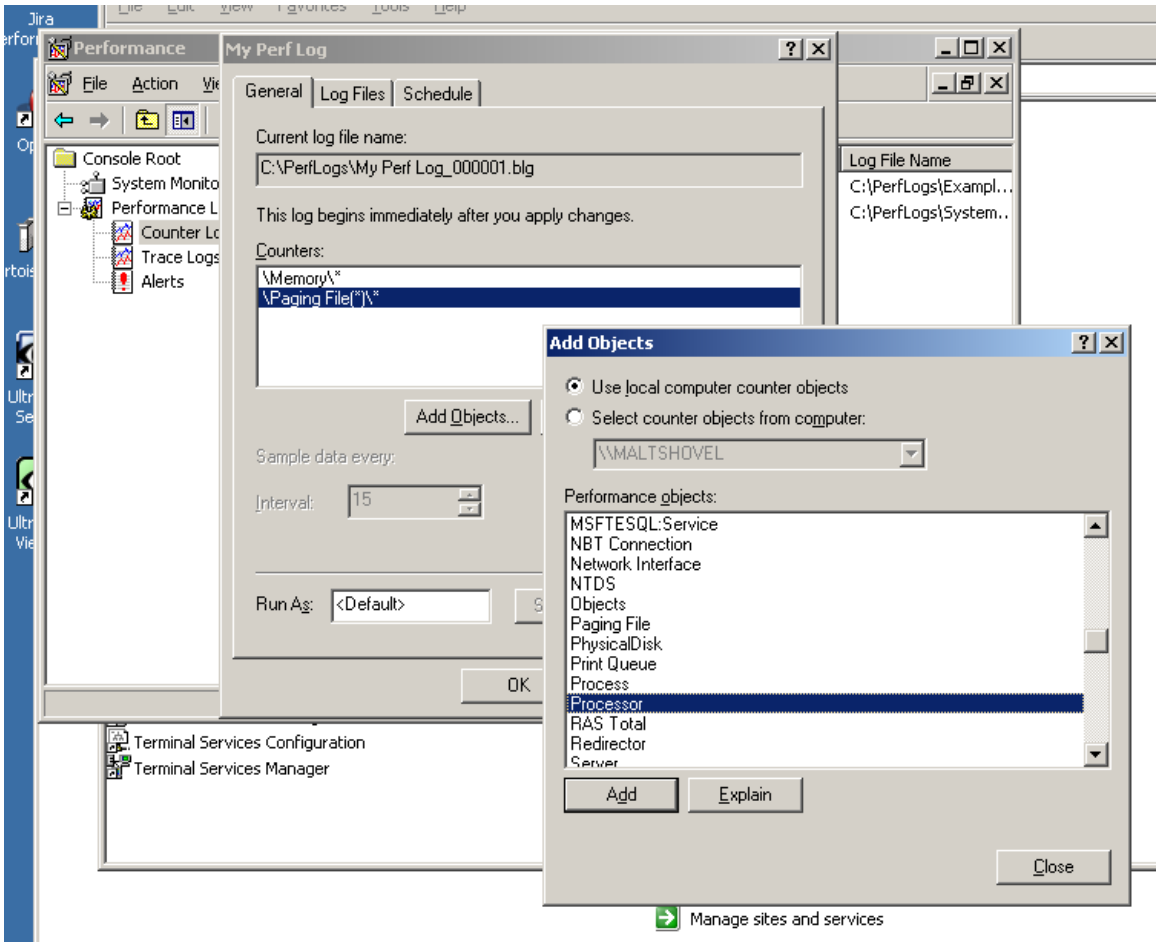
Windows Performance Manager

Windows includes a tool called **Windows Performance Manager** - a utility used to capture performance data on the host machine and the applications running on it.

Executing this tool on the server running JIRA, it is possible to capture performance related information that can help diagnose performance related problems.

Running Windows Performance Manager

1. Start **Windows Performance Manager** by navigating to:
    - Control Panel -> Administrative Tools -> Performance
2. Create a new log setting:
    - This is achieved by *right-clicking* on **Counter Logs** under **Performance Logs and Alerts**.
    - Specify a name and location for the log.
3. Select the following objects to add to the log:
    - Memory
    - Paging File
    - Processor
4. Set the interval to be **10 seconds**
5. Click on the **Log Files** tab
6. Select the following **Log File type**:
    - Text File (Comma delimited)
7. Uncheck the option to **End File Names With ...**
8. Click on the **Scheule** tab
9. Schedule the logs to run for an appropriate time



Duration

In order to capture useful data that is indicative of normal usage, it is recommended to run the performance manager for at least one day of normal server use.

It is also necessary to manually note when the server becomes unresponsive or performance degradation is experienced - in order to correlate the generated logs with actual times of unresponsiveness.

Logs

Please forward the generated logs, including details of when the system became unresponsive and a description of the performance degradation (e.g. *creating an issue is extremely slow* or *a bulk operation does not appear to complete*) to the Atlassian Support team by raising an issue at:

https://support.atlassian.com

Profiling Memory and CPU usage with YourKit

YourKit is a commercial Java profiling tool that allows to generate CPU and Memory profiles of running applications with no noticeable overhead. The profiling libraries of YourKit are free to use and redistribute, the console used to analyze the generated snapshots requires a commercial license to run.

If your JIRA instance seems to be running quite slowly or is utilizing too much memory (you are seeing OutOfMemoryErrors in log files), then generating a CPU or Memory profile can provide valuable insight into your running instance. Generating one of these snapshots and attaching the result to a JIRA support issue can help our support staff get to the bottom of some of the more nasty issues which are lurking out there.

This document will guide you in configuring JIRA to run with the YourKit profiler agent enabled. If you run into any troubles with this guide please look at the troubleshooting guide.

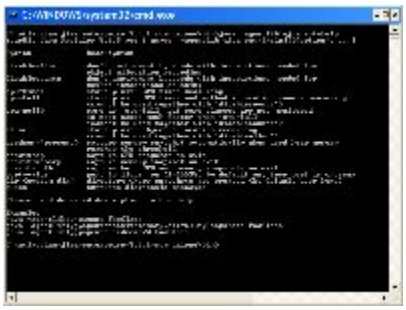
> Confluence also has a plugin to profile Confluence's memory and CPU usage with YourKit.

Make profiler agent library accessible to the Java Virtual Machine

In order for YourKit to be able to generate CPU and Memory profiles it needs to have a hook into the JVM which is running JIRA. To make the agent library accessible to the JVM you will need to do the following:

On Windows

- Download the attached files yjpagent.dll and yjp.ini. Save them to `c:\windows\system32` on the machine where JIRA is running (this seems to be the only directory the Windows Service picks them up from). Test that the agent can be found by opening a Command Prompt and running `java -agentlib:yjpagent=help` (assuming JDK 1.5; see below for 1.4):



On Linux

- Download the attached file libyjpagent.so. Save this file to a directory in the machine where JIRA is running (eg. /home/youruser/yourKitAgent).
- Include the directory in which you saved the file into your LD_LIBRARY_PATH. (NOTE: you do not need to do this for your global environment, it is only needed in the environment from which you start JIRA). You can modify this by setting and exporting the variable LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/youruser/yourKitAgent.
- If you are running JIRA Standalone, you can set this up by modifying the {PATH TO STANDALONE}/bin/setenv.sh file and adding a line that looks like:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/youruser/yourKitAgent
export LD_LIBRARY_PATH
```

Make the JVM running JIRA use the YourKit Agent

General Information

Once you have made the library accessible to the JVM you need to add some JVM parameters that tells the JVM to use the Yo agent.

Java applications all start by running a command

```
java <args>
```

where `<args>` varies. To include the YourKit agent in the running JVM, you must add a JVM parameter. The form of the parameter depends on your Java version.

If you are running JIRA with Java 1.3 or 1.4, use -Xrunyjpagent parameter:

E.g., java `-Xrunyjpagent` <args>

If you are using Java 5.0 (1.5), use -agentlib:yjpagent parameter:

E.g., java `-agentlib:yjpagent` <args>

`-Xrunyjpagent` or `-agentlib:yjpagent` has additional options. In most cases there's no need to use them.

The options are comma separated: `-Xrunyjpagent[:<option>, ...]` or `-agentlib:yjpagent[=<option>, ...]`.

To generate snapshots from a running JIRA instance the only required option is 'dir'. When snapshots are taken they will be written to the path specified by the 'dir' option on the machine where JIRA is running.

If JIRA is running out of memory (you are seeing OutOfMemoryError in the logs) it is recommended that you set the 'onexit=memory' option. This will generate a memory snapshot when the JVM is exiting, even if JIRA does not respond and appears hung.

Eg. to profile the application and have it write the profile snapshots to /path/to/write/snapshots while running a 1.3 or 1.4 JVM, run

```
java -Xrunyjpagent:onexit=memory,dir=/path/to/write/snapshots <args>
```

Eg. to profile the application and have it write the profile snapshots to /path/to/write/snapshots while running a 1.5 JVM, run

```
java -agentlib:yjpagent=onexit=memory,dir=/path/to/write/snapshots <args>
```

Please take careful notice of the difference in syntax between the -Xrunyjpagent:onexit=memory,dir=/path/to/write/snapshots and -agentlib:yjpagent=onexit=memory,dir=/path/to/write/snapshots. Specifically notice that the arguments in the Xrunyjpagent are started by a **:** and that the arguments in the agentlib:yjpagent are started by a **=**.

If you are using 1.5 JVM to run JIRA, please use the `-agentlib:yjpagent` parameter. Using `-Xrunyjpagent` with JVM 1.5 appears to start up fine, but can result in errors while JIRA is running.

Configuring Application Server

JIRA is run in an application server (Tomcat, Orion, Websphere, Weblogic, JBoss etc), and these app servers usually have scripts wrapping the actual `java` call. The `-Xrunyjpagent` parameter that is discussed above, is usually passed into these scripts through an environment variable.

JIRA Standalone as a Windows Service

If you are running JIRA Standalone, as a Windows Service (recommended), you will need to set the parameter by running:

```
tomcat5 //US//JIRA ++JvmOptions=
"-agentlib:yjpagent=onexit=memory,dir=c:\atlassian-jira-enterprise-3.6.5-standalone\logs"
```

(specify the correct path to the `logs` directory)

Test that memory profiles are being written by starting and stopping the service. When the service is stopped, it should pause while a `*.memory` file is written to the specified directory.

If you don't see the `.memory` file being created, make sure the directory you specified is the `logs` directory inside Tomcat. Other directories would not work for me. Generating the memory dump generally seemed flakey, with dumps occasionally failing on the first service restart and then succeeding, or the service shutdown hanging and needing to be shut down twice.

For more information on setting JVM parameters when running as a Windows Server please see JIRA documentation).

JIRA Standalone on Windows (started with `startup.bat`)

If you are running JIRA Standalone, on Windows, then you can set this up by modifying the {PATH TO STANDALONE}\bin\setenv.bat file and adding a line that looks like:

```
set JAVA_OPTS=%JAVA_OPTS%
-agentlib:yjpagent=onexit=memory,dir=c:\atlassian-jira-enterprise-3.6.5-standalone\logs
```

(assuming JDK 1.5; specify the correct path to your `logs` directory)

JIRA Standalone on Linux

If you are running JIRA Standalone, on Linux, then you can set this up by modifying the {PATH TO STANDALONE}/bin/setenv.sh file and adding a line that looks like:

```
JAVA_OPTS="-Xrunyjpagent:onexit=memory,dir=/home/detkin/snapshots $JAVA_OPTS"
```

Other Application Servers

For help setting this up in other application servers please see this page.

Including the jsp and jar file in your JIRA instance

Once you have configured the JVM running JIRA to enable the YourKit agent and have correctly configured the 'dir' option, you need to take steps to allow you to generate profiling snapshots from within JIRA. This is a simple process of making certain that a jar file is available to JIRA and that a jsp is included in the JIRA administration section.

You can manage the generation of CPU and Memory profiling snapshots via the JIRA administration section.

JIRA Version	Action
Prior to 3.7	• download the file manageyourkitprofiling.jsp • copy the file into the {YOUR JIRA INSTALL}/secure/admin directory
3.7 - 3.8.1	This version ships with an older version and will need to be replaced. • download the file manageyourkitprofiling.jsp • **replace** the file into the {YOUR JIRA INSTALL}/secure/admin directory
3.9 or greater	JIRA now ships with the latest version of this file

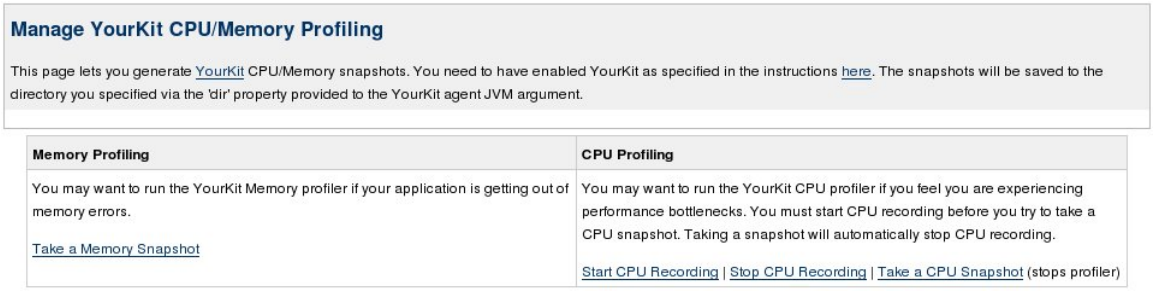
You will also need to include the YourKit jar into your instance of JIRA:

- download the file yjp-controller-api-redist.jar
- copy the file into the {YOUR JIRA INSTALL}/WEB-INF/lib directory

After making certain that the jsp is included and that the jar file has been put into the correct place you will need to restart your instance of JIRA. You should now be able to browse to http://<YOUR JIRA>/secure/admin/manageyourkitprofiling.jsp and you should see a screen that looks like the image in the next section. If instead you see an error please read through the troubleshooting section.

Manage profiling from JIRA administration section

Now that you have setup the YourKit agent and made certain that the jsp and jar files are correctly configured in JIRA you are ready to start managing Memory and CPU snapshots. If you browse to http://<YOUR JIRA>/secure/admin/manageyourkitprofiling.jsp you will see the following:



The page gives you two options, Memory Profiling and CPU Profiling.

Memory Profiling

At any point you can click the 'Take a Memory Snapshot' link and this will generate a memory snapshot of the application at that moment. This function can take a few seconds to complete and will slow JIRA down while executing. You should see a message like: *Successfully generated a memory snapshot to: /home/detkin/snapshots/snapshots-2006-08-16.memory please attach this file to a support request.* once it has completed. The JIRA support team should inform you as to the times when they would like to to get a memory snapshot from the running application.

If you run into situations where JIRA is unresponsive and you are not able to navigate to the JSP to trigger a memory snapshot, please ensure that you start JIRA with the `onexit=memory` parameter (as discussed in the above section) and simply shutdown JIRA. Before shutting down a memory snapshot will be created.

CPU Profiling

To generate a CPU snapshot you need to 'Start CPU Recording'. This tells the YourKit profiler that it should start keeping track of the CPU information. If at any time you decide you want to stop the CPU recording, but not take a snapshot all you need to do is click the 'Stop CPU Recording' link. To take a CPU snapshot you need to have started the CPU recording and when you want to take the snapshot just click the 'Take

a CPU Snapshot' link. This will generate the snapshot and automatically stop the CPU recording. When completed you should see a message like: *Successfully generated a cpu snapshot to: /home/detkin/snapshots/snapshots-2006-08-16.cpu please attach this file to a support request.* Generating a CPU snapshot does not take as long as generating a memory snapshot.

There will be a slow down

CPU profiling does not come with out a cost. There is an overhead that will slow the system down. This is best used to profile individual requests. E.g.

1. Start CPU Profiling
2. Perform an Issue Navigator Search
3. Create a Snapshot

Do not leave running for a long time!

Setting Up YourKit Agent Parameter

Here is a list of app servers, and how to set the parameter for each:

AppServer	Startup script	Variable to edit	Example
Orion	n/a	n/a	`java -Xrunyjpagent:onexit=memory,dir=/path/ -jar orion.jar -userThreads`
Tomcat (and JIRA Standalone)	bin/setenv.sh, bin/setenv.bat or as a Windows Service	JAVA_OPTS	`set JAVA_OPTS=-Xrunyjpagent:onexit=memory,dir=/` (Windows), or `export JAVA_OPTS=-Xrunyjpagent:onexit=memory,dir=/` (Unix)
Weblogic	$BEA_HOME/domain/<domain>/startWebLogic.sh	JAVA_OPTIONS	`export JAVA_OPTIONS="$JAVA_OPTIONS -Xrunyjpagent:onexit=memory,dir=/path/to/wr`
JBoss 3.2.x	bin/run.sh	JAVA_OPTS	`export JAVA_OPTS=-Xrunyjpagent:onexit=memory,dir=/`
Resin 3.0.x	bin/httpd.sh	n/a	Start with httpd.sh -J-Xrunyjpagent:onexit=memory,dir=/pa

Troubleshooting Profiling Memory and CPU usage with YourKit

If some of the steps in the Profiling Memory and CPU usage with YourKit page are skipped or not correctly followed you could run into some trouble. Here are some things to look out for:

'Oops - an error has occurred' when viewing manageyourkitprofiling.jsp

If you encounter an error page when trying to view the manageyourkitprofiling.jsp that looks like this:

> *org.apache.jasper.JasperException: Unable to compile class for JSP Generated servlet error: Only a type can be imported. com.yourkit.api.Controller resolves to a package An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method stopMemoryRecording An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method stopCPURecording An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method dumpMemoryInfo An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method dumpCPUInfo An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method startCPURecording An error occurred at line: 5 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved (or is not a valid type) for the parameter controller of the method startMemoryRecording An error occurred at line: 109 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved or is not a type An error occurred at line: 109 in the jsp file: /secure/admin/manageyourkitprofiling.jsp Generated servlet error: Controller cannot be resolved or is not a type*

It is likely that you have not correctly included the yjp-controller-api-redist.jar in your JIRA WEB-INF/lib directory or you have not restarted JIRA since you have placed the jar there. Please verify that the jar is present and that JIRA has been restarted and try again.

HTTP Status 404 - /secure/admin/manageyourkitprofiling.jsp

If you encounter this error when trying to view the manageyourkitprofiling.jsp then the likely cause is that you have not correctly placed the manageyourkitprofiling.jsp into your instance of JIRA. Please verify that the file is present in the correct place and try again.

Error occurred during initialization of VM Could not find -Xrun library: libyjpagent.so

If you encounter this error then it is likely that you have not made the profiler agent library accessible to the Java Virtual Machine correctly while trying to use the -Xrunyjpagent JMV parameter. Please re-read the section on this and ensure that you have set this up correctly and try again.

Error occurred during initialization of VM Could not find agent library on the library path or in the local directory: yjpagent

If you encounter this error then it is likely that you have not made the profiler agent library accessible to the Java Virtual Machine correctly while trying to use the -agentlib:yjpagent JMV parameter. Please re-read the section on this and ensure that you have set this up correctly and try again.

'Error: An unknown error occurred, caused by: To profile application, you should run it with the profiler agent' shown on manageyourkitprofiling.jsp page

If you see this error when viewing the manageyourkitprofiling.jsp page then it is likely that you have not instructed the JVM running JIRA to use the YourKit Agent. Please re-read the section on this and ensure that you have set this up correctly and try again.

Slow JIRA Operations

Slow JIRA Operations

Certain operations in JIRA can take quite some time to do. These operations include:

1. Deleting a project which contains many issues
2. Changing a Workflow Scheme for a large project in the Enterprise Edition, or activating a workflow in the Professional Edition
3. Running the Integrity Checker
4. Performing Bulk Operations (e.g. Bulk Edit, Bulk Transition) on a large set of issues

The duration of each operation depends on the number of issues it has to work on. We will be looking at improving the performance of these operations in JIRA in the next few months.

If you need to delete a large project, change project's workflow scheme (or activate workflow), or run the integrity checker, please do so during times when JIRA is not being heavily used.

With Bulk Operations, please either perform these during off-peak periods or choose to work with smaller sets of issues.

Usage FAQ

JIRA Administrators FAQ

Usage FAQ

- How do I unlock my JIRA home directory?
- Changing the Project Key
- Mail error - Unable to relay
- Importing user from LDAP
    - Neat JIRA LDAP tricks — Gianugo has an interesting blog up about how to take the pain out of migrating users from LDAP
- JIRA Standalone quick source modifications
- Commonly Asked CSV Questions and Known Issues
- Troubleshooting Issue Creation Via Email — Use this excerpt to provide a brief explanation of what this how-to will do.
- How to clear the resolution field when the issue is reopened
- How to re-order the list of issue operation in an issue
- How to resize Free Text Field customfield
- Using validators to make custom fields required on transition screens — Use the Fields Required workflow validator that is packaged in the JIRA Suite Utilities.
- Project-specific email templates
- How to configure sub-task to have a specific screen?
- User access logging — How to track user actions with page access logging
- Connecting to SSL services
- Editing a custom field option
- Allow editing of Closed Issues
- Updating your JIRA License Details

- Linking to local file under Firefox
- Creating Issues via direct HTML links
- How to create a download link to a file
- Current Reporter Browse Project Permission
- How to display custom field of the sub-task in the parent issue screen?
- How the CreateOrCommentHandler works?
- International characters in notification email subjects
- Sending a JIRA data backup to support
- Letting customers only create issues — This page describes a minor JIRA modification which redirects users to an arbitrary page after creating issues (and potentially other operations). It is mainly of interest to JIRA **Professional** and **Standard** users.
- Asking for an attachment on the Create Issue page
- Limiting the number of issues returned from a search view such as an RSS feed
- Anonymising data
- Single Sign-on
- Scheme Entity Relations Map — This diagram illustrates the relationships between various JIRA entities (Enterprise Edition) and schemes.
- Custom field column not visible in Issue Navigator
- XML format for import & export files
- Configuring project specific security
    - Controlling project visibility
    - Using Project Level Security with Project Roles — This tutorial (for JIRA Enterprise Customers) provides a step-by-step guide for creating project roles and using them in an issue security scheme. We recommend creating a test project and two test users for this tutorial.
    - Using Project Level Security with User Groups
- Changing Custom Field Types
- Re-order workflow transactions
- Creating an Unassigned Issue
- Workflow Properties
- Editing the work log
- Field Layout Schemes in JIRA 3.x
- How to display a different format for the Number customfield
- How to remove 'NONE' from a select list custom field
- Plans for JIRA's LDAP integration
- What are the differences between JIRA Enterprise, Professional and Standard editions?
- Obsolete guides — Administration guides applying to older versions of JIRA
    - Changing scope of existing custom fields
    - Large text fields in Oracle
    - Removing invalid characters from XML backups — In older versions of JIRA it was possible to cut & paste text containing control characters into JIRA issue fields. This causes problems, because JIRA's backup format is XML, and XML does not allow for the storage of most control characters .
- Changing JIRA's log output — A common task when identifying JIRA problems is to turn up the log level, to get more debug-level logs.
    - Logging email protocol details — How to enable email protocol (SMTP, IMAP, POP) logging in JIRA
- Changing the username in JIRA
- Changing the Due Date input format
- Extra CSV Import Features
- Allowing users to create issues anonymously
- Deleting a user account
- Changing the Issue Key format
- QuickSearch guesses the issue key prefix (sometimes)
- Outward Link Description and Inward Link Description
- What does JIRA mean?
- Can I store customer details, like company, address and contact information, in JIRA?
- Words ignored when searching
- Using JIRA to Manage reusable modules
- Retrieving the JIRA Administrator
- Merging 2 JIRA instances
- Group Name Guidelines for JIRA
- Where are the JIRA logs? — A decision tree for finding JIRA logs on your system
- Escalating issues (or sending email notifications) when the set turnaround time is exceeded
- Why doesn't JIRA have a Severity field like Bugzilla?
- How to search by number range in the Issue Navigator
- Troubleshoot Mail Notification Problems
- Resolved issues appearing in Open issues filters
- Parsing utf-7 emails
- Adding custom content to the front page
- Changing the default session timeout
- How is JIRA pronounced?
- Translating JIRA
- How to ensure the Road Map tab is visible
- How to have long component version names display properly in the Issue Navigator
- Login problems
- We already have users & groups defined elsewhere - can JIRA make use of these?

- Associating a new screen with issue operations — When you spend some quality time creating a new screen with custom fields, you want to make people use it. This quick and dirty how-to helps make your screen implementation dreams come true!
- CVS ssh Jira Integration
- JIRA as a Support System
    - Jelly Escalation
    - Simple Escalation
- Changing the default attachment size limit
- Importing data

How do I unlock my JIRA home directory?

Your JIRA home directory can become locked if you accidentally configure two JIRA instances to use the same directory or your JIRA instance terminates abnormally and fails to unlock the JIRA home directory. This will prevent your JIRA instance from running.

To **unlock your JIRA home directory**, follow the instructions below:

1. Verify that no running JIRA instances are configured to use the specified JIRA home. If there are running JIRA instances that are configured to use the specified JIRA home, shut them down and change the JIRA home directory to a different location.
2. Remove the file **.jira-home.lock**. This file is located in your JIRA home directory.
3. Restart JIRA.

Changing the Project Key

It is not currently possible to change the project key through JIRA's interface. The best way to do this is to:

1. Export your JIRA data to XML.
2. Go through the file and replace the instances of the project's key:
    - in the 'key' attribute of the Project entity
    - in the 'key' attribute of all Issue entities.
3. Search for the project key in the whole XML file. You should not find too many references. Change any that you find.
4. Rename all attachment folders, as the folder name depends on the project key. This includes one folder for each issue with an attachment, plus one top-level folder for the project.
5. Reimport your data.
6. Reindex the data by navigating to Administration -> System -> Indexing and selecting '**Re-Index**'.

Mail error - Unable to relay

I'm getting exceptions like "SMTPAddressFailedException: 550 5.7.1 Unable to relay for XXX@XXX". What does this mean?

The "Unable to relay" error means that your mail server doesn't allow relaying for the e-mail address that you are using for your SMTP server. (see http://www.chilkatsoft.com/faq/Smtp550.html). Please try getting your mail server admin to enable relaying for your e-mail address or use another address that has relaying enabled.

You can get more help on changing the e-mail addressed used by JIRA here.

Importing user from LDAP

JIRA's LDAP integration currently requires users to have accounts both in LDAP and in JIRA. For instance, if a user is added to LDAP, they will have no access to JIRA until someone creates them a JIRA username (and assigns it to groups).

The attached tool searches LDAP for user accounts, and generates a JIRA Jelly script which will create a JIRA user account for each LDAP account. Typically one would use this tool when first installing JIRA, to bulk-create JIRA users matching each LDAP account.

How to use

Download the current binary distribution. Alternatively, if you are Java-literate and keen, all current distributions contains source distributions. You can also get the source from Subversion at http://svn.atlassian.com/svn/public/contrib/jira/jira-ldap-userimporter/trunk.

Create a file, `ldap.properties`, to specify your LDAP server's details. If you are unsure of these, first test with an LDAP browser (there are many LDAP browsers available on the internet, you can try using this LDAP browser or search for an alternative one). Here is a `ldap.properties` configured for use against a local OpenLDAP directory:

```
# Configuration file for JIRA's LDAP user importer

# URL of your LDAP server, Eg:
java.naming.provider.url=ldap://192.168.0.74

# Username and password of account that has privileges to loop through all users, eg:
java.naming.security.principal=cn=admin,dc=atlassian,dc=com
java.naming.security.credentials=secret

# LDAP node below which we should search, eg:
searchbase=ou=People,dc=atlassian,dc=com

# LDAP query run below 'searchbase' identifying user nodes, eg:
query=(objectclass=*)

# Name of record in nodes which should become the username in JIRA, eg:
username_attr=uid

# Record that contains the user's full name. When commented out, defaults to username_attr value. Eg:
fullname_attr=cn

# Record that specifies the user's email address. When commented out, username_attr value with
email_suffix appended will be used
#email_attr=
email_suffix=@atlassian.com


# Generally you don't want to touch this
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
```

Once you have created `ldap.properties`, run `java -jar jira-ldap-userimporter-1.1.jar`. If you have the ldap.properties details correct, this command will result in XML text being printed to the console. Eg:

```
~/jira-ldap-userimporter-1.1$ java -jar jira-ldap-userimporter-1.1.jar
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreateUser username="nobody" password="nobody" confirm="nobody" fullname="nobody" email=
"nobody@atlassian.com"/>
  <jira:CreateUser username="jturner" password="jturner" confirm="jturner" fullname="Jeff Turner"
email="jturner@atlassian.com"/>
  <jira:CreateUser username="anonymous" password="anonymous" confirm="anonymous" fullname="anonymous"
email="anonymous@atlassian.com"/>
  <jira:CreateUser username="devuser" password="devuser" confirm="devuser" fullname="devuser" email=
"devuser@atlassian.com"/>
</JiraJelly>
```

This text can now be redirected to a file, and fed to the Jelly Runner (see the Jelly docs).
However, first make sure that **LDAP password checking is disabled** (ie. there is no LDAPCredentialsProvider section in osuser.xml), otherwise the Jelly script will fail, claiming these users already exist.

Feedback? Problem

Please raise a Support Request.

Exceeding your user limit on import?
If you are getting a `LimitExceededException`, you may find these instructions from one of our customers helpful.
Thanks to Ricardo Sueiras

Version History

Version	Comments
1.1	Username is converted to lowercase automatically

1.0	First Released

Neat JIRA LDAP tricks

Gianugo has an interesting blog up about how to take the pain out of migrating users from LDAP

JIRA Standalone quick source modifications

To make any substantial modifications or additions to JIRA's source, you should read the documentation on building JIRA from source. This implies building a .war, and deploying this to your own app server. However to make a small code modification/addition to a JIRA Standalone, there is a quicker way:

1. Download and install Apache Ant
2. Find JIRA Standalone's `external-source` directory, and create a `src` directory beneath.
3. In `external-source/src`, add your new .java file, or modified copy of an existing JIRA .java file.
4. Run 'ant' in `external-source`

For instance, to modify the FogBugz importer to use a different user mapper (`sed` replaces references from the default mapper to our `CustomUserMapper`, which can then be edited for custom behaviour):

```
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone$ mkdir external-source/src
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone$ cp
../atlassian-jira-enterprise-3.6.5-source/jira/src/java/com/atlassian/jira/imports/fogbugz/FogBugzConfigB
external-source/src/
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone$ cp
../atlassian-jira-enterprise-3.6.5-source/jira/src/java/com/atlassian/jira/imports/csv/mappers/FirstIniti
external-source/src/CustomUserMapper.java
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone$ sed -ie
's/FirstInitialFirstNameUserMapper/CustomUserMapper/g' external-source/src/*.java
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone$ cd external-source/
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone/external-source$ ant
Buildfile: build.xml

prepare:
mkdir Created dir: /tmp/atlassian-jira-enterprise-3.6.5-standalone/external-source/etc
mkdir Created dir: /tmp/atlassian-jira-enterprise-3.6.5-standalone/external-source/lib

compile:
javac Compiling 2 source files to
/tmp/atlassian-jira-enterprise-3.6.5-standalone/atlassian-jira/WEB-INF/classes
javac Note: /tmp/atlassian-jira-enterprise-3.6.5-standalone/external-source/src/FogBugzConfigBean.java
uses unchecked or unsafe operations.
javac Note: Recompile with -Xlint:unchecked for details.

BUILD SUCCESSFUL
Total time: 2 seconds
jturner@teacup:/tmp/atlassian-jira-enterprise-3.6.5-standalone/external-source$
```

Commonly Asked CSV Questions and Known Issues

- Overview
- Commonly Asked Questions
  - The importer simply doesn't work on my CSV file!
  - The importer fails at date fields, why?
  - Why does the importer always ask me to map values to column (at Step 3 of 5)?
  - Why doesn't the importer recognize the extra columns for comments?
- Known Issues
  - Why couldn't I import issues with other languages?
  - Why couldn't I import from cascading select fields?
  - Why couldn't I create subtasks?
  - Why couldn't I import component/version Custom Fields?

Overview

This page answers some of the commonly asked CSV questions our technical support staffs have encountered. If you are not able to find an

answer from this page and our issue tracker, feel free to create a support issue.

For more information about JIRA's built-in CSV Importer, please refer to this page.

Commonly Asked Questions

The importer simply doesn't work on my CSV file!

Please make sure that it is a valid and not-bad-formatted CSV file. You should be able to spot this with by turning on detailed logging and profiling . Also, please double check your configuration file and ensure that it's properly configured, e.g. exact delimiter, date format, etc.

The importer fails at date fields, why?

If you are seeing error message similar to this:

```
[00:55:28] FAILED: Customfield value 01/Nov/06 12:00 AM is invalid
[00:55:28] com.atlassian.jira.issue.customfields.impl.FieldValidationException: Invalid date format.
Please enter the date in the format "MMM/dd/yy".
at
com.atlassian.jira.issue.customfields.converters.DatePickerConverter.getTimestamp(DatePickerConverter.jav
com.atlassian.jira.issue.customfields.impl.DateCFType.getSingularObjectFromString(DateCFType.java:46)
at
com.atlassian.jira.imports.importer.impl.DefaultJiraDataImporter.importIssues(DefaultJiraDataImporter.jav
com.atlassian.jira.imports.importer.impl.DefaultJiraDataImporter.doImport(DefaultJiraDataImporter.java:10
com.atlassian.jira.imports.importer.impl.ImporterThread.run(ImporterThread.java:21)
```

There are a few possible reasons:

- Date format is not correctly set in the import configuration file
- Date Picker and Date Time Picker formats are not consistent, e.g.

```
jira.date.picker.java.format=dd/MMM/yy
jira.date.time.picker.java.format=MMM/dd/yy hh:mm a
```

should be corrected to,

```
jira.date.picker.java.format=dd/MMM/yy
jira.date.time.picker.java.format=dd/MMM/yy hh:mm a
```

Why does the importer always ask me to map values to column (at Step 3 of 5)?

It is because you have selected *Map Field Value* for the particular columns. To use the values from the CSV, you need just to map the column to the *Corresponding JIRA field*, otherwise, select the *Map field value* checkbox.

Why doesn't the importer recognize the extra columns for comments?

This is actually a known issue being raised at JRA-10945. Please try the workaround as suggested in the issue. Alternatively, you can write your own comment mapper for more flexibility to meet your requirements.

Known Issues

Why couldn't I import issues with other languages?

Setting the **`-Dfile.encoding=utf8`** parameter *should be* able to fix your problem. Otherwise, you may like to vote for this issue at JRA-11614.

Why couldn't I import from cascading select fields?

This is an open issue being tracked at JRA-5775. Feel free to comment and vote on it.

Why couldn't I create subtasks?

This is an open issue being tracked at JRA-6323. Feel free to comment and vote on it.

Why couldn't I import component/version Custom Fields?

This is an open issue being tracked at JRA-10933. Feel free to comment and vote on it.

Troubleshooting Issue Creation Via Email

Use this excerpt to provide a brief explanation of what this how-to will do.

1. Is the message reaching the e-mail account?

Have your mail server administrator confirm that e-mail sent to the account JIRA is using is successfully reaching the account's Inbox.

2. Is the Create Or Comment Handler service configured correctly in JIRA?

Please review this guide to confirm this:

Creating Issues and Comments from Email

3. Are permissions set properly?

Does the user submitting the issue have Create Issue permissions in the Permission Scheme? If you are having troubles adding comments, make sure your Issue Security Scheme is not restricting the user's access to the issue.

4. Still not working? Enable debug logging in JIRA:

First, we need to change the com.atlassian package from the *WARN* logging level to *DEBUG*. This can be done from the following menu:

Administration -> System -> Logging & Profiling -> Click Edit next to the com.atlassian package

5. Send two e-mails to the email address that JIRA is checking for new issues and comments. Wait 5 minutes and then submit a support request that includes the JIRA logs.

This can be done from the following menu:

Administration -> System -> Support Request

Remember to check the *Attach JIRA logs* box! Also, please note the e-mail address being used for testing and copy/paste the JIRA service settings for this Create Or Comment handler:

Administration -> System -> Services

Example of Service Info

handler: Create Or Comment Handler
popserver: POP server - JSP
handler.params: project=JSP, issuetype=1, createusers=true, stripquotes=true, bulk=forward
usessl: No SSL
forwardEmail: jira-support@atlassian.com

6. Change com.atlassian back from *DEBUG* to *WARN*.

Administration -> System -> Logging & Profiling -> Click Edit next to the com.atlassian package

WARNING: Leaving com.atlassian in debug mode will result in VERY large log files!

Note. If you want to logging on a protocol level (IMAP, POP3 or SMTP), please refer to Logging email protocol details

How to clear the resolution field when the issue is reopened

In the default JIRA workflow, issues have their resolutions cleared upon re-opening an issue. This is important because many reports/filters could be inspecting for the presence of a *Resolution* to be considered resolved.

The *Resolution* field is typically cleared by setting a post-function in the workflow transition you'd like to have this occur.

How to re-order the list of issue operation in an issue

To re-order the issue operation list, the 'order' value at the following file needs to be edited:

```
/atlassian-jira/WEB-INF/classes/system-issueoperations-plugin.xml
```

For example, change the 'order' value for 'Comment on this issue' operation from 50 to 10. By doing so, it will place this operation at the top of the issue operation list:

```
<issue-operation key="comment-issue" i18n-name-key="admin.issue.operations.plugin.comment.issue.name"
name="Comment on this issue" class="com.atlassian.jira.issue.operations.CommentIssueOperation"
state='enabled'>
        <resource type="velocity" name="view" location="templates/plugins/operations/commentissue.vm"
/>
        <order>10</order>
</issue-operation>
```

Please restart JIRA after the modification. Do note that this modification will be applied to all the projects in JIRA.

How to resize Free Text Field customfield

You can customise the size of customfield (Free Text Field) at
`%JIRA_HOME%/atlassian-jira/WEB-INF/classes/templates/plugins/fields/edit/edit-textarea.vm`

Using validators to make custom fields required on transition screens

Use the Fields Required workflow validator that is packaged in the JIRA Suite Utilities.

Project-specific email templates

Using email notifications, can separate templates be setup for projects or events?

Unfortunately templates are currently global. We anticipate adding this feature to JIRA Enterprise in future.

How to configure sub-task to have a specific screen?

By configuring a custom Issue Type Screen Scheme, it could able to have a specific screen for sub-task issue type. For example:

1. Create a screen via *Administration -> Issue Fields -> Screens* (e.g. Sub-task screen)
2. Create a Screen Scheme via *Administration -> Issue Fields -> Screen Schemes* (e.g. Sub-task Screen Scheme)
3. Configure this newly created screen scheme to have a 'sub-task screen' when creating issue
4. Create a Issue Type Screen Scheme via *Administration -> Issue Fields -> Issue Type Screen Schemes*5. Configure this newly created Issue Type Screen Scheme to have a 'Sub-task Screen Scheme' for 'sub-task' issue type
6. Associate this Issue Type Screen Scheme with the project

For more information on Issue Type Screen Scheme, please refer to this documentation:

- http://www.atlassian.com/software/jira/docs/latest/issuetype_screenschemes.html

Searching JIRA Knowledge Base

User access logging

Occasionally one wishes to get an overall picture of *which* users are accessing *which* pages in JIRA. Application servers are able to log the requested URL, but (it seems) they cannot determine the currently logged in user (probably because they run before the Seraph filter has a chance to set request.getRemoteUser()).

Similar to Confluence, JIRA 3.3 and above has a built-in URL logging mechanism, which shows the user and URL invoked:

```
2005-07-15 18:10:19,276 INFO [JIRACOM:jira.web.filters.AccessLogFilter] AccessLogFilter initialized.
Format is: <user> <url> <starting memory free (kb)> +- <difference in free mem (kb)> <query time (ms)>
2005-07-15 18:10:20,523 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/admin/jira/ViewLogging.jspa 109687-110 35
2005-07-15 18:10:20,523 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/admin/jira/ViewLogging.jspa 109687-110 35
2005-07-15 18:15:59,709 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/IssueNavigator.jspa 115762-3000 2385
2005-07-15 18:15:59,709 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/IssueNavigator.jspa 115762-3000 2385
2005-07-15 18:26:53,916 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/logout 108756 1
2005-07-15 18:26:53,916 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/logout 108756 1
2005-07-15 18:26:54,306 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/Logout!default.jspa 116660-2809 279
2005-07-15 18:26:54,306 INFO [JIRACOM:jira.web.filters.AccessLogFilter] joe
http://localhost:8090/jira/secure/Logout!default.jspa 116660-2809 279
2005-07-15 18:27:01,380 INFO [JIRACOM:jira.web.filters.AccessLogFilter] -
http://localhost:8090/jira/default.jsp 116393-512 42
2005-07-15 18:27:01,380 INFO [JIRACOM:jira.web.filters.AccessLogFilter] -
http://localhost:8090/jira/default.jsp 116393-512 42
2005-07-15 18:27:01,472 INFO [JIRACOM:jira.web.filters.AccessLogFilter] -
http://localhost:8090/jira/secure/Dashboard.jspa 115881-809 87
2005-07-15 18:27:01,472 INFO [JIRACOM:jira.web.filters.AccessLogFilter] -
http://localhost:8090/jira/secure/Dashboard.jspa 115881-809 87
```

Here you can see user 'joe' enable access logging, then log out, and view the dashboard anonymously.

Usage

URL logging is disabled in JIRA by default. To turn it on, go to **Admin -> System -> Logging & Profiling**, and change the log level for AccessLogFilter from `WARN` to `INFO`. To make this change permanent, you would need to edit the corresponding section in `WEB-INF/classes/log4j.properties` on disk, changing:

```
log4j.category.com.atlassian.jira.web.filters.AccessLogFilter = WARN, console, filelog
log4j.additivity.com.atlassian.jira.web.filters = false
```

to:

```
log4j.category.com.atlassian.jira.web.filters.AccessLogFilter = INFO, console, filelog
log4j.additivity.com.atlassian.jira.web.filters = false
```

and then restart JIRA.

Tomcat access logging

Tomcat (used in JIRA Standalone) can also produce access logs. These are enabled by default in JIRA Standalone, and result in `logs/access_log.<date>.log` files being generated. If you are not using JIRA Standalone, Tomcat access logging may be enabled in `conf/server.xml`, by adding the following line before `</Context>`:

```
<Valve className="org.apache.catalina.valves.AccessLogValve" pattern="%h %l %u %t &quot;%r&quot; %s %b
%T %S" resolveHosts="false" />
```

You will need to restart JIRA for the changes to take effect.

The Apache Tomcat Access Log Valve documentation describes each of the above parameters.

This will generate logs that include the IP address, like:

```
127.0.0.1 - - [19/Oct/2006:12:38:09 +0800] "GET / HTTP/1.1" 302 - 0.240 -
127.0.0.1 - - [19/Oct/2006:12:38:09 +0800] "GET / HTTP/1.1" 302 - 0.243 -
127.0.0.1 - - [19/Oct/2006:12:38:11 +0800] "GET /secure/Setup!default.jspa HTTP/1.1" 302 - 1.732
39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:16 +0800] "GET
/secure/Setup!input.jspa?title=Your+Company+JIRA&mode=public HTTP/1.1" 200 24678 4.885
39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:24 +0800] "GET /styles/global.css HTTP/1.1" 200 548 0.213
39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:26 +0800] "GET
/secure/Setup!input.jspa?title=Your+Company+JIRA&mode=public HTTP/1.1" 200 24678 0.443
39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:26 +0800] "GET /styles/global.css HTTP/1.1" 200 548 0.001
39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:28 +0800] "GET /includes/js/combined-javascript.js HTTP/1.1" 200
65508 1.712 39F9C7F048F7F060A4B9883A7348082D
127.0.0.1 - - [19/Oct/2006:12:38:28 +0800] "GET /includes/js/combined-javascript.js HTTP/1.1" 200
65508 4.386 39F9C7F048F7F060A4B9883A7348082D
```

Related pages

No content found for label(s) logging.

Connecting to SSL services

This page describes how to get web applications like JIRA and Confluence connecting to external servers over SSL, via the various SSL-wrapped protocols. For instance, you may want to:

- reference a `https://...` URL in a Confluence macro
- use an IMAPS server to retrieve mail in JIRA.
- use SMTP over SSL (SMTPS) to send mail in JIRA
- Connect to a LDAP directory over SSL.
- Set up **Trusted Applications** over SSL.

If you want to run JIRA *itself* over SSL, see Running JIRA over SSL or HTTPS.

Problem symptoms

Simply entering the 'https' URL, or specifying IMAPS in JIRA will result in odd java.net.ssl.* exceptions in the logs, for example:

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
at com.sun.mail.imap.IMAPStore.protocolConnect(IMAPStore.java:441)
at javax.mail.Service.connect(Service.java:233)
at javax.mail.Service.connect(Service.java:134)
....
```

The cause

The problem is that our webapp is now acting as a SSL client, and as a client, it needs to obtain and 'trust' the server's *public key*.

This is identical to what happens when you visit a https://... URL in a browser - the browser fetches the public key and (if not signed by a trusted agent) presents it to you for inspection. If you trust the key, the browser saves it, and uses it to encrypt all subsequent communication with the site. We need to emulate this process before our webapp can access https resources.

The fix

Obtain the server's public key.

To quote Microsoft; "consult your system administrator". The public/private key pair will live somewhere on the server. The public key should be located and copied to the server hosting JIRA/Confluence. For example:

```
scp root@mail.yourcompany.com:/etc/ssl/certs/imapd.pem .
```

If you have openssl installed locally, the key can be retrieved with a command like:

```
jturner@teacup:~$ openssl s_client -connect imap.atlassian.com:imaps
CONNECTED(00000003)
depth=0 /C=AU/ST=NSW/L=Sydney/O=Atlassian/CN=imap.atlassian.com/emailAddress=info@atlassian.com
.....
.....
Server certificate
-----BEGIN CERTIFICATE-----
MIICiTCCAfKgAwIBAgIBADANBgkqhkiG9w0BAQQFADB/MQswCQYDVQQGEwJBVTEM
MAoGA1UECBMDTlNXMQ8wDQYDVQQHEwZTeWRuZXkxEjAQBgNVBAoTCUF0bGFzc2lh
bjEaMBgGA1UEAxMRY3ZzLmF0bGFzc2lhbi5jb20xITAfBgkqhkiG9w0BCQEWEmlu
Zm9AYXRsYXNzaWFuLmNvbTAeFw0wNTA5MjMwNjUyNTNaFw0wNjA5MjMwNjUyNTNa
MH8xCzAJBgNVBAYTAkFVMQwwCgYDVQQIEwNOU1cxDzANBgNVBAcTBlN5ZG5leTES
MBAGA1UEChMJQXRsYXNzaWFuMRowGAYDVQQDExFjdnMuYXRsYXNzaWFuLmNvbTEh
MB8GCSqGSIb3DQEJARYSaW5mb0BhdGxhc3NpYW4uY29tMIGfMA0GCSqGSIb3DQEB
AQUAA4GNADCBiQKBgQDhwAgx/gDgKe9tBjUCj7JtVkwQSzj2Dq0PHiJu1AWUYWFW
ivbBWaWSYbt/w9vIRSL8OlGVOLnlFOH5o7QIpIBZvd3xBMv6DxMijM86/hu8QTPt
KcMuqBTGpu1T846SzNncj883wSE1hXxezCgEFCsqyC7dVX4l0Ay6zgzkt2wc3QID
AQABoxUwEzARBglghkgBhvhCAQEEBAMCBkAwDQYJKoZIhvcNAQEEBQADgYEAJOgg
O4brCcQa3IgONo8UmLcHo6Rq+Py6ZA3ueUegy/uyQ358JUeL4kktXuYL9gAPCuMc
hsC1iyaOrWY/S9S67w2ZWqc+uYX9ophFHkxK1r3YiaiMpEzMyB12VWSrOITcR0LV
7NTWfxfPLUpkDbj+Mw/66QJkI0lqBvcKn3KXI74=
-----END CERTIFICATE-----
```

Cut and paste the certificate (including BEGIN and END lines) into a local file (eg. imapd.pem).

Import the public key.

To do this, you need to use the `keytool` program that comes with Java. If you haven't already, add `$JAVA_HOME/bin` to your PATH, and then run the following:

```
jturner@teacup:~$ sudo keytool -import -alias mail.yourcompany.com -keystore
$JAVA_HOME/jre/lib/security/cacerts -file imapd.pem
Enter keystore password:  changeit
Owner: EMAILADDRESS=info@atlassian.com, CN=atlassian.com, O=Atlassian, L=Sydney, ST=NSW, C=AU
Issuer: EMAILADDRESS=info@atlassian.com, CN=atlassian.com, O=Atlassian, L=Sydney, ST=NSW, C=AU
Serial number: 0
Valid from: Fri Feb 11 14:09:05 EST 2005 until: Sat Feb 11 14:09:05 EST 2006
Certificate fingerprints:
MD5:  CB:AE:7D:5D:1A:08:06:77:93:3B:0F:53:BB:40:C0:D4
SHA1: 7C:02:44:0D:A9:8F:F9:FB:BB:7B:C6:F1:52:DE:CA:00:17:D9:3A:A0
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

This will import the public key (imapd.pem) into Java's default keystore, and marks it as trusted.

On Windows the command is similar, eg.:

```
C:\Program Files\Java\jre1.6.0_05>bin\keytool -import -file c:\certs\imapd.pem -alias
mail.yourcompany.com -keystore lib\security\cacerts
Enter keystore password:
Owner: CN=*.atlassian.com, OU=IT, O=ATLASSIAN SOFTWARE SYSTEMS PROPRIETARY LIMITED, L=Sydney, ST=NSW,
C=au
Issuer: CN=DigiCert Global CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Serial number: a2d7047dc5d47ba988c9685e1efb860
Valid from: Thu Jan 10 11:00:00 EST 2008 until: Fri Jan 14 10:59:59 EST 2011
Certificate fingerprints:
         MD5:  9D:B4:9F:3D:0A:DE:6A:BD:BC:3D:95:BE:60:BD:70:02
         SHA1: 67:C6:E9:C8:3F:F1:7A:3C:66:E2:CE:62:78:A1:66:84:35:5E:62:1E
         Signature algorithm name: SHA1withRSA
         Version: 3
.....

Trust this certificate? [no]:  yes
Certificate was added to keystore

C:\Program Files\Java\jre1.6.0_05>
```

Restart the app server

Restart, and if everything is correct, your webapp should now connect to the SSL resource without problems.

Note: alternative keystore locations

Java will normally use a system-wide keystore in `$JAVA_HOME/jre/lib/security/cacerts`, but it is possible to use a different keystore by specifying a parameter, **-Djavax.net.ssl.trustStore=/path/to/keystore**, where '/path/to/keystore' is the absolute file path of the alternative keystore.

Setting this **is not recommended**, however, because if Java is told to use a custom keystore (eg. containing a self-signed certificate), then Java will not have access to the root certificates of signing authorities found in `$JAVA_HOME/jre/lib/security/cacerts`, and accessing most CA-signed SSL sites will fail. It is better to add new certificates (eg. self-signed) to the system-wide keystore (as above).

There is also a per-user truststore (~/.keystore) but (at least on Linux), but its contents do not appear to be logically appended to those in the system-wide keystore; ie. it is entirely separate, and only used if one specifies `-Djavax.net.ssl.trustStore=/home/<user>/.keystore`. This has the same disadvantage described above with custom keystores, so the per-user truststore is best avoided.

Debugging

Problems are one of two forms:

- Java is not referring to the correct keystore.
- The keystore does not contain the certificate of the SSL service you're connecting to.

Using Java

The attached SSLPoke.class Java program (source) is useful for debugging. It simply connects to a SSL service, sends a byte of input, and watches the output. For instance, connecting to a local HTTPS server on port 443 (the HTTPS default) with a untrusted (self-signed) certificate:

```
jturner@psyche:~$ java SSLPoke localhost 443
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to
requested target
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:285)
        at sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:191)
        at sun.security.validator.Validator.validate(Validator.java:218)
        at com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:126)
        at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:209)
        at
com.sun.net.ssl.internal.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:249)
        at com.sun.net.ssl.internal.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:954)
        at com.sun.net.ssl.internal.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:123)
        at com.sun.net.ssl.internal.ssl.Handshaker.processLoop(Handshaker.java:511)
        at com.sun.net.ssl.internal.ssl.Handshaker.process_record(Handshaker.java:449)
        at com.sun.net.ssl.internal.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:817)
        at com.sun.net.ssl.internal.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1029)
        at com.sun.net.ssl.internal.ssl.SSLSocketImpl.writeRecord(SSLSocketImpl.java:621)
        at com.sun.net.ssl.internal.ssl.AppOutputStream.write(AppOutputStream.java:59)
        at com.sun.net.ssl.internal.ssl.AppOutputStream.write(AppOutputStream.java:73)
        at SSLPoke.main(SSLPoke.java:28)
Caused by: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target
        at sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBuilder.java:174)
        at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:238)
        at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:280)
        ... 15 more
```

and connecting to a CA-verified certificate:

```
jturner@psyche:~$ java SSLPoke mail.atlassian.com 443
Successfully connected
jturner@psyche:~$
```

Similarly you would test port 636 to test LDAPS connections.

Make sure that the version of Java you are using is the same as that used in your production Java application. On Unix systems, 'ps -ef | grep java' will show the full command for Java processes. Check for the presence of a `-Djavax.net.ssl.trustStore` parameter. If `-Djavax.net.ssl.trustStore` is present in the command, this may well be the cause of your problems (see discussion above). You can verify whether the `-Djavax.net.ssl.trustStore` parameter is causing problems by running the `SSLPoke` test utility above with it, eg:

```
java -Djavax.net.ssl.trustStore=/my/custom/keystore SSLPoke localhost 443
```

If this fails (confirming the problem), the solution is to remove the `-Djavax.net.ssl.trustStore` parameter, import your custom keystore certificates into the main keystore with `keytool -importkeystore -srckeystore /my/custom/keystore -destkeystore $JAVA_HOME/jre/lib/security/cacerts`, and restart the application.

If you are sure the certificate is trusted and found by Java, and you are having low-level SSL problems, you can get debug information in the stdout logs by setting the `-Djavax.net.debug=all` property.

Using openssl

The `openssl` commands are very useful for debugging SSL problems. For instance, to print the server's certificate:

```
jturner@psyche:~$ openssl s_client -connect localhost:443 2>/dev/null
CONNECTED(00000003)
---
Certificate chain
 0 s:/C=AU/ST=NSW/L=Sydney/O=Atlassian/OU=Support/CN=localhost/emailAddress=jeff@atlassian.com
   i:/C=AU/ST=NSW/L=Sydney/O=Atlassian/OU=Support/CN=localhost/emailAddress=jeff@atlassian.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICizCCAfQCCQCt7NSYJaxDETANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMC
QVUxDDAKBgNVBAgTA05TVzEPMA0GA1UEBxMGU3lkbmV5MRIwEAYDVQQKEwlBdGxh
c3NpYW4xEDAOBgNVBAsTB1N1cHBvcnQxEjAQBgNVBAMTCWxvY2FsaG9zdDEhMB8G
CSqGSIb3DQEJARYSamVmZkBhdGxhc3NpYW4uY29tMB4XDTA4MDEwMzAwMTEzMVoX
DTA4MDIwMjAwMTEzMVowgYkxCzAJBgNVBAYTAkFVMQwwCgYDVQQIEwNOU1cxDzAN
BgNVBAcTBlN5ZG5leTESMBAGA1UEChMJQXRsYXNzaWFuMRAwDgYDVQQLEwdTdXBw
b3J0MRIwEAYDVQQDEwlsb2NhbGhvc3QxITAfBgkqhkiG9w0BCQEWEmplZmZAYXRs
YXNzaWFuLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEApOEOBbKKj189
15w078TUsR68gOahr6KiXg3s2zOyVaFbAepTnvhgtzej/qOJ8ESU5quCNIeBALov
PRyxUyySBLOV14i3vrhqugQLKeYUiCIehri2iRaU+/6FsX8kIgfSBIMu75uV0bnM
9A6t1zZizSdcP+1k2DzaIc8sXzkpJ2sCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCA
jm1xW5DvCYvHCa76QIwmsEIxCvya2zEqiwhAYC9gLbuQ1kjFgpn17U9x2ltJ1jWE
/5BQOeRk+cBaVVZJpfrKAUZoRczsNIQcaDs3JEUrZ6qBUxXheT8MijXhhhnKN0Fl
CvOuhw5DsgtEa1MT3CWetf9XFD3K4NI5+X7ihehs9g==
-----END CERTIFICATE-----
subject=/C=AU/ST=NSW/L=Sydney/O=Atlassian/OU=Support/CN=localhost/emailAddress=jeff@atlassian.com
issuer=/C=AU/ST=NSW/L=Sydney/O=Atlassian/OU=Support/CN=localhost/emailAddress=jeff@atlassian.com
---
```

(add `-showcerts` to print all the certificates in the chain)

save it to a local file:

```
jturner@psyche:~$ cat > localhost.pem
-----BEGIN CERTIFICATE-----
MIICizCCAfQCCQCt7NSYJaxDETANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMC
QVUxDDAKBgNVBAgTA05TVzEPMA0GA1UEBxMGU3lkbmV5MRIwEAYDVQQKEwlBdGxh
c3NpYW4xEDAOBgNVBAsTB1N1cHBvcnQxEjAQBgNVBAMTCWxvY2FsaG9zdDEhMB8G
CSqGSIb3DQEJARYSamVmZkBhdGxhc3NpYW4uY29tMB4XDTA4MDEwMzAwMTEzMVoX
DTA4MDIwMjAwMTEzMVowgYkxCzAJBgNVBAYTAkFVMQwwCgYDVQQIEwNOU1cxDzAN
BgNVBAcTBlN5ZG5leTESMBAGA1UEChMJQXRsYXNzaWFuMRAwDgYDVQQLEwdTdXBw
b3J0MRIwEAYDVQQDEwlsb2NhbGhvc3QxITAfBgkqhkiG9w0BCQEWEmplZmZAYXRs
YXNzaWFuLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEApOEOBbKKj189
15w078TUsR68gOahr6KiXg3s2zOyVaFbAepTnvhgtzej/qOJ8ESU5quCNIeBALov
PRyxUyySBLOV14i3vrhqugQLKeYUiCIehri2iRaU+/6FsX8kIgfSBIMu75uV0bnM
9A6t1zZizSdcP+1k2DzaIc8sXzkpJ2sCAwEAATANBgkqhkiG9w0BAQUFAAOBgQCA
jm1xW5DvCYvHCa76QIwmsEIxCvya2zEqiwhAYC9gLbuQ1kjFgpn17U9x2ltJ1jWE
/5BQOeRk+cBaVVZJpfrKAUZoRczsNIQcaDs3JEUrZ6qBUxXheT8MijXhhhnKN0Fl
CvOuhw5DsgtEa1MT3CWetf9XFD3K4NI5+X7ihehs9g==
-----END CERTIFICATE-----
jturner@psyche:~$
```

We can now calculate the fingerprint of the certificate with `openssl x509`:

```
jturner@psyche:~$ openssl x509 -fingerprint -md5 -noout -in localhost.pem
MD5 Fingerprint=29:37:D6:C3:03:B6:0B:29:76:55:B5:35:E0:BE:21:5C
jturner@psyche:~$
```

and verify that this fingerprint matches what is in Java's keystore:

```
jturner@psyche:~$ keytool -keystore /usr/lib/jvm/java-6-sun/jre/lib/security/cacerts -list | grep -A2
localhost
Enter keystore password:  changeit
localhost, 27/06/2008, trustedCertEntry,
Certificate fingerprint (MD5): 29:37:D6:C3:03:B6:0B:29:76:55:B5:35:E0:BE:21:5C
verisignclass3ca, 28/10/2003, trustedCertEntry,
jturner@psyche:~$
```

Editing a custom field option

At the moment it is not possible to rename an option of a custom field, e.g. a Select List custom field. This is fairly easy to do using SQL. Please shutdown JIRA then execute:

```
update customfieldoption set customvalue = 'New Option' where CUSTOMFIELD = <cfid> and id = <id>;
```

To rename the option. Where <cfid> is teh id of the custom field and <id> is the id of the option you would like to rename.

To get a list of all custom fields do:

```
select * from customfield;
```

Then update all issues with this value:

```
update customfieldvalue set STRINGVALUE = 'New Option' where CUSTOMFIELD = <cfid> and STRINGVALUE =
'Old Value';
```

Replace <cfid> with the custom field's id and 'Old Value' with the text value of the option.

Then restart JIRA and **re-index** the data (Administration -> System -> Indexing).

For details on editing the custom field tables, see the custom field tables documentation.

Allow editing of Closed Issues

By default, it is not possible to edit an issue while in the "Closed" state. However, if you would like to allow editing of closed issues, then you can do this by editing the workflow:

Default Workflow

In the file **./atlassian-jira/WEB-INF/classes/jira-workflow.xml** change:

```
<step id="6" name="Closed">
  <meta name="jira.status.id">6</meta>
  <meta name="jira.issue.editable">false</meta>
  <actions>
    <common-action id="003"/>
  </actions>
</step>
```

to:

```
<step id="6" name="Closed">
  <meta name="jira.status.id">6</meta>
  <meta name="jira.issue.editable">true</meta>
  <actions>
    <common-action id="003"/>
  </actions>
</step>
```

Restart JIRA after you've made this change.

Customised Workflows

You will need to add/edit the property to the Workflow Step.

1. Navigate to the Workflow step
2. Click on `View steps properties`.
3. Add or edit the property - Key: `jira.issue.editable` Value: `true`

* Note that you can not edit **Active** Workflows. You must deactivate before making this change.
(Alternatively, you can make the same change to the xml as described for the Default Workflow. You will need to edit the data in the database directly)

Updating your JIRA License Details

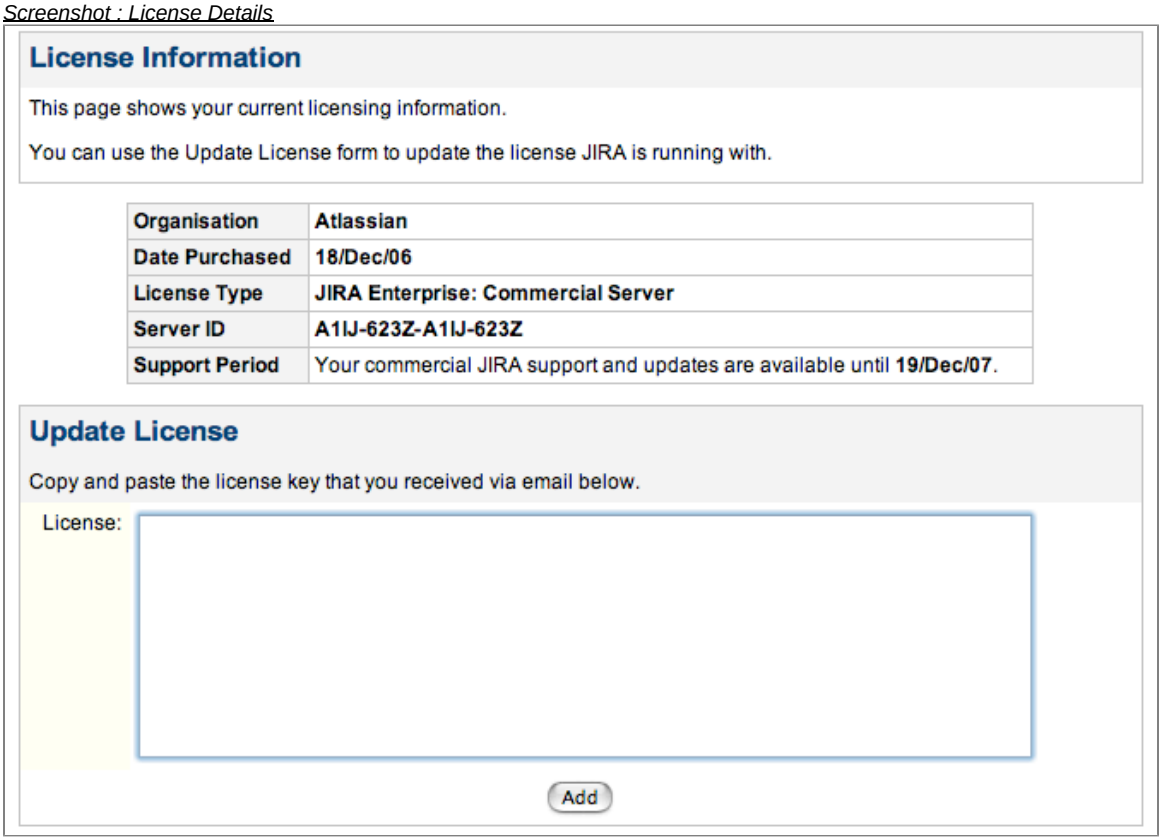
When you upgrade or renew your JIRA license, you will receive a new license key. You will need to update your JIRA server with the new license key.

> You can access your license key via http://my.atlassian.com

To update your JIRA license key,

1. Log into JIRA as a user with **admin** access.
2. Click the '**Administration**' link in the top navigation bar.
3. Click the '**License Details**' link in the left navigation column (under '**System**').
4. This will display your existing JIRA license details, and an empty box called '**Update License**'. Paste your new license into this box.
5. Click the '**Add**' button to update the JIRA installation with the new license.

Screenshot : License Details

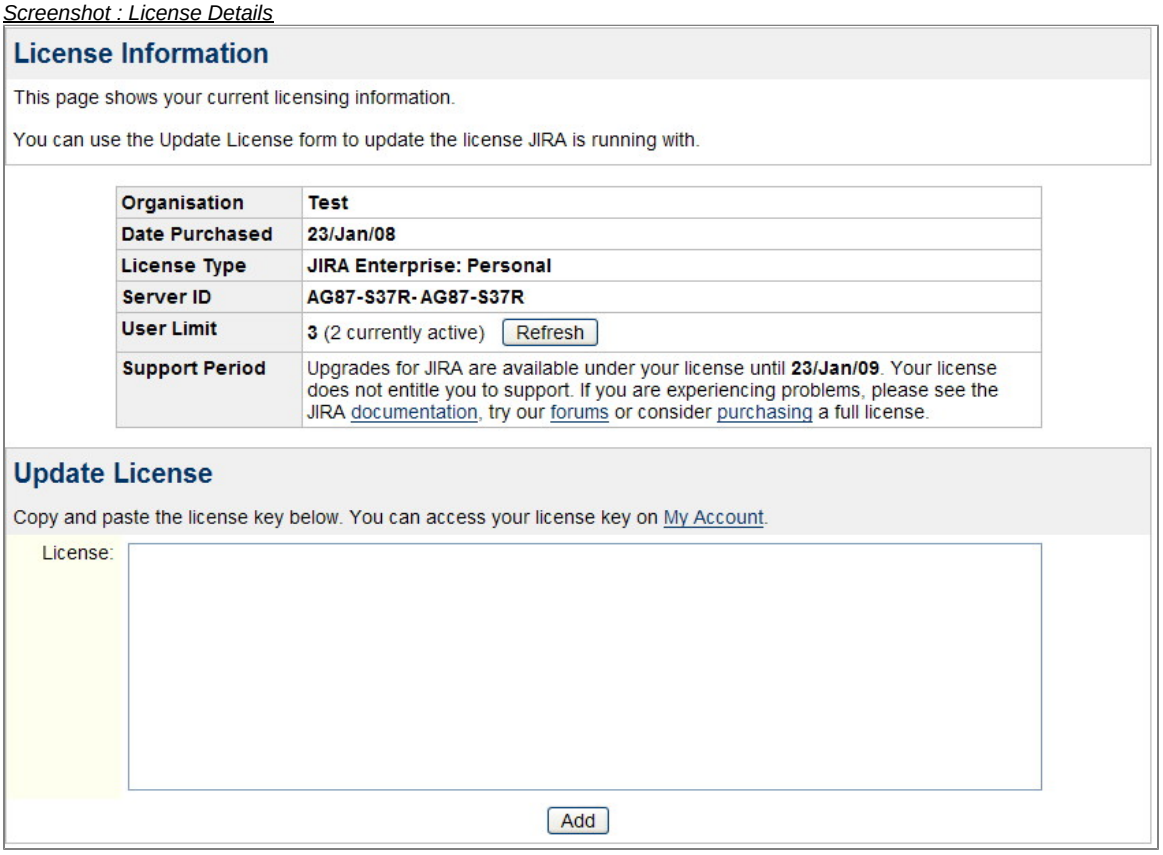


Have a user-based licence?

If you have a user-based (e.g. personal) license, then the user limit of your license will be displayed on the '**License Details**' page. This field is periodically refreshed, but you can retrieve the latest user limit by clicking the '**Refresh**' button.

> **Need more information about licensing?**
>
> - Want to find out more about **personal licenses**? Please see the Personal Licenses page.
> - **Other licensing questions**? Please see the Licensing FAQ.

Screenshot : License Details



Linking to local file under Firefox

Wiki markup allows you to links to files on the network / server with the format:

```
[file:///c:/temp/foo.txt]
```

This works fine under Internet Explorer, but Firefox and Mozilla block links to local files for security purposes. If you are happy with the risk of linking to local content, you can override the security policy and also enable linking in Firefox

The instructions for this can be found at http://kb.mozillazine.org/Links_to_local_pages_don't_work and you may also want to check out the other network preferences.

Please note that you need to use full URL syntax for your link (from http://kb.mozillazine.org)

> *You also need to use proper URI syntax for local file references. It is not proper to enter an operating-system-specific path, such as c:\subdir\file.ext without converting it to a URI, which in this case would be `file:///c:/subdir/file.ext`. In general, a file path is converted to a URI by adding the scheme identifier file:, then three forward slashes (representing an empty authority or host segment), then the path with all backslashes converted to forward slashes.*

Creating Issues via direct HTML links

If you would like your users to create issues from another site, you can by putting links to your JIRA's create issue page. You can also populate the fields on the page with values to select the project, the issue type or even the summary of the issue. This document will detail how to construct theses links and how to populate the fields. This feature is available from JIRA 3.5 onwards.

How to construct the link

The minimal HTML link to create issues has the following structure:

```
<a href="[JIRA BASE URL]/secure/CreateIssueDetails!init.jspa?[ARGUMENTS]">[DESCRIPTION]</a>
```

where

Component	Description	Example
[JIRA BASE URL]	The Base URL of the JIRA you wish to create issues in	http://jira.atlassian.com
[ARGUMENTS]	List of key value pairs separated by '&' which represent the field and its value to be set in the create issue screen	pid=10420&issuetype=4
[DESCRIPTION]	The link description visible to users	create issue in Test Project

JIRA Base URL

This Base URL is the same as the JIRA Base URL you wish to create issues in. This can be found under the admin section -> General Configuration -> Settings. For example, http://jira.atlassian.com is the base URL of the JIRA running at Atlassian.

The Arguments

The list of key value pairs included define which fields will have what values set. The argument list has the following properties:

- Each key value pair is separated by an '&'
  - **For Example:** [keyValuePair]&[keyValuePair]&[keyValuePair]...
- Each key value pair has the form 'key=value' where key is a field name and the value is the desired value to be set for its corresponding field
  - **For Example:** 'pid=10420&issuetype=1&summary=helloWorld&description=greetings'...
- The list must comply with HTML link syntax - that is all characters must be escaped.
  - Characters like space cannot be used directly, they must be encoded (escaped). Hence to use a space, we would replace the space with a '+' or '%20' which is the space equivalent. An excellent *HTML URL-encoding reference* listing all the characters and their corresponding encoded symbol can be found here
  - **For Example:** 'summary=This+is+a+summary%20with%20escaped+spaces'

As you can see, constructing the argument list is relatively simple. All we need is the name of the fields we want to set values for, and just structure it as above.

Fields thats not set will simply be assigned their normal default values. And the issue is not created until the user submits the form (this includes a validation check to confirm the field values are correct).

Finding out the field names and its possible values

The key in the key-value pair is the fields name, and to set a value for that field, we first need to know its name. The name of the field can be found by examining the source code of the page in which the field is in (To view the source code of a page, right click on the browser and select 'View source' or alike). Each field has a *name* attribute which represents the fields name. So all you need to do is find that attribute.

To find the possible values you can set is a bit more tricky. For any fields which accept plain text (such as summary, description and environment) there are no restrictions. However for other fields (such as Project, Issue Type, etc which take in Id) will require you to find the Id values. The range of Id values you can set can be found examining the same source code you found the field name from.

For example, the following is the HTML source code from the create issue page. From this we know that the Components field has the key *'components'* with values *'10013'*, *'10014'* and *'10015'* for each of the 3 components.

```
...
<select multiple name="components" id="components" size=" 3 ">
  <option value="-1">Unknown</option>
  <option value="10013" title="New Component 1  - ">New Component 1</option>
  <option value="10014" title="New Component 2  - ">New Component 2</option>
  <option value="10015" title="New Component 3  - ">New Component 3</option>
</select>
...
```

The following table shows a sample list of the standard JIRA fields with their name (key), the type of value expected and an example of the value

Display Name	Key	Value Type	Value Examples
Project	pid	Project Id	' 10420'
Issue Type	issuetype	Issue Type Id	standard JIRA issue type values range from '1' to '4'
Summary	summary	Plain Text	'issue+created%20via+link'
Priority	priority	Priority Id	standard JIRA priority values range from '1' to '5'
Due Date	duedate	Date	'15-Dec-2005' - may have different format depending on your JIRA date settings
Components	components	Component Id	'10014'
Affects Version/s	versions	Version Id	'10015'
Fix Version/s:	fixVersions	Version Id	'10015'
Assign To	assignee	Username	'admin' or 'sam@atlassian.com'
Reporter	reporter	Username	'admin' or 'sam@atlassian.com' To have the reporter field default to the currently logged in user, the user must be logged in and must not have the `Modify Reporter` permission.
Environment	environment	Plain Text	'this+is+the+environment'
Description	description	Plain Text	'this+is+the+description'

Custom Fields
Custom Fields key and value can be found by examining the source code also. There name/key are prefixed by 'customfield_' followed by their custom field id. For Example: 'customfield_10000'

Examples

Here are some simple examples to help you on your way. These examples provide links to create issue in JIRA Atlassian Test Project.

Source Code

```
To create an improvement issue in the Test project, click
<a href="http://jira.atlassian.com/secure/CreateIssueDetails!init.jspa?pid=10420&issuetype=4">here</a>
```

```
To Create a task with summary 'say hello world', click
<a href=
"http://jira.atlassian.com/secure/CreateIssueDetails!init.jspa?pid=10420&issuetype=3&summary=say+hello+worl
>here</a>
```

A more detailed example to
<a href="http://jira.atlassian.com/secure/CreateIssueDetails!init.jspa?pid=10420&issuetype=2&summary=detailed+example
&description=description+goes+here&components=10240&duedate=7%2dDec%2d2005
&customfield_10010=this+is+a+custom+field">
create an issue. Has description, components, duedate and a custom field preset.

How to create a download link to a file

Symptoms

You can perform the following steps to create a download link to a file in JIRA:

1. Use 'Text Field' custom field type to represent the link to a file (Administration -> Issue Fields -> Custom Fields)
2. Enable the 'Wiki Style Renderer' for this 'Text Field' custom field (Administration -> Issue Fields -> Field Configurations)
3. In the Issue Screen, try to add a link in the 'Text Field' custom field. For example:

```
[file:///V:/Jira test/example.xls]
```

The file can be opened by right clicking the link and select on the 'Save Target As...'. Do note that this only works on Internet Explorer but can also be enabled in FireFox. Please see:

- Linking to local file under Firefox

For more information on renderers, please refer to:

- Configuring Rich-Text Renderers
- http://jira.atlassian.com/secure/WikiRendererHelpAction.jspa?section=links

Current Reporter Browse Project Permission

Some JIRA installations have a use-case where they want a user to only see projects they **can** report issues in. Normally when you add the "Current Reporter" group to the "Browse Project" permission of one project, this project instantly becomes visible to **all** users (via the project table portlet), even if they are unable to report an issue in that project or not.

This guide is for those who want a user to only see issues they've reported **and** also not see any projects that's irrelevant to them (they are unable to create issues for). This permission is available as an optional permission type (since JIRA 3.2). You will need to uncomment the lines below in the file the `WEB-INF/classes/permission-types.xml`. Restart JIRA and this type will be available in your standard permissions page.

If you're running a WAR deployment, you'll need to rebuild the WAR after the change and redeploy. You may need to remove your old exploded WAR directory for the new one to take effect.

please note, this permission is only available in the Enterprise edition of JIRA.

```
<type id="reportercreate" enterprise="true">
    <class>com.atlassian.jira.security.type.CurrentReporterHasCreatePermission</class>
</type>
```

When using this special permission, users will only see projects where thy have create permission to and issues within that project where they are the reporter.

Why isn't this included in JIRA by default?
This permission is deliberately commented out of the `permission-types.xml` file. This is to ensure that only advanced JIRA administrators are able to access it. There are two reasons behind this:

- Firstly, the permission itself is used in fairly sophisticated scenarios.
- Secondly, the implementation of this permission is potentially dangerous. For example, it is possible to put your JIRA instance in an infinite loop by mapping this permission to the Create Issue function.

How to display custom field of the sub-task in the parent issue screen?

Symptoms

Cause

Resolution

By editing the value of `jira.table.cols.subtasks` property in the `jira-application.properties` could solve the problem. For example:

```
jira.table.cols.subtasks = priority, issuetype, status, resolution, assignee, customfield_10000
```

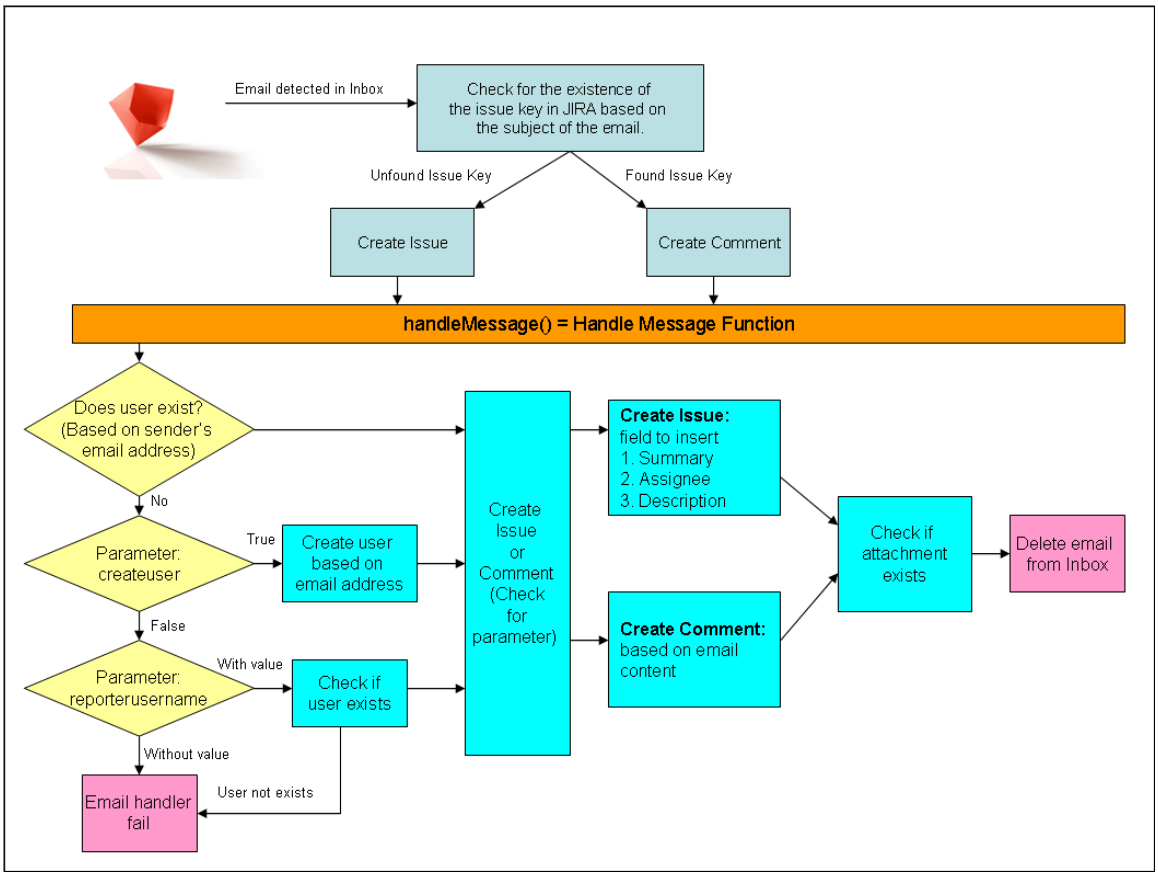
A restart of JIRA needs to be performed after making the modification.

Additional Information

Severity	Low
Regular Expression	$body
Article ID	JIRAKB163939652

Searching JIRA Knowledge Base

How the CreateOrCommentHandler works?



International characters in notification email subjects

Why are international characters in notification email subjects are being replaced with '?'

This happens if the system encoding is not the same as the JIRA encoding (by default UTF-8). System encoding can be seen in Administration -> System -> System Info, and JIRA encoding can be seen in Admin -> Global Settings -> General Configuration.

If there is a discrepancy, this can be fixed by setting the system encoding with a command-line option (**-Dfile.encoding=utf-8**) when starting JIRA. Eg. with Tomcat (JIRA Standalone), set the following before running the startup script:

```
JAVA_OPTS=-Dfile.encoding=utf-8
```

See JRA-5176 for more details.

Sending a JIRA data backup to support

To replicate reported problems, Atlassian support staff may ask you for a copy of your JIRA data. A backup can be generated as follows:

1. Login as a user with global administrator access.
2. Bring up the administration page by clicking either on the "Administration" link on the top bar or the title of the Administration box on the dashboard:

3. On the panel on the left, under the title "Import & Export", click "Backup Data to XML". This will bring up the "Backup JIRA data" page.

4. In the form, fill in the File path data entry box with a full path, including filename, that JIRA can write to.
5. Select the "Backup as Zip" checkbox if you don't want to anonymize the data (see below).
6. Click the Backup button, and be patient.
7. JIRA will return, confirming that it has written out the content to the file specified above.
8. Attach the generated file on disk to a support request. As the support site runs over SSL this is more secure than email, and you can remove attachments if you like.

Anonymizing data

A Confluence data anonymiser is also available.

As for JIRA 3.7+, data sent via Administration -> Support Request is anonymised by default, and it is thus the easiest route to sending us anonymised data (but be sure your mail server has a username/password specified, so relaying is allowed).

Support requests are resolved **much** faster if people attach their data export, however with some companies this is not an option, because the data contains sensitive commercial information.

In JIRA 3.7+, JIRA automatically anonymises data sent to Atlassian from the Administration -> Support Request page. For earlier versions, or people who want to anonymise JIRA data from the command-line, we've created a data 'anonymiser', which replaces most text in JIRA XML backups with **x**'s.

The anonymiser can be downloaded from here.

Unzip the package, then open a console and in the `jira_anon` directory run:
`java -jar joost.jar` ***<name of your backup file.xml>*** `anon.stx >` ***<name of the anonymised backup file to be generated.xml>***

For example, `java -jar joost.jar backup.xml anon.stx > anon-backup.xml`

Then zip the generated backup XML file, and attach it to a support case on https://support.atlassian.com

The anonymiser currently replaces the following text with x's:

- Issue summary, environment, and description
- Comments, work logs, change logs
- Project descriptions
- Descriptions for most elements (notification schemes, permission schemes, resolutions)
- Attachment file names.
- "Unlimited text" custom fields

Check anon-backup.xml to ensure it's clean enough for your needs before you send to us.

Problems?

Invalid XML Characters

If, when you run the anonymiser, you get an error indicating that there are invalid XML characters in the XML backup of your database, run our utility to remove invalid XML characters first before anonymising.

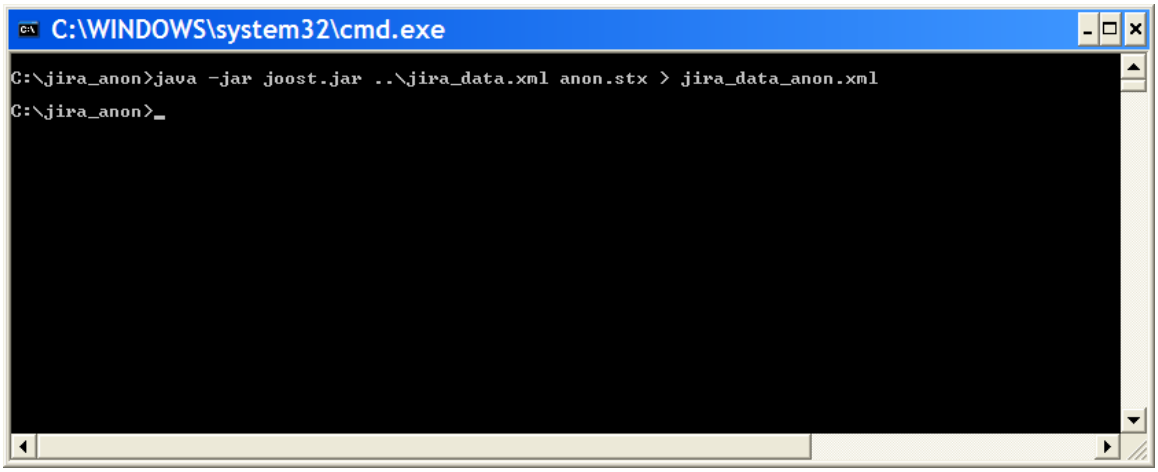
Java Version

You will need Java 1.4 or above to run this. You can check your Java version by running `java -version`, eg:

```
$ java -version
java version "1.5.0_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_07-b03)
Java HotSpot(TM) Client VM (build 1.5.0_07-b03, mixed mode, sharing)
```

If you find yourself using JDK 1.3 or earlier, check your path (`echo %PATH%` on Windows, `echo $PATH` on Unix) and ensure that the right version of Java is at the beginning. See the docs for more info on setting up Java.

The screenshot below is a simple example of how it is run in the command prompt of Windows XP:



Letting customers only create issues

This page describes a minor JIRA modification which redirects users to an arbitrary page after creating issues (and potentially other operations). It is mainly of interest to JIRA **Professional** and **Standard** users.

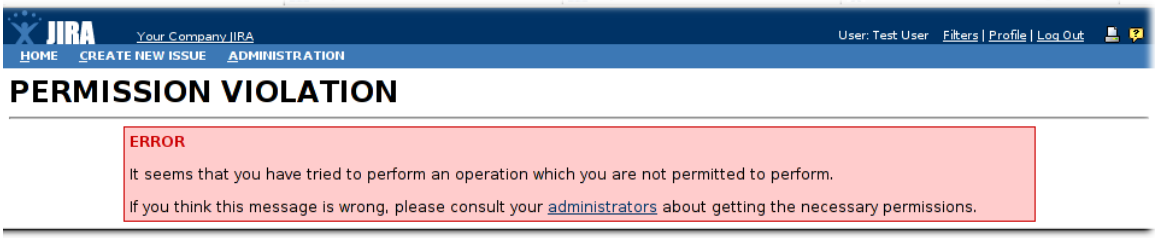
Scenario

When JIRA is used in a public environment, it is often useful for customers to be able to raise issues directly, but not see other customers' issues.

In JIRA Enterprise one can grant the **Reporter** (and your company groups) the **Browse Issue** permission. Customers can then view issues they have raised.

In JIRA Professional and Standard, **Reporter** isn't available, and permissions can only be granted/denied per group. We want the **Create Issue** permission granted to everyone, but **Browse Projects** denied:

Users will see a permission error after creating an issue - not very customer-friendly!



Redirecting to a custom page.

What we want is the ability to redirect the user to a nice "Thanks for raising an issue" page. We might want to direct to a different page depending on which groups the user is in. This can be done as follows:

Modify actions.xml

If you are using JIRA **Standalone distribution**, open *atlassian-jira/WEB-INF/classes/actions.xml*. If you are deploying JIRA as a webapp and have the **WAR/Webapp distribution**, first copy *webapp/WEB-INF/classes/actions.xml* to *edit-webapp/WEB-INF/classes* and edit *actions.xml*

there.

Locate the section:

```
<action name="issue.ViewIssue" alias="ViewIssue">
<view name="success">/secure/views/issue/viewissue.jsp</view>
<view name="rss">/secure/views/issue/viewissue-rss.jsp</view>
<view name="issuenotfound">/secure/views/issuenotfound.jsp</view>
<view name="permissionviolation">/secure/views/permissionviolation.jsp</view>

<command name="moveIssueLink" alias="MoveIssueLink">
<view name="error">/secure/views/issue/viewissue.jsp</view>
</command>
</action>
```

Modify the **permissionviolation** page to `/redirectusers.jsp`:

```
<view name="permissionviolation">/redirectusers.jsp</view>
```

Create a redirect JSP

Now create *atlassian-jira/redirectusers.jsp* (Standalone) or _edit-webapp/_redirectusers.jsp (WAR/Webapp), containing something like this:

```
<%@ page import="com.opensymphony.user.User"%>
<%
User user = com.opensymphony.user.UserManager.getInstance().getUser(request.getRemoteUser());
if (user.inGroup("customerA-users")) {
response.sendRedirect("http://localhost/thankyou.jsp?user="+user);
} else {
response.sendReirect("http://localhost/thankyou.jsp");
}
%>
```

Your logic (group(s) to check for and redirect URLs will be different. If you don't want to create a custom page, you can redirect to *request.getContextPath()+"/secure/Dashboard.jspa"*

Deploy

If you are running JIRA Standalone, simply restart JIRA. If you are using the WAR/Webapp edition, run *build.bat* or *build.sh* to regenerate the .war file, and redeploy this in your app server.

Asking for an attachment on the Create Issue page

To prompt for an attachment on the Create Issue page, go to the relevant Field Configuration page (eg. Administration -> Issue Fields -> Field Configurations -> Default Field Configuration), and unhide the 'Attachment' field.

Limiting the number of issues returned from a search view such as an RSS feed

JIRA allows you to view search results in several different formats, including Word, Excel, RSS or XML.

A search view that returns too many issues can take a long time for JIRA to complete and can use a large amount of memory. It can be a factor in OutOfMemoryErrors in JIRA.

An large RSS feed of search results can be particularly problematic, because:

- the user's RSS reader will continue to make the request periodically (for example, every hour)
- since the RSS reader makes the request, not the user directly, the user is unaware that the request takes a long time or is failing

You can use the following three properties in jira-application.properties to limit the number of issues returned by a search view.

See Making changes to jira.application.properties for the details of how to make and apply changes to **jira-application.properties**.

jira.search.views.default.max

`jira.search.views.default.max` sets a "soft" limit on the number of issues returned. It has a default value of 1000. You can set it to, for example, 100 as follows:

```
##
# The maximum number of results to issue navigator will request from a query
# - set this to zero or negative to be unrestricted (note that the server may forbid this, see below)
#
jira.search.views.default.max = 100
```

For an RSS or XML view, JIRA applies the limit by appending the `tempMax` parameter to the URL of the search view. For example:

- http://jira.atlassian.com/sr/jira.issueviews:searchrequest-xml/temp/SearchRequest.xml?&type=2&pid=10240&resolution=-1&sorter/field=is

In the above example, JIRA will limit the number of issues returned to 200 (in this example).

However users can override this "soft" default by removing the `tempMax` parameter from the URL or by increasing the value of `tempMax`.

jira.search.views.max.limit

`jira.search.views.default.max` sets a "hard" limit on the number of issues returned. It is disabled (commented out) by default. You can set it to, for example, 200 as follows:

```
##
# Whether search results are capped to a hard limit, otherwise return a 403 (Forbidden) to the client.
#
# Set this to negative (-1) or empty for unlimited results.
#
# NOTE: this SHOULD BE EQUALS OR GREATER THAN than jira.search.views.default.max above, otherwise all
issue navigator links will fail with a 403
#
jira.search.views.max.limit = 200
```

If a user makes a issue view request that would return more than 200 issues (in this example), JIRA does not return the issues but instead returns a 403 (Forbidden) error. While the user might not be happy, it prevents JIRA from consuming lots of resources and possibly running out of memory.

Make sure you set the value of `jira.search.views.max.limit` to greater than or equal to the "soft" limit set by `jira.search.views.default.max`. Otherwise all search views that would return issues limited by the default "soft" limit will instead return a 403 (Forbidden) error.

The `jira.search.views.max.limit` property was added to JIRA in version 3.10. It is not available in versions earlier than 3.10.

jira.search.views.max.unlimited.group

You may have a requirement for most users to have the limit imposed on them, but a few users to be exempt from the limit. One example of this is if your JIRA instance is Internet facing. You may want external (Internet) users to have the limit imposed on them, but for internal users to be able to produce unlimited search views. You can use the `jira.search.views.max.unlimited.group` parameter to achieve this.

The `jira.search.views.max.unlimited.group` property is disabled (commented out) by default. If you enable it, you must set it to a valid group or leave it empty. For example:

```
##
# Regardless of the above, users in this group will be able to request search requests that are
unlimited.
#
# This MUST be a valid group or empty.
#
jira.search.views.max.unlimited.group = jira-administrators
```

The `jira.search.views.max.unlimited.group` property was also added to JIRA in version 3.10. It is not available in versions earlier than 3.10.

Users exempted from the limit via this technique will still have to add the `tempMax` parameter to the URL for an RSS or XML view, as described above, in order to exceed the `jira.search.views.default.max` soft limit.

Anonymising data

A Confluence data anonymiser is also available.

As for JIRA 3.7+, data sent via Administration -> Support Request is anonymised by default, and it is thus the easiest route to sending us anonymised data (but be sure your mail server has a username/password specified, so relaying is allowed).

Support requests are resolved **much** faster if people attach their data export, however with some companies this is not an option, because the data contains sensitive commercial information.

In JIRA 3.7+, JIRA automatically anonymises data sent to Atlassian from the Administration -> Support Request page. For earlier versions, or people who want to anonymise JIRA data from the command-line, we've created a data 'anonymiser', which replaces most text in JIRA XML backups with **x**'s.

The anonymiser can be downloaded from here.

Unzip the package, then open a console and in the `jira_anon` directory run:
`java -jar joost.jar` ***<name of your backup file.xml>*** `anon.stx >` ***<name of the anonymised backup file to be generated.xml>***

For example, `java -jar joost.jar backup.xml anon.stx > anon-backup.xml`

Then zip the generated backup XML file, and attach it to a support case on https://support.atlassian.com

The anonymiser currently replaces the following text with x's:

- Issue summary, environment, and description
- Comments, work logs, change logs
- Project descriptions
- Descriptions for most elements (notification schemes, permission schemes, resolutions)
- Attachment file names.
- "Unlimited text" custom fields

Check anon-backup.xml to ensure it's clean enough for your needs before you send to us.

Problems?

Invalid XML Characters

If, when you run the anonymiser, you get an error indicating that there are invalid XML characters in the XML backup of your database, run our utility to remove invalid XML characters first before anonymising.

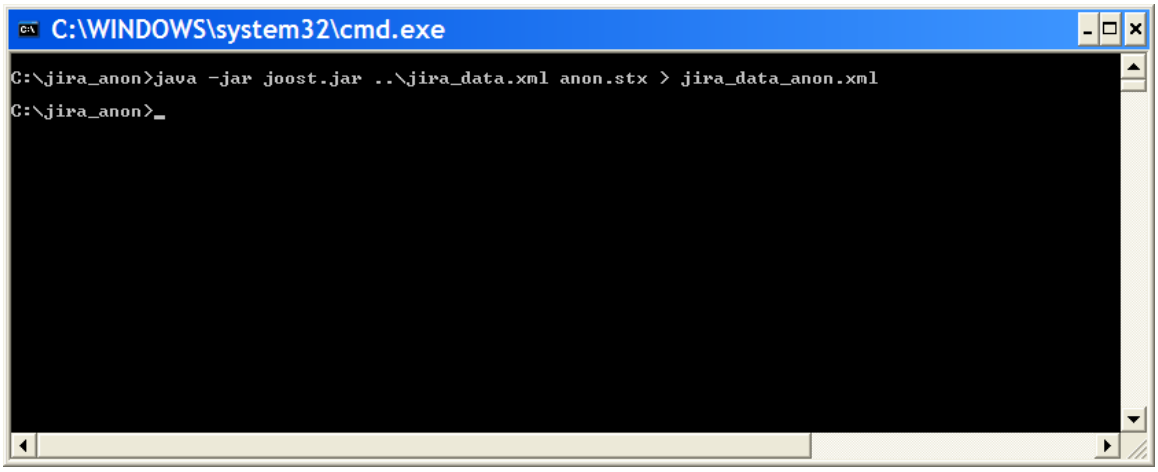
Java Version

You will need Java 1.4 or above to run this. You can check your Java version by running `java -version`, eg:

```
$ java -version
java version "1.5.0_07"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_07-b03)
Java HotSpot(TM) Client VM (build 1.5.0_07-b03, mixed mode, sharing)
```

If you find yourself using JDK 1.3 or earlier, check your path (`echo %PATH%` on Windows, `echo $PATH` on Unix) and ensure that the right version of Java is at the beginning. See the docs for more info on setting up Java.

The screenshot below is a simple example of how it is run in the command prompt of Windows XP:

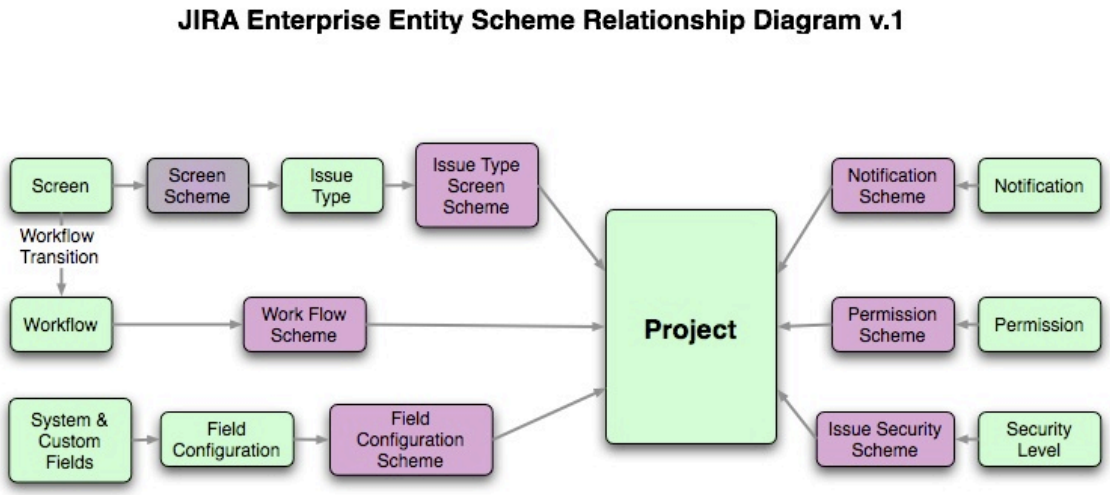


Single Sign-on

Single Sign-on Information

Scheme Entity Relations Map

This diagram illustrates the relationships between various JIRA entities (Enterprise Edition) and schemes.



Custom field column not visible in Issue Navigator

Symptoms

Issue Navigator is configured to display an extra column - Custom filed and the column appears every time you go to the "Configure your Issue Navigator" view, and allows you to move it to the different location, in the end it never shows when just looking at the filter in Issue Navigator.

Cause

The reason you are not seeing the custom field column in your Issue Navigator is that since it is constrained either by Issue Type or Context you will only ever see it if all the issues in the view are of the same issue type and that issue type is the one your custom field is configured against. The reason for this is that it can be misleading to show a column with no values for issues that can not even have that value assigned.

FYI. The improvement request regarding this behaviour is being tracked at JRA-9367 so please add your comments to the discussion, vote on it and add yourself as a watcher for future updates.

Resolution

A possible workaround this restriction/limitation is to configure the custom fields with the global context or for all issue types.

Additional Information

XML format for import & export files

Is the XML format for the import/export files (which also contains the configuration) documented?

Not as such - it is an XML version of the underlying entity model, pulled out of the database. As a result it is always changing with new fields and entities being added. The entity model itself is defined in **WEB-INF/classes/entitymodel.xml**

Configuring project specific security

We are often asked the following:

How do we configure the system so that a user/user group can only register/see issues on one specific project?

In order to configure the above please follow the below instructions and tweak as necessary for your organization:

1. Create a new Permission Scheme (Administration -> Schemes -> Permission Scheme) for Project External say **External_Permission_Scheme**
2. Create a new user group say **Group_External** (Administration -> Users & Groups -> Group Browser)
3. Add the External Users to that group
4. Associate **External_Permission_Scheme** to Project External (Administration -> Projects -> Project -> select Project External)

Note.

When users are created they are automatically a member of the jira-user group in order to allow them to login. The thing to note here is that the **Default Permission Scheme** grants users within the **jira-user group** certain permissions so those projects using the **Default Permission Scheme** will essentially give those users access to it.

To get around this either:

- Remove your external users from the **jira-user group** and give **Group_External** the ability to login by granting them the **global JIRA Users** permission (Administration -> Global Settings -> Global Permissions)
  OR
- Edit any Permission Schemes that grant the **jira-user group** specific permissions

It is also important to add that with the release of 3.7 and the introduction of Roles within JIRA it will not be necessary to create Groups for the above configuration.

For a detailed example using **Group Permissions** please see the following documentation: Using Project Level Security with User Groups

For a detailed example using **Project Role's** please see the following documentation: Using Project Level Security with Project Roles

Controlling project visibility

You can restrict project visibility to particular groups of users by using project permissions.

For example, if customers from Company X were put into the group "Cust-X" and given "Browse" permission for project Y, they will only be able to see Project Y (assuming you did not grant them the "Browse" permission for any other projects).

You should of course also give your developers permission to browse and operate on the project.

If you would like to restrict users to issues which they have created, set the "Browse Project" permission to be "Reporter." This way the user will have access to only the issues which they have created across all projects, but they will not be able to see any other issues.

JIRA Enterprise edition even finer-grained permissions. With this edition, you can set security on an issue-by-issue basis. For more information on JIRA's Issue Level Security, please consult the documentation.

Using Project Level Security with Project Roles

This tutorial (for JIRA Enterprise Customers) provides a step-by-step guide for creating project roles and using them in an issue security scheme. We recommend creating a test project and two test users for this tutorial.

Tutorial Goal

When completed, this tutorial will provide two issue security levels. One for issues that specific customers and your company can view, another for internal company eyes only.

1.Adding Project Roles

First we need to create project roles for our scheme to use:

Administration -> Users, Groups and Roles -> Project Role Browser -> Add Project Role

1. Create a project role called Customer A.
2. Create a project role called Customer B.
3. Create a project role called My Company.

In this example, the My Company project role will always have the same users/groups for each project. As a result, we'll set default members that will be used for all projects that use this scheme.

1. Next to the My Company project role, click Manage Default Members.
2. Add the users or groups for your company by clicking Edit next to the appropriate default (users or groups).

2. Adding an Issue Security Scheme

Next, we need to create the issue security scheme which will be hooked to our test project a bit later in this tutorial:

Administration -> Schemes -> Issue Security Schemes -> Add Issue Security Scheme

1. Add a new issue security scheme called Customers and My Company Issue Security Scheme.

3. Adding Issue Security Levels

We need to add security levels for this new issue security scheme. These levels will be available for selection to those that have permission to add issue security levels to issues. Users can only see levels of which they are members.

Administration -> Schemes -> Issue Security Schemes -> Click the Security Levels link next to Customers and My Company Issue Security Scheme

First we need to add a level for each customer:

1. Add a new issue security level called Customers and My Company.
2. Add the Customer A, Customer B and My Company project roles to this issue security level.
3. Click Default to make this the Default.

Next, we want a level for internal company eyes only:

1. Add another issue security level called My Company
2. Add the My Company project role to this issue security level.

4. Associating your Issue Security Scheme with a Project

Explain the step here and use the following syntax and color for menu notations:

Administration -> Project -> Projects -> Click on your test project name

1. For the Issue Security Scheme option, click Select.
2. Choose the Customers and My Company Issue Security Scheme from the list.
3. Click Next.
4. If you would like to associate existing issues with a security level, select it from the list, if not leave it at None.
5. Click Associate.

5. Adding project-specific members to a project role.

We need to specify the project-specific role members for the Customer A & B project roles.

Administration -> Project -> Projects -> Project Team -> Project Roles -> View Members

1. Add the first test user to the Customer A project role by clicking Edit in the Users column.
2. Add the second test user to the Customer B project role by clicking Edit in the Users column.

Project roles allow you to use the same permission scheme for multiple projects. We can change the members of project roles via the project!

Did it work?

1. Create one issue and set the Issue Security Level to My Company.
2. Create another issue and set the Issue Security Level to Customer A.
3. Create one more issue and set the Issue Security Level to Customer B.
4. Try logging in as each test user to ensure that they only see the appropriate issue.

Using Project Level Security with User Groups

This documentation is meant to give an in-depth analysis of Configuring project specific security — allowing full access to all projects for internal users, and limited access to external users by using JIRA groups and a project permission schemes. It is also possible to use Project Roles, but in this case we did not.

The example is based on the Atlassian Project Permission documentation documentation. While that documentation tells you everything you can do, we get a lot of questions about how exactly to set your system up to have two or more classes of users:

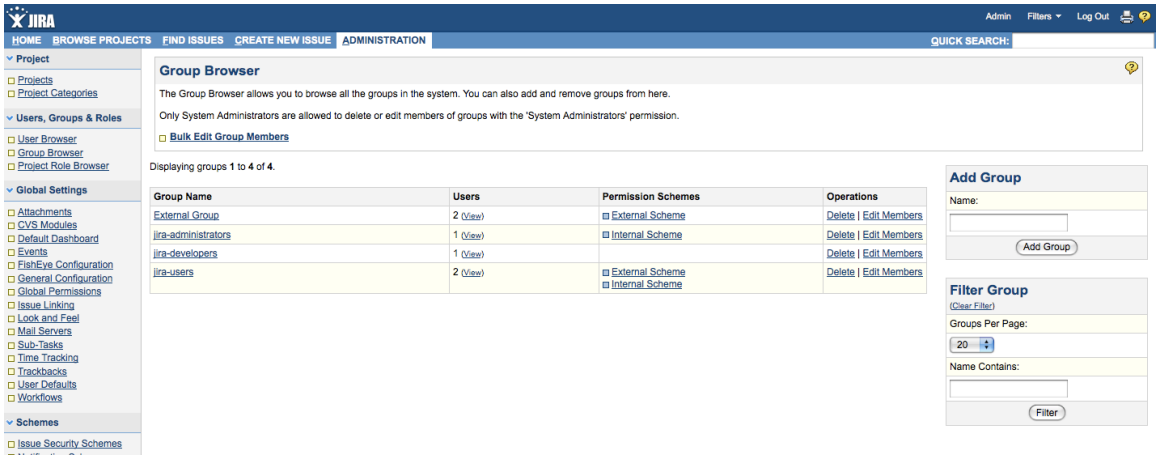
- Internal users (such as employees at your company) who have full permission
- External users (such as customer at your company) who have limited permission

Usually, though, in order to accomplish a security configuration which fits your company exactly, it will require a good amount of time, effort, and imagination on your part. At the moment JIRA is only able to support security at a project level or issue level. **Currently there is no field level security available.**

The first step for project level security is to define user groups. In this case a group called "external group" was created. All internal users will just be in the default "jira-users" group. In a default JIRA instance, when a user is created they will automatically be put into the jira-users group. Anyone who is external will have to be manually assigned to the external group and be removed from the jira-users group. There is no way to automatically assign users to certain groups without massive customizations to the JIRA environment. The reason for taking the approach of assigning all internal users to the jira-users group, is because this documentation is assuming that clients already have many internal users. Assigning a small group of users to one group as opposed to reassigning hundreds or thousands of users is easier.

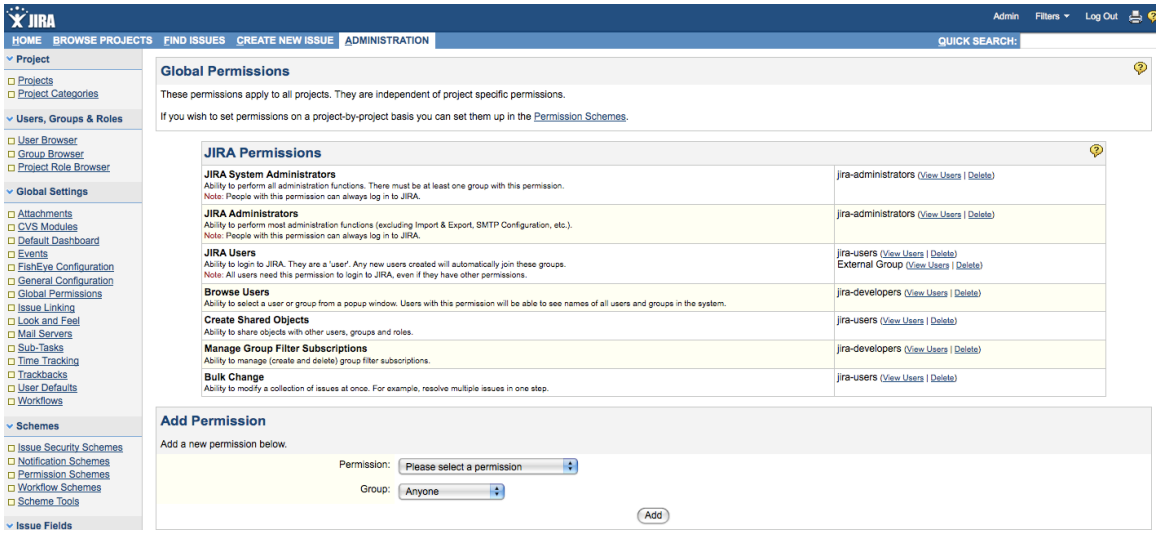
If starting from scratch, it is better to define and assign groups new groups from the beginning. For example, an "internal group" as well as an external group. But, in this example we will just look at jira-users and the external group. See the group settings in the image below for more detail:

To get to this screen: go to Administration > Users, Groups & Roles > Group Browser.



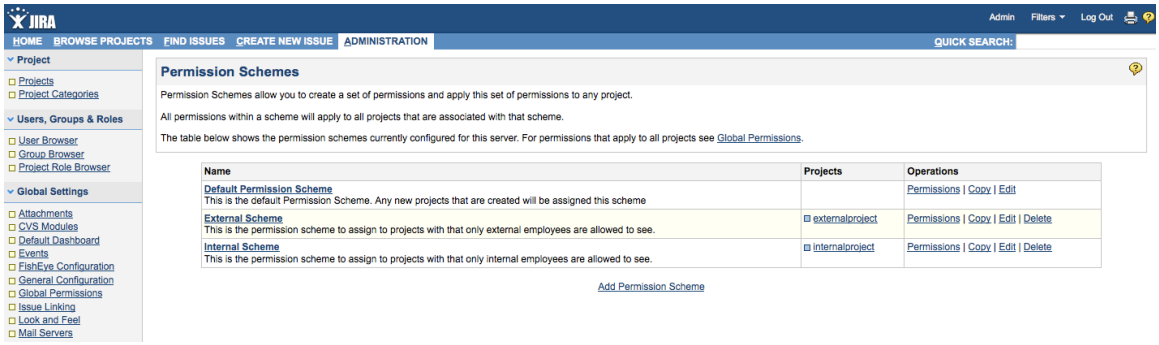
Now make sure that the External Group is added to the global JIRA Users permission so that they have access to JIRA. All users must be in the global JIRA Users group in order to access JIRA. **Note: The JIRA Users group is different from the jira-users group. JIRA Users is global while jira-users is group specific.** See the image below for more detail.

To get to this screen: go to Administration > Global Settings > Global Permissions.



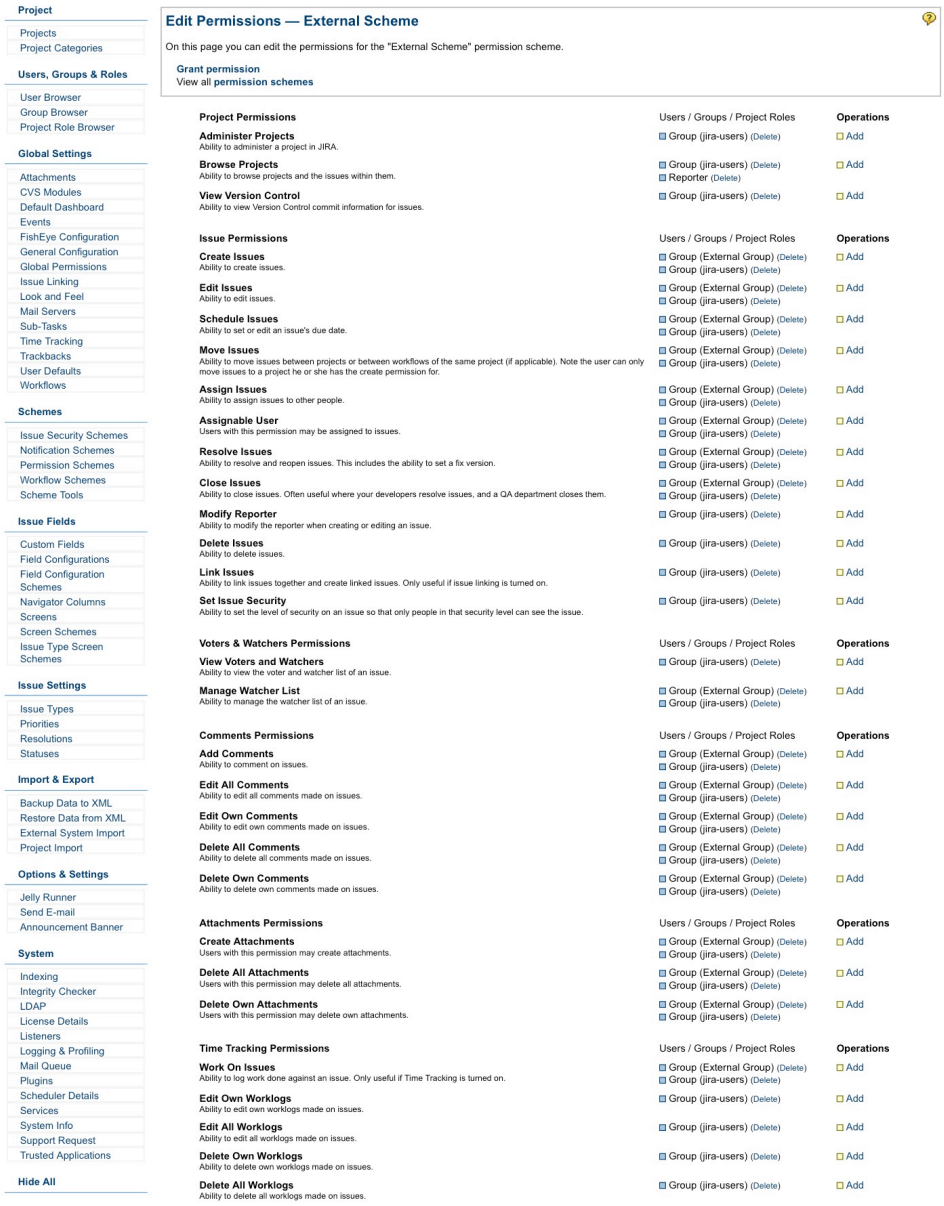
After creating the desired groups, separate permission schemes for each group need to be made. In the below image two schemes were created; an internal scheme and an external scheme. Obviously the internal scheme is for internal users and the external scheme is for external users. If your company has multiple users from multiple companies, you will need to make multiple schemes and groups for each project.

To get to this screen: go to Administration > Schemes > Permission Schemes.



After the schemes have been created, they must be tailored to meet your needs. For example: In the external scheme attach below, jira-users are given all permissions, while the External Group is given limited rights. Both groups must be present in this permission scheme to ensure that both internal users and external users have access to whichever project this scheme is assigned to. Only jira-users should be assigned to the Internal Scheme. See images below for more detail. **Please note that in the External Permission Scheme the "Browse Projects" category has both jira-users and reporters (rather than External Group).** This was done so external users can only see tickets they have created in the External Project and not others tickets. However, if "Reporter" is replaced with "External Group" then the External Group users will be able to view all tickets associated with the project..

To get to this screen: go to Administration > Schemes > Permission Schemes > Click on External Scheme.



Project
- Projects
- Project Categories

Users, Groups & Roles
- User Browser
- Group Browser
- Project Role Browser

Global Settings
- Attachments
- CVS Modules
- Default Dashboard
- Events
- FishEye Configuration
- General Configuration
- Global Permissions
- Issue Linking
- Look and Feel
- Mail Servers
- Sub-Tasks
- Time Tracking
- Trackbacks
- User Defaults
- Workflows

Schemes
- Issue Security Schemes
- Notification Schemes
- Permission Schemes
- Workflow Schemes
- Scheme Tools

Issue Fields
- Custom Fields
- Field Configurations
- Field Configuration Schemes
- Navigator Columns
- Screens
- Screen Schemes
- Issue Type Screen Schemes

Issue Settings
- Issue Types
- Priorities
- Resolutions
- Statuses

Import & Export
- Backup Data to XML
- Restore Data from XML
- External System Import
- Project Import

Options & Settings
- Jelly Runner
- Send E-mail
- Announcement Banner

System
- Indexing
- Integrity Checker
- LDAP
- License Details
- Listeners
- Logging & Profiling
- Mail Queue
- Plugins
- Scheduler Details
- Services
- System Info
- Support Request
- Trusted Applications

Hide All

Edit Permissions — External Scheme

On this page you can edit the permissions for the "External Scheme" permission scheme.

Grant permission
View all **permission schemes**

Project Permissions	Users / Groups / Project Roles	Operations
Administer Projects Ability to administer a project in JIRA.	Group (jira-users) (Delete)	Add
Browse Projects Ability to browse projects and the issues within them.	Group (jira-users) (Delete) Reporter (Delete)	Add
View Version Control Ability to view Version Control commit information for issues.	Group (jira-users) (Delete)	Add

Issue Permissions	Users / Groups / Project Roles	Operations
Create Issues Ability to create issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Edit Issues Ability to edit issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Schedule Issues Ability to set or edit an issue's due date.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Move Issues Ability to move issues between projects or between workflows of the same project (if applicable). Note the user can only move issues to a project he or she has the create permission for.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Assign Issues Ability to assign issues to other people.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Assignable User Users with this permission may be assigned to issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Resolve Issues Ability to resolve and reopen issues. This includes the ability to set a fix version.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Close Issues Ability to close issues. Often useful where your developers resolve issues, and a QA department closes them.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Modify Reporter Ability to modify the reporter when creating or editing an issue.	Group (jira-users) (Delete)	Add
Delete Issues Ability to delete issues.	Group (jira-users) (Delete)	Add
Link Issues Ability to link issues together and create linked issues. Only useful if issue linking is turned on.	Group (jira-users) (Delete)	Add
Set Issue Security Ability to set the level of security on an issue so that only people in that security level can see the issue.	Group (jira-users) (Delete)	Add

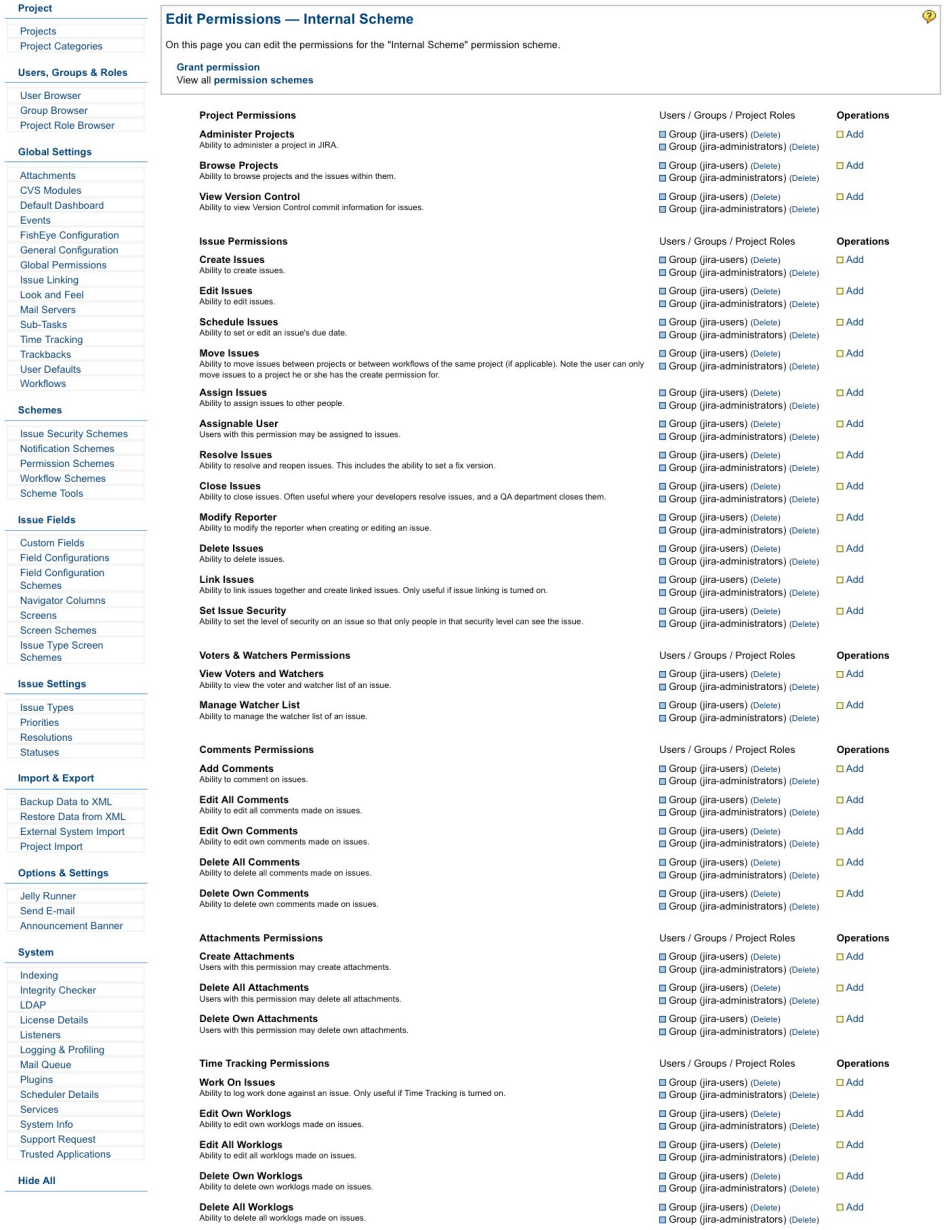
Voters & Watchers Permissions	Users / Groups / Project Roles	Operations
View Voters and Watchers Ability to view the voter and watcher list of an issue.	Group (jira-users) (Delete)	Add
Manage Watcher List Ability to manage the watcher list of an issue.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add

Comments Permissions	Users / Groups / Project Roles	Operations
Add Comments Ability to comment on issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Edit All Comments Ability to edit all comments made on issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Edit Own Comments Ability to edit own comments made on issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Delete All Comments Ability to delete all comments made on issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Delete Own Comments Ability to delete own comments made on issues.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add

Attachments Permissions	Users / Groups / Project Roles	Operations
Create Attachments Users with this permission may create attachments.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Delete All Attachments Users with this permission may delete all attachments.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Delete Own Attachments Users with this permission may delete own attachments.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add

Time Tracking Permissions	Users / Groups / Project Roles	Operations
Work On Issues Ability to log work done against an issue. Only useful if Time Tracking is turned on.	Group (External Group) (Delete) Group (jira-users) (Delete)	Add
Edit Own Worklogs Ability to edit own worklogs made on issues.	Group (jira-users) (Delete)	Add
Edit All Worklogs Ability to edit all worklogs made on issues.	Group (jira-users) (Delete)	Add
Delete Own Worklogs Ability to delete own worklogs made on issues.	Group (jira-users) (Delete)	Add
Delete All Worklogs Ability to delete all worklogs made on issues.	Group (jira-users) (Delete)	Add

To get to this screen: go to Administration > Schemes > Permission Schemes > Click on Internal Scheme.



Project
- Projects
- Project Categories

Users, Groups & Roles
- User Browser
- Group Browser
- Project Role Browser

Global Settings
- Attachments
- CVS Modules
- Default Dashboard
- Events
- FishEye Configuration
- General Configuration
- Global Permissions
- Issue Linking
- Look and Feel
- Mail Servers
- Sub-Tasks
- Time Tracking
- Trackbacks
- User Defaults
- Workflows

Schemes
- Issue Security Schemes
- Notification Schemes
- Permission Schemes
- Workflow Schemes
- Scheme Tools

Issue Fields
- Custom Fields
- Field Configurations
- Field Configuration Schemes
- Navigator Columns
- Screens
- Screen Schemes
- Issue Type Screen Schemes

Issue Settings
- Issue Types
- Priorities
- Resolutions
- Statuses

Import & Export
- Backup Data to XML
- Restore Data from XML
- External System Import
- Project Import

Options & Settings
- Jelly Runner
- Send E-mail
- Announcement Banner

System
- Indexing
- Integrity Checker
- LDAP
- License Details
- Listeners
- Logging & Profiling
- Mail Queue
- Plugins
- Scheduler Details
- Services
- System Info
- Support Request
- Trusted Applications

Hide All

Edit Permissions — Internal Scheme

On this page you can edit the permissions for the "Internal Scheme" permission scheme.

Grant permission
View all **permission schemes**

Project Permissions	Users / Groups / Project Roles	Operations
Administer Projects Ability to administer a project in JIRA.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Browse Projects Ability to browse projects and the issues within them.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
View Version Control Ability to view Version Control commit information for issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

Issue Permissions	Users / Groups / Project Roles	Operations
Create Issues Ability to create issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Edit Issues Ability to edit issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Schedule Issues Ability to set or edit an issue's due date.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Move Issues Ability to move issues between projects or between workflows of the same project (if applicable). Note the user can only move issues to a project he or she has the create permission for.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Assign Issues Ability to assign issues to other people.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Assignable User Users with this permission may be assigned to issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Resolve Issues Ability to resolve and reopen issues. This includes the ability to set a fix version.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Close Issues Ability to close issues. Often useful where your developers resolve issues, and a QA department closes them.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Modify Reporter Ability to modify the reporter when creating or editing an issue.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete Issues Ability to delete issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Link Issues Ability to link issues together and create linked issues. Only useful if issue linking is turned on.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Set Issue Security Ability to set the level of security on an issue so that only people in that security level can see the issue.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

Voters & Watchers Permissions	Users / Groups / Project Roles	Operations
View Voters and Watchers Ability to view the voter and watcher list of an issue.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Manage Watcher List Ability to manage the watcher list of an issue.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

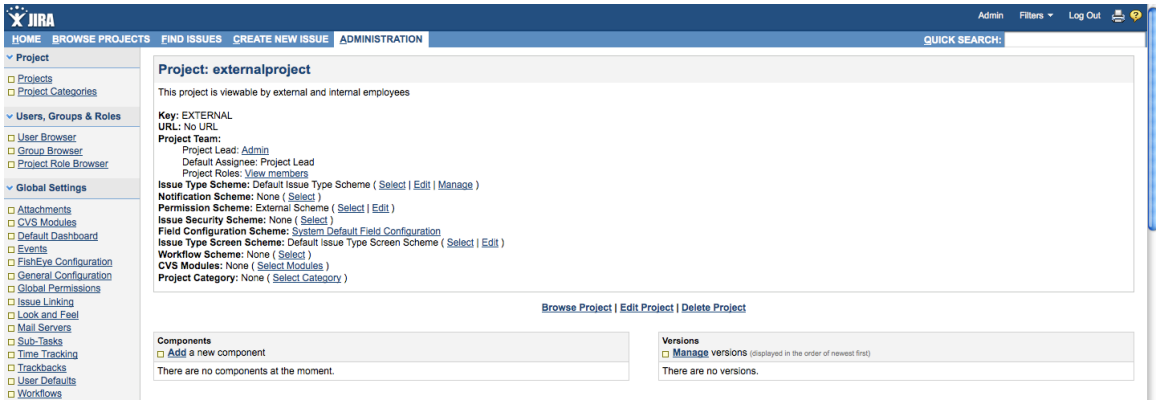
Comments Permissions	Users / Groups / Project Roles	Operations
Add Comments Ability to comment on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Edit All Comments Ability to edit all comments made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Edit Own Comments Ability to edit own comments made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete All Comments Ability to delete all comments made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete Own Comments Ability to delete own comments made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

Attachments Permissions	Users / Groups / Project Roles	Operations
Create Attachments Users with this permission may create attachments.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete All Attachments Users with this permission may delete all attachments.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete Own Attachments Users with this permission may delete own attachments.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

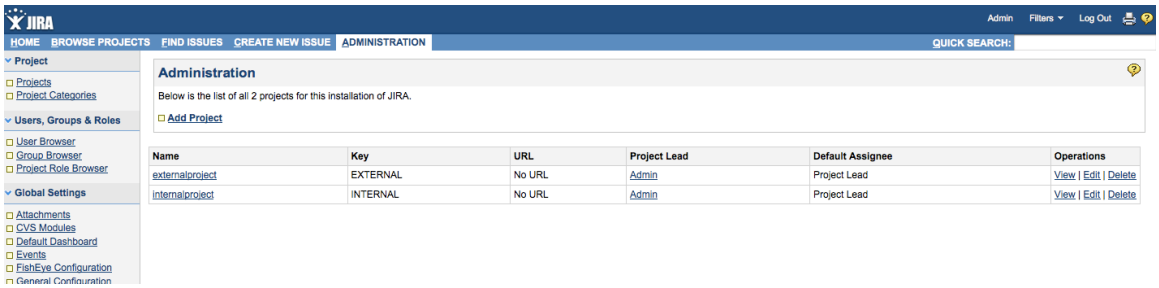
Time Tracking Permissions	Users / Groups / Project Roles	Operations
Work On Issues Ability to log work done against an issue. Only useful if Time Tracking is turned on.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Edit Own Worklogs Ability to edit own worklogs made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Edit All Worklogs Ability to edit all worklogs made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete Own Worklogs Ability to delete own worklogs made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add
Delete All Worklogs Ability to delete all worklogs made on issues.	Group (jira-users) (Delete) Group (jira-administrators) (Delete)	Add

Now assign the appropriate permission scheme to the appropriate project. For this example the internal scheme will be assigned to the the internal project and the external scheme will be assigned to the external project. See the images below for more detail:

To get to this screen: go to Administration > Project > Projects > Click on External Project.

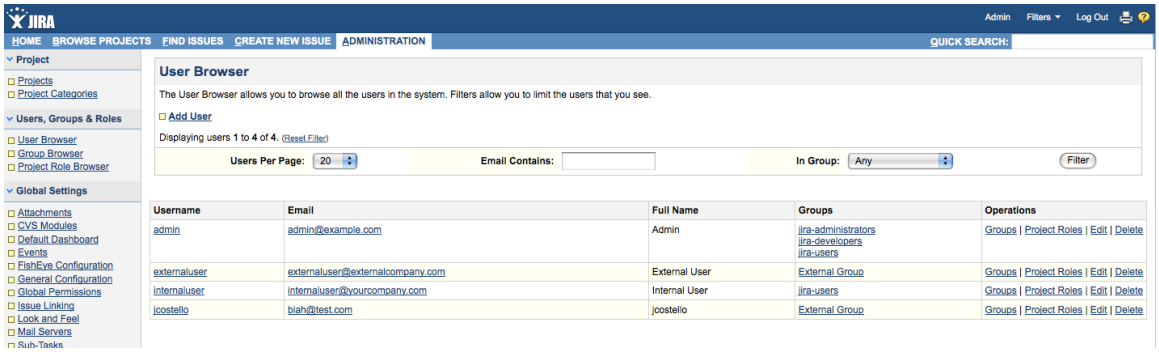


To get to this screen: go to Administration > Project > Projects > Click on Internal Project.



Once the above steps have been completed create users and add them to the appropriate group as seen in the image below. Note: When users are created will automatically be created belonging to jira-users and External Group. THe administrator will be responsible for manually removing the users from groups that the user should not belong to.

To get to this screen: go to Administration > Users, Groups & Roles > User Browser.



If done correctly the internal employees(jira-users) will have access to all projects, while the external users will only have access to their projects. Feel free to download the XML backup of this example on your local test instance.

The sample file

- Please ensure you have backed up your existing JIRA Enterprise instance
- You can download the JIRA helpdesk sample file here: ExampleSecurity.zip
- Restore the sample data file. You can learn how to restore a file here

User list and logins

- All user passwords are the same: **admin**
- The main username to login with is: **admin**
    - Full JIRA admin rights
    - Access to all projects

- Internal users are: **internaluser**
- These users are in the group: **jira-users**
- Access to both the Internal and External Projects and all issues.

- External users are: **externaluser** and **jcostello**
- These users are in the group: **External Group**

- Access to External Project and Issues Created only.

Changing Custom Field Types

You generally can't shift between custom field types since the data type they store may not match.

Having said that, certain fields can be safely *upgraded*, such as Version and Select lists to their multiple values counterpart. You can change the "customfieldtypekey" in the "customfield" table to whatever you need it to be. The table below lists the keys for commonly changed fields.

Custom Field Type	Type Key
Single Version	com.atlassian.jira.plugin.system.customfieldtypes:version
Multi Version	com.atlassian.jira.plugin.system.customfieldtypes:multiversion
Single Select	com.atlassian.jira.plugin.system.customfieldtypes:select
Multi Select	com.atlassian.jira.plugin.system.customfieldtypes:multiselect
Multi User	com.atlassian.jira.plugin.system.customfieldtypes:multiuserpicker

When moving back from a multi select list a select list, you **have** to make sure that only one item is selected for each multi select list.

When moving from multi-select to multi-user, you *have to ensure that each select-list value is a username (`userbase.username` value).

For select lists, you also need to update the "customfieldsearcherkey" field to use an appropriate searcher:

- For multi-selects, it is "com.atlassian.jira.plugin.system.customfieldtypes:multiselectsearcher"
- For select lists, use "com.atlassian.jira.plugin.system.customfieldtypes:selectsearcher"
- For multi-user pickers, use "com.atlassian.jira.plugin.system.customfieldtypes:userpickersearcher"

Examples

For example if you want to update all the version custom fields to become multiple version custom fields, you can use the SQL below.

```
UPDATE customfield
    SET customfieldtypekey = 'com.atlassian.jira.plugin.system.customfieldtypes:multiversion'
WHERE customfieldtypekey = 'com.atlassian.jira.plugin.system.customfieldtypes:version'
```

Or if you wanted to convert multi-select-list custom field to a multi-user custom field, first check that all custom field values map to users:

```
select * from customfieldvalue where id=
    (select id from customfield where cfname='multisel3') and
    stringvalue not in (select username from userbase);
Empty set (0.02 sec)
```

Then you can change the custom field type:

```
UPDATE customfield
    SET CUSTOMFIELDTYPEKEY='com.atlassian.jira.plugin.system.customfieldtypes:multiuserpicker',
        CUSTOMFIELDSEARCHERKEY='com.atlassian.jira.plugin.system.customfieldtypes:userpickersearcher'
  where cfname='MyMultiSelect';
```

Or if you wanted to convert text-field custom field to a free-text-field(unlimited text) custom field, first assign the value from stringvalue field to textvalue:

```
UPDATE customfieldvalue SET textvalue=stringvalue WHERE ID=(SELECT ID FROM customfield WHERE
customfieldtypekey='com.atlassian.jira.plugin.system.customfieldtypes:textfield' AND cfname='Text
Field');
```

Then, change the custom field type by updating the customfield table as below:

```
UPDATE customfield SET
CUSTOMFIELDTYPEKEY='com.atlassian.jira.plugin.system.customfieldtypes:textarea',
CUSTOMFIELDSEARCHERKEY='com.atlassian.jira.plugin.system.customfieldtypes:textsearcher'
where cfname='Text Field';
```

Restart JIRA. Then reindex (Administration -> Indexing) to update the search index.

Re-order workflow transactions

If you need to re-order the workflow actions of a workflow step, for example:

From:

- action1
- action 2
- action 3

To:

- action 2
- action 3
- action 1

...You cannot do this from the web interface, but you can do it.

- Download the workflow as an XML file
- Open up the XML file and change the order of the `<common-actions>` and `<actions>` for every `<step>`
- Upload the XML and view the changes

There are some details here in our documentation.

Creating an Unassigned Issue

You can choose to leave new issues unassigned. This can be achieved by altering the 'Allow Unassigned Issues' flag in the configuration options. To do this go to the General Configuration page of the **Administration** section. Now simply edit the configuration and turn the 'Allow Unassigned Issues' flag on. If a project already exists prior to making this change, please make sure to edit the project and change the "Default Assignee:" to be "Unassigned." Otherwise, the default assignee will continue to be whoever was originally assigned to the project. For more detail please refer to the documentation relating to this function.

This function is not enabled by default, as different companies tend to have different approaches to handling issues. We have found that many of our customers prefer to have issues always assigned to an owner, to ensure that somebody is responsible for its handling and resolution.

Workflow Properties

Not Everything is Recommended

You can use the workflow properties to implement restrictions on certain steps or transitions of a workflow. However, it is not really recommended as we do not guarantee some data and operations (e.g. bulk operations) will not be broken.

So, use it at your own risk.

Available JIRA Workflow Properties

There are a few workflow properties which you can use in a transition or step of a workflow. Here're some helpful links:

- JIRA Forum - Field Properties
- JIRA API Documentation - JiraWorkflow constant values

Name	Values	Related Issues	References	Notes

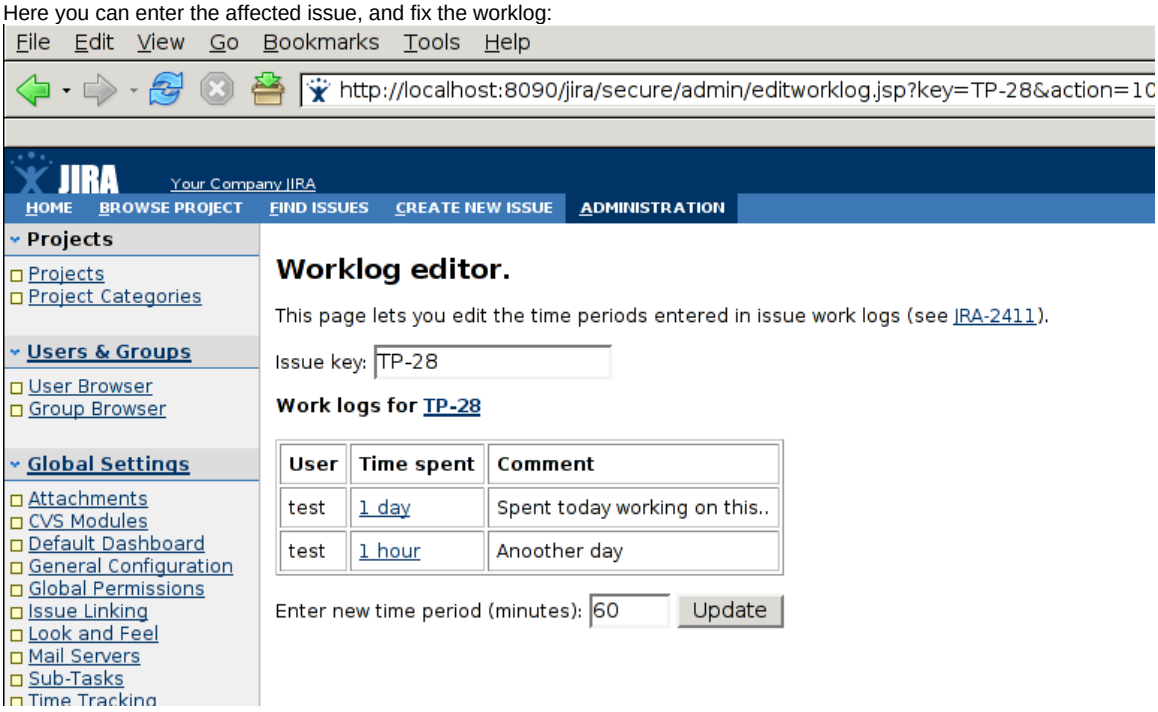
`jira.field.resolution.exclude`	Resolution `id`		• Resolutions per workflow step	Transition
`jira.i18n.submit`	i18n property key	• JRA-6798		Transition (usage: action submit button name)
`jira.i18n.title`	i18n property key	• JRA-6798		Transition (usage: action name, etc.)
`jira.issue.editable`	`true`, `false`		• JIRA Documentation - Workflow	Step
`jira.permission.*`	user1, user2 / group1, group2 / ...?	• JRA-6381	• WorkflowBasedPermissionManager • Link #2	Step

Editing the work log

Currently in JIRA, it is not possible to edit JIRA work log entries after they have been made. Until this oversight is fixed see JRA-2411, you can use the following workaround.

JIRA 3.3.1 and above

- Log into JIRA as an administrator
- Point your browser http://<your server>/secure/admin/editworklog.jsp, eg. http://localhost:8080/secure/admin/editworklog.jsp on JIRA Standalone
- Here you can enter the affected issue, and fix the worklog:



JIRA 3.3 and earlier:

- Download and save the editworklog.jsp for your version:
  - Jira 3.3
  - Jira 3.2.x
  - Pre 3.2

- If you are using JIRA Standalone:
    - Copy editworklog.jsp to `atlassian-jira/secure/admin/`, and then restart JIRA (run bin/shutdown.bat / shutdown.sh, and then bin/startup.bat / startup.sh).
- If you are using JIRA built as a webapp deployed in your own app server:
    - Locate the directory where JIRA was unpacked and built. In the `edit-webapp` directory, create the subtree secure/admin/, and copy editworklog.jsp there (yielding `edit-webapp/secure/admin/editworklog.jsp`).
    - Rebuild the JIRA webapp by running build.bat / build.sh in the root
    - Deploy the rebuilt webapp in your app server.
- In the restarted JIRA instance, log in as an adminstrator, and access http://localhost:8080/secure/admin/editworklog.jsp

Field Layout Schemes in JIRA 3.x

Unknown macro: {version-warn}

JIRA 3.1.1 or earlier

Field Layout Schemes in JIRA 3.x

This document describes creating field layout schemes per issue type per project in JIRA up to (but not including) version 3.2.

Field Layout Schemes

Through the use of Field Layout Schemes, it is possible to configure the visible and required fields per issue type per project.

For example, the issue type **Bug** within project **A** could be associated with one field layout while the issue type **Improvement** also in project **A** could be associated with another field layout. Furthermore, the project default field layout (i.e. all issue types without an associated layout within a project) could be associated with yet another field layout. In this way, each issue type can be associated with a configurable field layout for each project.

System Field Layout Scheme

The **System Field Layout Scheme** governs the field layout for all issue types in all projects not associated with a specific field layout.

This field layout can be edited by navigating to the following **Administration** section:

Administration -> Issue Fields -> Field Layout (System)

The field layout is displayed and can be edited as required.

Creating a Field Layout Scheme

In order to create a specific field layout association for an issue type within a project, it is necessary to create a field layout scheme:

Administration -> Issue Fields -> Field Layout Schemes -> Add Issue Field Scheme

Once created, it is then possible to configure the field layout as required.

Field Layout Association per Issue Type per Project

With a custom field layout, it is possible to associate the field layout with a particular issue type within a particular project.

By navigating to the **Project Administration** section:

Administration -> Project - > <Project Name>

it is then possible to manage the field layout associations for that project by selecting the **Manage** link within the **Field Layout Schemes** table or within the project summary table. From here, it is possible to create a default field layout association for all unassigned issue types within the project or to create a field layout association with a specific issue type for the selected project.

The issue type specific association will overwrite the project default association.

How to display a different format for the Number customfield

If you do not like how the Number custom field is displaying in Jira (for example, if you do not want it to display the commas), you can modify a velocity file to configure this. In particular, look at the file `WEB-INF/classes/templates/plugins/fields/view/view-number.vm`. As mentioned here:

> *At the moment to change the way the numbers are printed the easiest thing to do is to edit:*
> *WEB-INF/classes/templates/plugins/fields/view/view-number.vm*
> *under the JIRA web application and replace:*

> *$!numberTool.format($value)*
> *with:*
> *$value.longValue()*
>
> *and restart.*

How to remove 'NONE' from a select list custom field

There are some instances where the 'NONE' in every select list needs to be removed. By default in Jira you cannot do this from the web interface, however this is a workaround to this. In order to achieve this, you will have to edit a velocity template - /WEB-INF/classes/templates/plugins/fields/edit/edit-select.vm. In the source, delete the indicated lines:

```
...
<select name="$customField.id" id="$customField.id">
~~#if (!$fieldLayoutItem || $fieldLayoutItem.required == false~~
~~<option value="-1">$i18n.getText("common.words.none")</option>~~
~~#else~~
~~<option value="">$i18n.getText("common.words.none")</option>~~
~~#end~~
#foreach ($option in $configs.options)
<option value="$textutils.htmlEncode($option.value)"
#if ($value && $value == $option.value)selected#end
>$option.value</option>
#end
</select>
```

Make sure to back up the velocity file before changing it and to keep in mind the notes here.

Plans for JIRA's LDAP integration

A number of JIRA customers have been asking about the status of full LDAP support in JIRA. This page provides an overview of the current situation.

We intend to integrate JIRA with LDAP, however the task is quite an undertaking, and requires significant changes to JIRA internals in order to ensure the integration works correctly and smoothly.

Atlassian has taken some serious steps in making this feature available by building a framework called 'Atlassian User', which allows Atlassian products to work with LDAP. We have recently integrated Atlassian User with Confluence, a product with a slightly smaller code-base than JIRA, so that we can learn from the experience.

Integrating Confluence with Atlassian User (and hence LDAP) turned out to be a very complex process, which uncovered a number of issues. At present, Confluence-LDAP integration and configuration is the number 1 issue for Confluence Support. Currently, we are working on ironning out the problems that Confluence is facing with LDAP integration, and fixing Atlassian User bugs that we find. After that we will be looking at migrating JIRA to Atlassian User.

As you might be aware, Atlassian also has a product called Crowd, an identity management and SSO system, which helps to integrate JIRA (and other web applications) with LDAP. However, launching Crowd does **not** mean that we are not planning to deliver JIRA-LDAP integration.

See also:

JIRA-1962

What are the differences between JIRA Enterprise, Professional and Standard editions?

Please visit this page for a comparison: http://www.atlassian.com/software/jira/features/compare.jsp

Obsolete guides

Administration guides applying to older versions of JIRA

- Changing scope of existing custom fields
- Large text fields in Oracle
- Removing invalid characters from XML backups — In older versions of JIRA it was possible to cut & paste text containing control characters into JIRA issue fields. This causes problems, because JIRA's backup format is XML, and XML does not allow for the storage of most control characters .

Changing scope of existing custom fields

This is all obsolete as of JIRA 3.2, where one can change the scoping of custom fields from within the field interface

Overview

In JIRA, custom fields can apply to all issue, all issues of a certain type, or issues in a single project. Occasionally one wants to change the scope of a custom field, eg. to make a project-specific custom field global. JIRA does not yet have a means of doing this in the interface, but it is easy to do by manipulating the database directly. Here we show how to do it.

Load a SQL console.

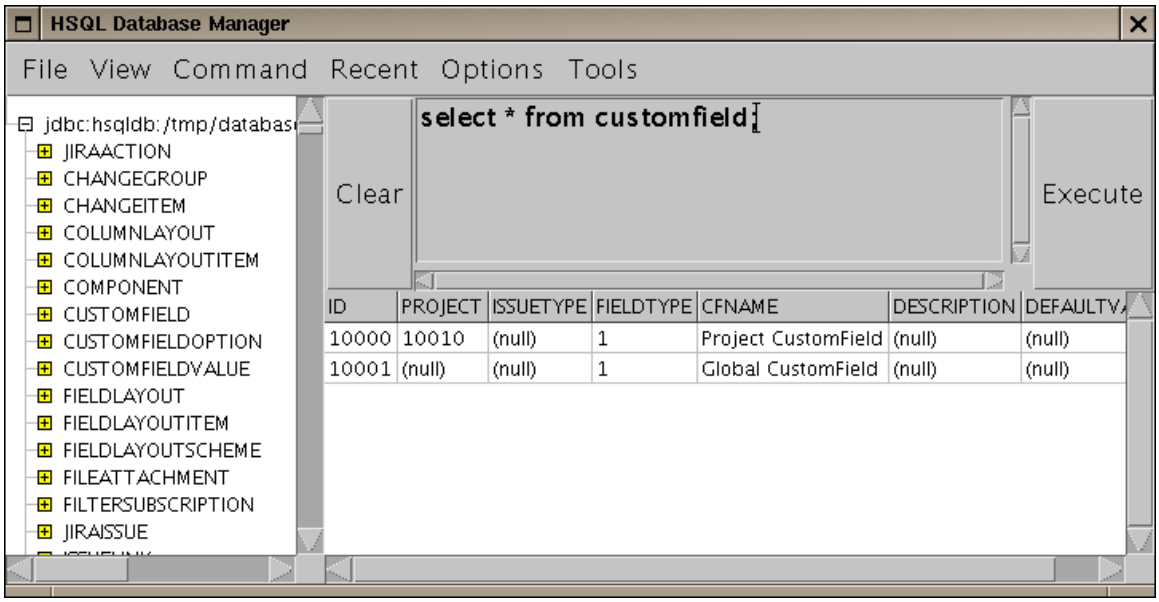
Load your JIRA database in a SQL console. This operation is database-dependent. Oracle users will run `sqlplus`, MySQL users `mysql` or mysqlcc. Users of JIRA Standalone should follow this guide.

Display the customfield table.

By running the query

```
select * from customfield;
```

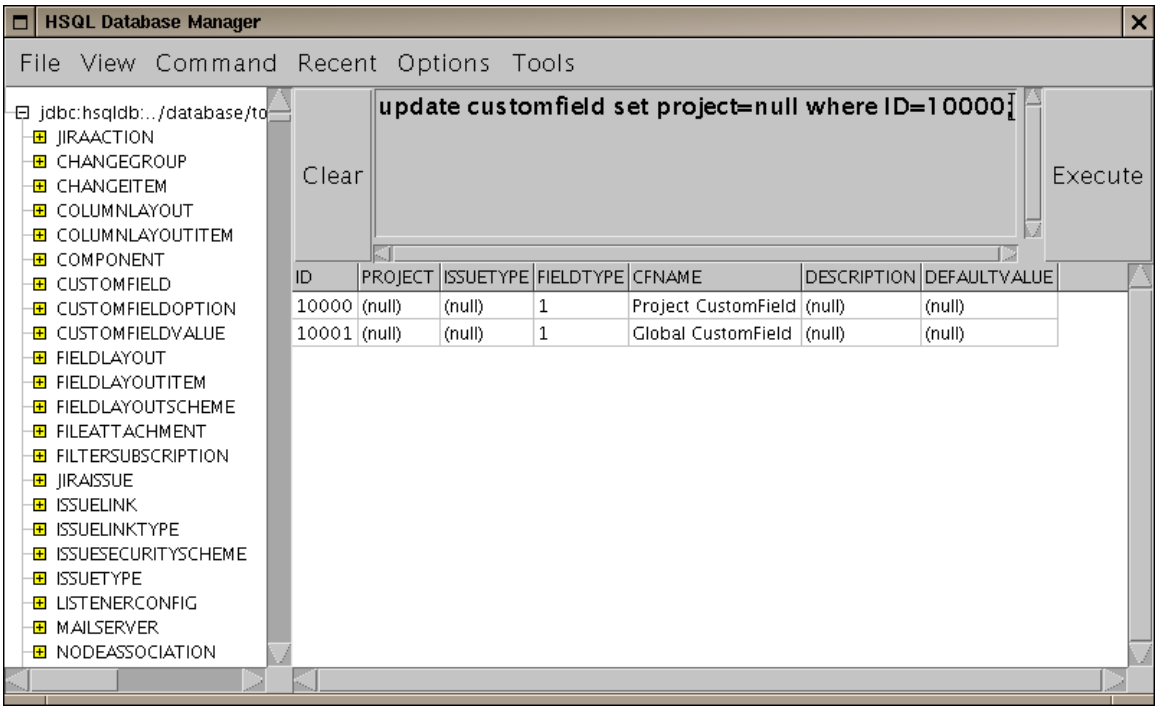
you will see how custom fields are defined in the database:



Here, we have one global and one project-specific custom field.

Edit the table flags.

Say we wanted to make the project-specific custom field global. We should set the custom field's `project` field to `null`:



Restart JIRA

Exit the SQL console and restart JIRA. The scope of the custom field should have changed.

Large text fields in Oracle

Applies to:	Installing JIRA 3.1 and earlier

JIRA 3 stores workflows in text format inside the database. In Oracle, VARCHAR2 text fields have a maximum size of 4000 characters. Most medium to large workflows therefore cannot be stored in Oracle. The same limitation prevents other text fields (issue summaries, descriptions and comments) from being over 4000 characters.

Fortunately Oracle have worked around the VARCHAR2 limitation in their latest Oracle 10g JDBC driver. This fix (described online here) works with Oracle 9 and 10, with some doubt regarding Oracle 8.

Does this apply to you?

For **new installations** of JIRA 3.2 and above, please follow the normal Oracle configuration notes. This page does not apply to you.

When **upgrading** from 3.1 or earlier to 3.2 and above, you will likely need to change your config files. Please note the example of upgrading JIRA with Oracle in the Upgrade docs. This page may be of use as a reference.

For **installing earlier versions** (3.1.1 and earlier), here is how to apply the fix:

Download the Oracle 10g driver

From Oracle's site download the ojdbc14.jar (or applicable) JDBC driver, and copy it to your app server's lib/ directory (eg. common/lib/ for Tomcat). Remove the old JDBC jar used previously.

Rebuild JIRA to use CLOBs

If running JIRA Standalone, edit atlassian-jira/WEB-INF/classes/entitydefs/fieldtype-oracle.xml. If running a JIRA webapp, copy webapp/WEB-INF/classes/entitydefs/fieldtype-oracle.xml to edit-webapp/WEB-INF/classes/entitydefs/fieldtype-oracle.xml and edit it there. Locate the lines:

```
<field-type-def type="very-long" sql-type="VARCHAR2(4000)" java-type="String"></field-type-def>
<field-type-def type="extremely-long" sql-type="VARCHAR2(4000)" java-type="String"></field-type-def>
```

and change them to:

```
<field-type-def type="very-long" sql-type="CLOB" java-type="String"></field-type-def>
<field-type-def type="extremely-long" sql-type="CLOB" java-type="String"></field-type-def>
```

JIRA 3.2 and above have a `fieldtype-oracle10g.xml` file containing these changes

(in JIRA 3.0.x and earlier the names may be different - make the change for all VARCHAR2(4000) entries)

If not running Standalone, rebuild the webapp by running build.sh / build.bat in the root directory. Do not yet redeploy the webapp.

Set the SetBigStringTryClob flag

In your application server, where the JDBC DataSource is configured, a parameter needs to be added to tell the Oracle JDBC driver to handle CLOBs in a standard way.

Tomcat 4 and 5.0:

Edit conf/server.xml (Tomcat 4) or conf/Catalina/localhost/jira.xml (Tomcat 5.0), locate the section where the 'jdbc/JiraDS' DataSource is set up, and add:

```
<parameter>
  <name>connectionProperties</name>
  <value>SetBigStringTryClob=true</value>
</parameter>
```

For instance, in JIRA Standalone one would then have:

```
    <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"/>
    <ResourceParams name="jdbc/JiraDS">
      <parameter>
        <name>driverClassName</name>
        <value>oracle.jdbc.driver.OracleDriver</value>
      </parameter>
      <parameter>
        <name>url</name>
        <value>jdbc:oracle:thin:@<database host machine>:<port>:<SID></value>
      </parameter>
      <parameter>
        <name>username</name>
        <value>...</value>
      </parameter>
      <parameter>
        <name>password</name>
        <value>...</value>
      </parameter>

<parameter>
  <name>connectionProperties</name>
  <value>SetBigStringTryClob=true</value>
</parameter>

      <parameter>
        <name>factory</name>
        <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
    </ResourceParams>
```

Tomcat 5.5

In Tomcat 5.5, the format for the added section would be:

```
<connectionProperties>SetBigStringTryClob=true</connectionProperties>
```

Orion / OC4J

For Orion/OC4J, edit config/data-sources.xml, and add the property as a nested tag:

```
<data-source
    class="<datasource driver class>"
    name="<name>"
    location="<location>"
    xa-location="<xa-location>"
    ejb-location="<ejb-location>"
    url="<url>"
    connection-driver="<driver>"
    username="<login>"
    password="<password>"
    inactivity-timeout="30"
>
  <property name="SetBigStringTryClob" value="true" />
</data-source>
```

Other app servers

Consult the relevant JIRA app server guide and the app server documentation to find how to add the property.

Use CLOBs in the database

You now have JIRA configured to use CLOBs, and the JDBC driver configured to handle them correctly. However the database in use still uses VARCHARs.

To fix this:

- generate an XML backup of the data (with the app server configured *without* the SetBigStringTryClob flag). Backing up the Oracle database using native tools is also a good idea.
- drop and recreate the Oracle database (or create a new parallel database to import into).
- restart JIRA (recreating the database with CLOBs), deploying the new (rebuilt) webapp.
- import the XML data.

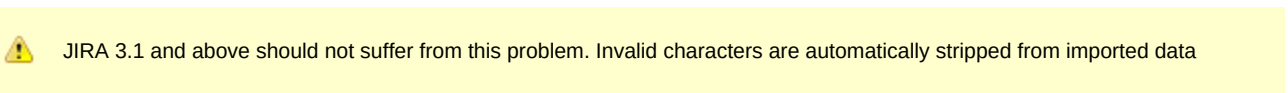
Alternatively (and assuming Oracle supports it), an SQL `ALTER TABLE` command can be issued to change the various column types (which columns can be seen from warning messages in the logs when JIRA is started after making the above changes).

Done!

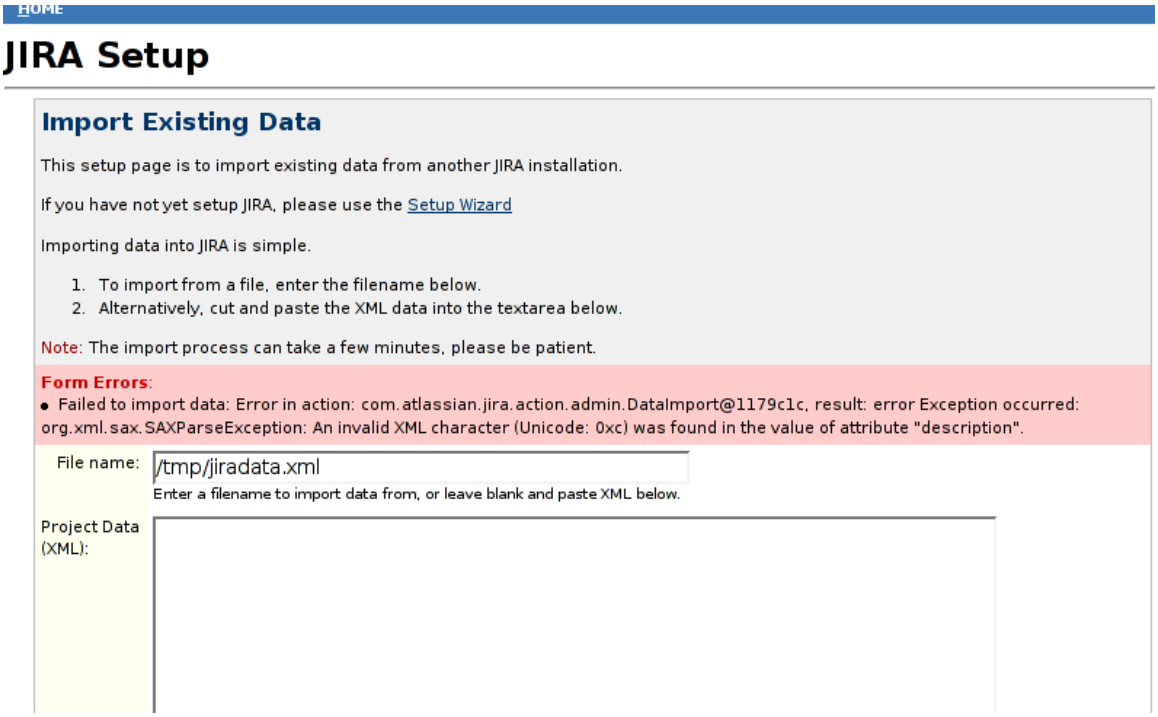
JIRA should now restart without warnings, and you should be able to add 4000+ character comments and descriptions.

If you have any feedback on this process, please let us know

Removing invalid characters from XML backups



In older versions of JIRA it was possible to cut & paste text containing control characters into JIRA issue fields. This causes problems, because JIRA's backup format is XML, and XML does not allow for the storage of most control characters . When XML containing control characters is imported into JIRA, the import fails with an error:



To fix this, one needs to remove the control characters from the JIRA backup file. This can be done as follows:

- Download atlassian-xml-cleaner-0.1.jar
- Open a DOS console or shell, and locate the XML or ZIP backup file on your computer, here assumed to be called jiradata.xml
- Run:

```
java -jar atlassian-xml-cleaner-0.1.jar jiradata.xml > jiradata-clean.xml
```

This will write a copy of jiradata.xml to jiradata-clean.xml, with invalid characters removed. You should now be able to import jiradata-clean.xml without problems.

Changing JIRA's log output

A common task when identifying JIRA problems is to turn up the log level, to get more debug-level logs. For short-term logging, this can be done in **Administration** -> **Logging and Profiling**, but this won't persist across JIRA restarts. To set the log level more permanently, do as follows:

In JIRA Standalone

1. Edit `atlassian-jira/WEB-INF/classes/log4j.properties`
2. **In 3.7 and above**: locate the section:

```
log4j.category.com.atlassian = WARN, console, filelog
log4j.additivity.com.atlassian = false
```

and change the `WARN` to `DEBUG`.
In 3.6.5 and earlier: locate the section:

```
#####################################################
# CLASS-SPECIFIC LOGGING LEVELS
#####################################################
# This stuff you may wish to debug, but it produces a high volume of logs.
# Uncomment only if you want to debug something particular
```

and below it, **add** the section:

```
log4j.category.com.atlassian = DEBUG, console, filelog
log4j.additivity.com.atlassian = false
```

3. Restart JIRA Standalone by running the shutdown and startup scripts.

JIRA deployed as a Webapp

In the Webapp/WAR distribution (used to build a webapp for deployment):

1. copy `webapp/WEB-INF/classes/log4j.properties` to `edit-webapp/WEB-INF/classes/log4j.properties`
2. edit `edit-webapp/WEB-INF/classes/log4j.properties`, making the same changes as above, so in the end you have:

```
log4j.category.com.atlassian = DEBUG, console, filelog
log4j.additivity.com.atlassian = false
```

3. Run build.sh or build.bat to rebuild the webapp
4. Redeploy the new webapp in your app server.

Background

JIRA's logging output is classified by importance, with the levels being:

- DEBUG: low-level details most people never need to know about.
- INFO: Informational messages on what JIRA is doing. Usually not interesting.
- WARN: Warnings that something may have gone wrong, or other messages a sysadmin may wish to know.
- ERROR: Something went wrong in JIRA. The person responsible for configuring JIRA should be notified

The default level is WARN, meaning warnings and errors are displayed. Sometimes it is useful to adjust this level to see more details.

Alternatively you can turn up logging for just one section. For instance, to see user authentication logs, locate the lines:

```
log4j.category.com.opensymphony = WARN, console
log4j.additivity.com.opensymphony = false
```

and change WARN to DEBUG.

Once the app server is restarted, you should see significantly more logs being generated.

Related pages

No content found for label(s) logging.

Logging email protocol details

JIRA's interaction with mail servers is a common cause of problems. It is often useful to know exactly what is passing over the wire between JIRA and SMTP, POP or IMAP servers. This page describes how to enable protocol-level logging.

JIRA 3.7.2+

Set **-Dmail.debug=true** and restart JIRA.

JIRA 3.6.5 -> 3.7.1

Due to a bug, just setting `-Dmail.debug=true` won't take effect. Instead the variable needs to be set in a properties file.

In JIRA Standalone..

1. Create `atlassian-jira/WEB-INF/classes/atlassian-mail.properties`, containing one line:
   `mail.debug=true`
2. Restart JIRA

In JIRA WAR/Webapp..

1. Create `edit-webapp/WEB-INF/classes/atlassian-mail.properties`, containing: `mail.debug=true`
2. Rebuild the .war (run `build.sh` or `build.bat`)
3. Redeploy the .war in your app server.

JIRA pre-3.6.5

Untested - probably the `atlassian-mail.properties` variant will work.

Output

In the logs, you should then see JavaMail initialize the first time a mail operation is run:

```
DEBUG: JavaMail version 1.3.2
DEBUG: java.io.FileNotFoundException: /usr/local/jdk1.6.0/jre/lib/javamail.providers (No such file or
directory)
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.providers
DEBUG: successfully loaded resource: /META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name:
{com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTranspor
Microsystems, Inc],
com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun
Microsystems, Inc],
com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun
Microsystems, Inc],
com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun
Microsystems, Inc],
com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun
Microsystems, Inc],
com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc]}
DEBUG: Providers Listed By Protocol:
{imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Sun Microsystems, Inc],
imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun Microsystems, Inc],
smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems, Inc],
pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun Microsystems, Inc],
pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun Microsystems, Inc],
smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun Microsystems, Inc]}
DEBUG: successfully loaded resource: /META-INF/javamail.default.address.map
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.address.map
DEBUG: java.io.FileNotFoundException: /usr/local/jdk1.6.0/jre/lib/javamail.address.map (No such file
or directory)
DEBUG: getProvider() returning javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc]
DEBUG POP3: connecting to host "localhost", port 110, isSSL false
S: +OK Dovecot ready.
C: USER pop-test
S: +OK
C: PASS pop-test
[Filter: profiling] Using parameter [jira_profile]
[Filter: profiling] defaulting to off [autostart=false]
[Filter: profiling] Turning filter off [jira_profile=off]
S: +OK Logged in.
C: STAT
S: +OK 2 1339
C: NOOP
S: +OK
C: TOP 1 0
S: +OK
Return-path: <pop-test@atlassian.com>
Envelope-to: pop-test@localhost
Delivery-date: Wed, 28 Feb 2007 16:28:26 +1100
Received: from pop-test by teacup.atlassian.com with local (Exim 4.63)
        (envelope-from <pop-test@atlassian.com>)
        id 1HMHMY-0007gB-8O
        for pop-test@localhost; Wed, 28 Feb 2007 16:28:26 +1100
Date: Wed, 28 Feb 2007 16:28:26 +1100
From: Jeff Turner <jeff@atlassian.com>
To: pop-test@localhost
Subject: Testing to me - Wed Feb 28 16:28:23 EST 2007
Message-ID: <20070228052826.GA29514@atlassian.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
User-Agent: Mutt/1.5.13 (2006-08-11)
Lines: 0
```

Related pages

No content found for label(s) logging.

Changing the username in JIRA

Currently JIRA does not have the ability to change usernames. Manually editing the database is not recommended due to the number of places the username is referenced (comments and filters for example).
The best option in this case is to edit the XML backup:

1. Create an XML Backup
2. Execute a global find and replace for the username
3. Restore JIRA with the new backup

> ⚠ If the username is a common word (e.g. admin), you may replace text that is not relevant to the user. So please be aware of this issue when performing the replace.

Changing the Due Date input format

The default JIRA input format is locale-specific (eg. 12/Jan/05). You may wish to use another format, such as the yyyy-mm-dd ISO format instead. To do this, set the following in **WEB-INF/classes/jira-application.properties**:

```
jira.date.picker.java.format = yyyy-MM-d
jira.date.picker.javascript.format = %Y-%m-%e
```

Extra CSV Import Features

Occasionally, there is a need to customise the CSV Importer to use more advanced operations. For example, it is possible to have Jira search for users using its Full Name instead of searching for a First Name Last Name combination To implement these features you will just need to add the property to the properties file created for the CSV Import.

Below is a list of the extra properties currently available.

Property Key	Description	Available Values
duplicate	Creates a duplicate field containing the same value	
settings.advanced.mapper.user	Sets the mapper used to search for current users	• com.atlassian.jira.imports.csv.mappers.FirstInitialFirstNameUserMapper - Expects users that are in the format **John Smith** and creates users with the name **jsmith** • com.atlassian.jira.imports.csv.mappers.FullNameUserMapper - Expects users that are in the format **John Smith** and creates users with name **johnsmith** • com.atlassian.jira.imports.csv.mappers.ConcatNameMapper - Expects usernames that are in the format **JSmith** and creates users with name **jsmith**
settings.advanced.mapper.comment	Set the mapper used to create comments	• com.atlassian.jira.imports.csv.mappers.SimpleCommentMapper - Default comment mapper • com.atlassian.jira.imports.csv.mappers.PvcsComment - Parses comments from a specific format to create comments with the right author and date. See JRA-6199

Attached is an example of how the properties are used.

Allowing users to create issues anonymously

JIRA can be configured to allow users to create issues without having logged into JIRA. There are two related actions:

1. Allowing users to browse and search issues in the project without logging in.
2. Allowing users to create issues in that project without logging in.

These can be achieved by adding the *Anyone* group to the **Browse Project** and **Create Issue** permissions in the permission scheme for the project. Additionally, *Reporter*, in the project's field configuration scheme, must be set as optional.

Any issue created by a user who is not logged in will display 'Anonymous' for the reporter of the issue.

Deleting a user account

Someone has left the company. How do I delete their user account if they have reported issues?

It's best not to - rather, disable their account by removing their membership from all groups. This will prevent that account from being used and being able to login.

Please note, if you disable a user account, the work log entries associated with the user will remain, the user will show up in license checks and email notifications will continue to be sent to the user (you can change the user's email address to an imaginary address to handle email notifications, e.g. user@example.com). There is currently no *"disabled user"* user type in JIRA, please see JRA-3774 and vote for the issue if you wish.

If you **really** want to delete the account (and historical revisionism doesn't bother you), you can bulk-edit the issues involved and change the reporter to someone else. You'll need the **Modify Reporter** permission to do this. You can then delete the user.

Changing the Issue Key format

By default, JIRA issue keys are of the format **<project key>-<issue id>**, eg. ABC-123. This general format is not changeable. However it is possible to alter the project key format in the **jira-application.properties** file, where it is defined as a regular expression:

```
jira.projectkey.pattern = ([A-Z][A-Z]+)
```

As you can see, the default is two or more A-Z characters. If, for instance, you wanted to allow numbers (eg. 'ABC2-123'), you would change the regexp to ([A-Z][A-Z0-9]+).

Also, although JIRA normally starts counting from 1 ('ABC-1', 'ABC-2' etc) you can adjust the starting count by editing the **project.pcounter** row in the database. JIRA caches this value in memory, so first shutdown your JIRA instance before updating the database. Then make the change in the database, then restart JIRA.

You should be able to find this file in your JIRA installation directory under:
\atlassian-jira\WEB-INF\classes

You may need to search hidden files and folders for it to appear.

QuickSearch guesses the issue key prefix (sometimes)

The Quick Search box (at the top-right of your JIRA screen) can sometimes find issues when you type just the number (e.g. '53'). Other times, you need to type the prefix too (e.g. 'JRA-53').

This is due to the concept of a 'selected project' - a bit of JIRA magic if you like. Basically, if you have recently done something in a project, that project becomes your 'selected project'. JIRA tries to 'guess' which issue you are looking for, given the 'selected project'.

But if you've just logged into JIRA, and not yet gone to an issue or a project, you will need to type the complete issue key (including the prefix).

Outward Link Description and Inward Link Description

When creating a new Issue Link Type, you need to specify an **Outward Link Description** (e.g. "duplicates") and an **Inward Link Description** (e.g. "is duplicated by").

What do these mean?

When a JIRA user links two issues together,

- the **Outward Link Description** applies to the issue from within which they clicked "Link this issue to another issue".
- the **Inward Link Description** applies to the issue that they choose to link to.

What does JIRA mean?

Like all good names in the software industry, it started as an in-house code name.

We originally used Bugzilla for bug tracking and the developers in the office started calling it by the Japanese name for Godzilla, **Gojira** (the original black-and-white Japanese Godzilla films are also office favourites). As we developed our own bug tracker, and then it became an issue tracker, the name stuck, but the **Go** got dropped - hence JIRA!

Further investigation into the name has revealed that **Gorira** is Japanese for "gorilla", whilst **Kujira** is Japanese for "whale". So **Gojira** is roughly translated to mean "gorilla the size of a whale"! (Thanks to yusuke_arclamp — Oct 2002)

For those who care - it sounds best if you yell it loudly, as though charging into battle. C'mon - try it!

Related

1. Filed your request in JIRA. What does that mean?
2. Talking that JIRA-slang language, or jiralang, if I may

Can I store customer details, like company, address and contact information, in JIRA?

JIRA itself stores only minimal user data (username, name, email, preferences). Since JIRA 3.7, you are able to store data in user 'properties'. You can store each customer detail as a separate user property, or create a wiki page for the customer and link to that instead. You could do this in Confluence by adding a Customer space and creating a page for every customer with their details. Then in JIRA, add a user property containing the link to that customer's page.

An alternative is to store user data in an LDAP server such as Active Directory or OpenLDAP. You can then authenticate users in JIRA against their LDAP password (see Configuring LDAP) and link to their full LDAP profile if available online.

There is also an open feature request for improved user properties at JRA-6354. You may wish to sign up for a user account and vote or comment to help influence our product roadmap.

Words ignored when searching

In JIRA certain *common words* are ignored from the search and search index. This can sometimes lead to unexpected results.

For example, suppose one issue contains the words "VSX will crash" and another issue contains the words "VSX will not crash".

A phrase search for "VSX will crash" will return both of the above issues. This is because the words `will` and `not` are part of the common ignored words list.

The full list of ignored English words is:

```
"a", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into", "is", "it", "no", "not", "of",
"on", "or", "s", "such", "t", "that", "the", "their", "then", "there", "these", "they", "this", "to",
"was", "will", "with"
```

You can alter the behavior of JIRA by modifying in relation to these English stop words by changing the Indexing Language from English to Other under Administration > General Configuration.

Using JIRA to Manage reusable modules

Many software products use external modules that are shared with other software products. The external projects are often managed separately, and have their own versions and lifecycles. So the question of how to "map" this scenario to JIRA often arises.

Currently, the best way to solve this in JIRA would be to create a separate JIRA project for each module and application. Then create issues in each JIRA project as needed and use issue linking to [link] related issues. Using Issue Links, issues can be easily linked across projects.

JIRA also has a clone issue function which can be used to copy an issue. The cloned issue can be then moved to another project. This should save the trouble of manually duplicating issues.

To get an idea of where each product and each module is "up to", JIRA's dashboard can be used. For example, one could place a portlet for each JIRA project that shows all open scheduled issues. This way the dashboard will provide an overview of all outstanding work for each project.

If all relevant issues for external modules have an issue in the product's JIRA project the standard reports and project summary panels (e.g. Change Log and Release Notes) should have all the information they need to be useful. Otherwise, JIRA can be extended by creating a custom project tab panel and/or a report that can look at more than one JIRA project and produce desired summaries. If you decide to write a custom report this tutorial should be useful.

In future we hope to better support this style of project organization, eg. through shareable subprojects (JRA-1072). Please vote/add your thoughts to the issue to increase its popularity. Also, please refer to this document which explains the way Atlassian shcedules new features.

Retrieving the JIRA Administrator

On this page:

- Scenario A: I don't know which user has the *JIRA Administrators* or *JIRA System Administrators* global permission
- Scenario B: I know which user has the *JIRA Administrators* or *JIRA System Administrators* global permission, but I have forgotten the password
  - 1. Send it via email
  - 2. Set the password directly in the database
- Alternative approach for JIRA versions earlier than JIRA 3.12: temporarily remove security restrictions
  - 1. Remove security restrictions
  - 2. Restore administrative privileges
  - 3. Re-apply security restrictions

Scenario A: I don't know which user has the *JIRA Administrators* or *JIRA System Administrators* global permission

You first need to find out which group(s) have been granted the global permission.

> The *JIRA System Administrators* global permission was added to JIRA in version 3.12. Anyone granted the *JIRA System Administrators* global permission can perform **all** administration tasks in JIRA, whereas anyone granted the *JIRA Administrators* global permission can perform most but not all administration tasks. Prior to version 3.12, anyone granted the *JIRA Administrators* global permission can perform **all** administration tasks.

To find out which group(s) have been granted the *JIRA Administrators* global permission, run the following database query:

```
select perm_parameter from schemepermissions where PERMISSION=0;
```

To find out which group(s) have been granted the *JIRA System Administrators* global permission, run the following database query:

```
select perm_parameter from schemepermissions where PERMISSION=44;
```

Now that you know which group(s) have the global permission, run the following database query to find out which users are in that group (replace "jira-administrators" with the group returned by the above query):

```
select user_name from membershipbase where group_name='jira-administrators';
```

If you don't know the password for the user(s) returned by this query, move on to Scenario B.

Scenario B: I know which user has the *JIRA Administrators* or *JIRA System Administrators* global permission, but I have forgotten the password

Here are two different ways you can solve this problem:

1. Send it via email

If you have configured JIRA to send email, just click on the **Forgot Password** link on the login page, enter your username and click the **Send it to me** button. Your password will be emailed to you.

2. Set the password directly in the database

You can also update the password hash stored for a user in your database. Run the following command to set the user called XXXX's password to the word *sphere*. Remember to restart Jira after making any changes directly to the database.

```
update userbase set
password_hash='uQieO/1CGMUIXXftw3ynrsaYLShI+GTcPS4LdUGWbIusFvHPfUzD7CZvms6yMMvA8I7FViHVEqr6Mj4pCLKAFQ==';
where username='XXXX';
```

Then restart your JIRA instance.

Alternative approach for JIRA versions earlier than JIRA 3.12: temporarily remove security restrictions

If you temporarily remove security restrictions from the JIRA instance, you will be able to carry out administrative actions without logging in. The login panel will be displayed if you go to the home page, but you can go to any page without logging in if you know the full URL for that page. Using this technique, you can correct the problem, then re-enable security restrictions.

This approach will only work for JIRA versions earlier than JIRA 3.12.

1. Remove security restrictions

1. Locate the `WEB-INF/classes/seraph-config.xml` file. (In JIRA 2.4.1 and earlier, it is called `security-config.xml`.)
2. Comment out everything between the <services> tags:

```
<security-config>
    ...
    <services>
       <!--
        <service class="com.atlassian.seraph.service.PathService">
            <init-param>
                <param-name>config.file</param-name>
                <param-value>/seraph-paths.xml</param-value>
            </init-param>
        </service>

        <service class="com.atlassian.seraph.service.WebworkService">
            <init-param>
                <param-name>action.extension</param-name>
                <param-value>jspa</param-value>
            </init-param>
        </service>
         -->
    </services>
    ...
</security-config>
```

3. Restart JIRA.

You will now be able to access all of JIRA without logging on. However you will have to know the full URL for the pages you want to access.

2. Restore administrative privileges

1. Find out which group(s) have been granted the *JIRA Administrators* global permission

1. In your web browser, go to `http://<your-server>/secure/admin/jira/GlobalPermissions.jspa`
2. Find out which group(s) have been granted the *JIRA Administrators* global permission (if no one has changed this one your instance, it will be the *jira-administrators* group), or grant this global permission to a group.

2. Grant back administrative privileges

You have a couple of options:

Option 1. Add admin privileges to an existing user:

1. Go to `http://<your-server>/secure/admin/user/UserBrowser.jspa`
2. Select a user.
3. Click on **Edit Groups** and add the user to the group that has the *JIRA Administrators* global permission.

Option 2. Creating a new admin user:

1. Go to `http://<your-server>/secure/admin/user/UserBrowser.jspa`
2. Click on **Add User**, fill in the form, then click **Create**.
3. Click on **Edit Groups**, and add the user to the group that has the *JIRA Administrators* global permission.

3. Re-apply security restrictions

After you have fixed up the permissions, remember to re-enable the commented-out section in `seraph-config.xml`, and restart JIRA.

Merging 2 JIRA instances

This page has been deprecated. Please refer to the documentation on restoring a project from backup, which supersedes this document.

There are a number of tools and procedures available, that make it possible to merge projects and issues of several JIRA instances into a single instance. This may be useful, for example, if you have several JIRA instances in your organisation, and would like to combine them into a single instance for easier management and administration.

Merging JIRA data from multiple JIRA instances is not a straightforward process at the moment, and unfortunately we do not have extensive tools that get the job done completely. There are however a number of tools contributed by the community, which will be described here. Please note that none of these tools are officially supported by Atlassian. Also, most of these tools involve some customisations and/or a number of manual steps. For more information see JRA-1604.

CSV Export and Import

This procedure involves creating a CSV export of your data in one JIRA instance, then importing this CVS data into another JIRA instance using the CSV importer. This will not include all data (e.g. Attachments, Change history, ...), but it provides the easiest way of migrating projects into a single JIRA instance.

As of JIRA 3.2, the Issue Navigator Excel View has been vastly improved and a lot more fields can be included in the export. In order to merge data from two JIRA instances, you will need to

1. Export the issues of interest, using the Issue Navigator Excel view. In order to do this, search for all the issues you would like to move (do one project at a time). Click on the 'All fields' Excel view, and save the resulting Excel file, as a CSV (comma-separated-values) file.
2. Copy the CSV file to a temporary directory on your new JIRA server. (Note that the directory needs to be readable by the account that is used to run your JIRA instance.)
3. Navigate to your new JIRA instance in a browser and import the CSV file using the CSV importer.

Data that won't be migrated includes:

- Attachments
- Change History
- Comments
- Issue Links
- Issue Watchers & Voters
- Permission Schemes for the Project
- Notification Schemes for the Project
- Groups

The CSV Export and Import method is the easiest approach to merging two JIRA instances. Novice users should follow this approach. Whilst some data can't be migrated using this procedure, all the information stored in an issue's fields will be migrated.

SOAP Extraction and Insertion - Swizzle

JIRA provides a SOAP API to query and update its data. Using this API it is possible to pull information from one JIRA instance and push this information to another instance. In order to do this, you will need to create a SOAP Client to carry out this operation. Fortunately there's already been a client contributed by the JIRA community.

Swizzle is an open-source project that aims to extract data from an unconsumable source and turn this data into a consumable form. With regards to JIRA this means extracting data from one JIRA instance and turning it into a form that can be imported by another JIRA instance. Swizzle in fact includes a JIRA Migration module which is aimed directly at making the migration of data from one JIRA instance to another possible.

Swizzle achieves the migration by extracting data via JIRA's SOAP interface and turning this data into a Jelly script, that can be executed on another JIRA instance. In order to carry out a data migration using Swizzle, please follow its instructions as closely as possible.

In order for Swizzle to work correctly, you will need to setup your JIRA instances using the following settings:

- You are not using HSQL as your DB on the destination JIRA.

- XML-RPC is turned on in the source JIRA.
- Issues are publicly viewable (attachment and subtask data fetched via HTML) in the source JIRA.
- The required Issue Types are available in the destination JIRA, and have the same names.
- You have file system access on the destination JIRA.

Despite being able to migrate more data than the CSV method described above, Swizzle still suffers from various limitations:

- Custom fields are not migrated.
- Links between issues are not migrated.
- Issue Watchers are not migrated.
- Permission Schemes are not migrated (a "starter" scheme will be created).
- Groups are not migrated (a yourproject-developers group will be created; anyone who has been assigned an issue will be added to this group).
- Notification Schemes are not migrated (the JIRA default will be used).

Swizzle provides a more complete data migration than a CSV migration; however, it is also more difficult to carry out. This should only be considered by power users who are not afraid to execute Java code from the command line and modify XML files and Jelly scripts directly.

Merging XML Backup or Databases

The procedures outlined in this section are not for the faint of heart. Modifying XML backups or the database directly should only be done if you know what you are doing! It is very easy to get your data into an inconsistent state using this method.

Merging XML backups or modifying the database directly is the most difficult, but can also be the most complete data merge mechanism described in this guide. This section will not list a definitive guide for how this can be carried out (as this would simply be too difficult to list consistently for all the different versions of JIRA), but provide a few pointers and external resources that will be useful.

When merging JIRA data, the main things to worry about are:

- Ensure no clashes occur in primary keys of any entity (e.g. users, groups, issues).
- If primary keys are changed, ensure that all the relevant foreign keys are changed.
- Ensure that each issue key and project key is globally unique.
- Ensure that global items are not duplicated. For example, JIRA supports only one mail server, so please ensure that only one is used. Similarly, the global options, such as 'whether unassigned issues are allowed' must be defined only once.
- Ensure that, after the merge, each entry of the SequenceValueItem entity has seqId value set to a larger value than the maximum primary key value of the corresponding entity. The entries for the SequenceValueItem are used to generate the next primary key of a new record.
- Review the Database Schema for JIRA.

A set of instructions has been contributed by the JIRA community that makes this procedure easier, using XML backups produced by JIRA. Please follow the instructions outlined here for merging two JIRA XML backups safely.

Merging two JIRA instances via XML or the database is by far the most difficult method. It involves a lot of manual steps and has by far the greatest potential to corrupt your data. It may, however, yield the most complete data migration of all the procedures described here.

Group Name Guidelines for JIRA

We do not have a formal set of naming conventions for groups in JIRA. However we do have some current, or recently resolved, issues related to group names in JIRA, and based on those issues, we suggest at least the following guidelines:

- Don't use commas: JRA-12675
- Don't use ampersands (&): JRA-13780
- Keep group names to less than 60 characters: JRA-13329
- Don't use group names with only one character in JIRA versions prior to 3.12.3: JRA-14495
- Don't use '#' characters in JIRA versions prior to 3.12: JRA-13509
- We suggest standardising on lower case names for groups: JRA-13798, JRA-5434

Beyond those guidelines, our more general recommendation is to keep group names simple, preferably restricting them to alphanumeric characters, and '-', '_' or a space for word separators - e.g. "jira-users".

If you use non-ASCII characters in your group names, ensure that your database character encoding scheme supports those characters. For MySQL, ensure that the database has a character set encoding of UTF8 by following our instructions for creating the database.

If you are integrating JIRA with LDAP, ensure that you conform to any naming restrictions imposed by your LDAP server.

Where are the JIRA logs?

Please always provide us with both:

1. `atlassian-jira.log`

2. the application server log file, as it can contain useful error information that is not in `atlassian-jira.log`

Finding the application server log file can be a frustrating process, because its location is application server-specific and in some cases operating system-specific. Here is a decision tree:

- **If you are on Windows**
  - ..using Tomcat (**JIRA Standalone** or JIRA EAR/WAR running in **Tomcat**)
    - ..installed as a **Windows Service**:
      - ..then the logs are in the `logs\stdout_*.log` file under your JIRA Standalone directory, or for JIRA EAR/WAR, under your Tomcat directory
    - ..started via `startup.bat`:
      - ..then some logs are effectively being lost (to the popup DOS window, where it cannot be recaptured). Some logs do go to the `atlassian-jira.log` file in the current directory (wherever you ran `startup.bat` from) but this might not work if your current directory isn't writeable (eg. c:\WINNT\system32, the default). Even if you see an `atlassian-jira.log` it may be an old one, created from a previous startup. **If you value your sanity** (and ours) please install JIRA as a service, even if only to get all of the right logs appearing in a consistent place.
  - ..and you are running another application server (JBoss, Websphere, Orion, etc):
    - logs are usually found in a `logs` directory. We assume you know more than us here - if not please use Standalone, where we can at least help.

- **If you are on Unix** (Linux, Solaris, etc)
  - and you are using Tomcat (**JIRA Standalone** or JIRA EAR/WAR running in **Tomcat**)
    - ..then the logs are in `logs/catalina.out` under your JIRA Standalone directory, or for JIRA EAR/WAR, under your Tomcat directory
  - you are running another application server (JBoss, Websphere, Orion, etc):
    - ..if you don't know where your app server logs to, you can often discover this by running 'ps axuwwww' (linux) or 'ps -fax' (solaris). See if it redirects the application server process' stdout to a file, or go to its directory and look for logs there.

Escalating issues (or sending email notifications) when the set turnaround time is exceeded

Can JIRA send notifications based on a set issue turnaround time being exceeded? Can it automatically escalate issues that have exceeded a preset turnaround time?

No, not out-of-the box — but this is exactly what services are for. In particular, a Jelly script can be written to find and escalate relevant issues, and the Jelly script can be run every day via a Jelly Service.

Why doesn't JIRA have a Severity field like Bugzilla?

Originally, JIRA did have both a Priority and a Severity field. The Severity field was removed for a number of reasons, but principally because it was confusing to business users. To a software developer, it seems obvious that the severity of the bug ("The system crashes completely") is unrelated to the priority of it ("There is a one in a million chance of this occurring"). However, JIRA succeeds so well because business users can actually use it. If you present a business user with these two fields, they are instantly confusing (which is why the Severity field was removed).

In order to re-implement Severity, you can create a select-list custom field, order it (with field layouts), put it on your filters (with column layouts) and indeed search and filter it (in the Navigator).

For details, please see Custom Fields.

How to search by number range in the Issue Navigator

When searching for some issues based on the "Number Field" custom field type, there may be a requirement to search issues for a range of numbers based on the custom field.

The "Number Field" custom field type can be configured to use the "Number range searcher". Hence, the issues can be searched within a range of values for that customfield. The search template can be changed by clicking on the "Edit" operation via Administration -> Issue Fields -> Custom Fields. For more information, please refer to the custom field documentation

Please note that a re-index needs to be performed via Administration -> System -> Indexing after changing the custom field searcher.

Troubleshoot Mail Notification Problems

JIRA has a very flexible built-in facility for sending out email notifications under a wide variety of conditions. This guide is meant to help troubleshoot when email notifications are not being received.

Symptoms

No email notifications are being received by anyone

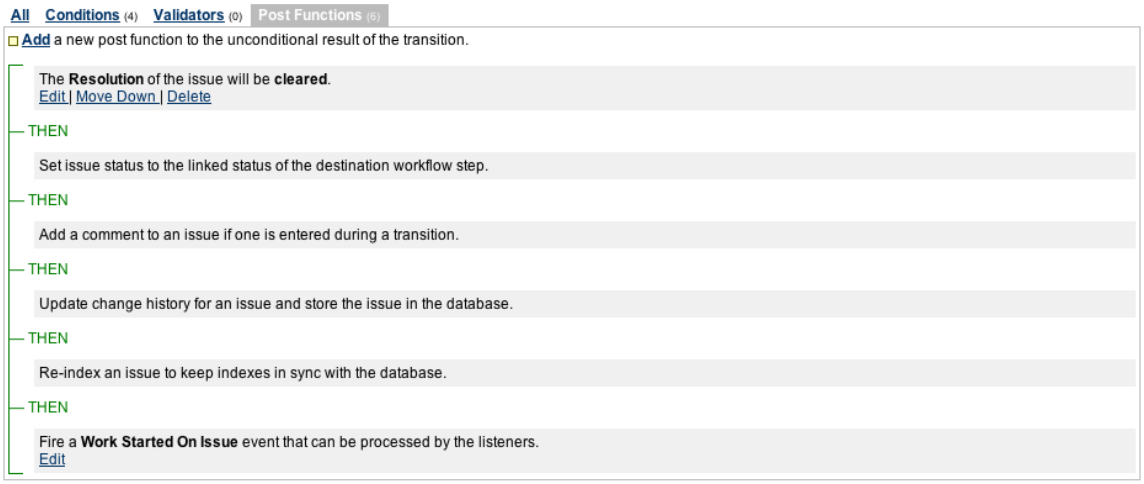
- Ensure you've properly configured an SMTP Server. Send a Test Mail inside the SMTP Server configuration setup screen. Make a note of any error that is returned from the test.
- Check JIRA's log files and the application server log files for Out of Memory errors. Typically, the log file will show **java.lang.OutOfMemoryError: Java heap space**. This has been known to cause the service responsible for sending emails out to fail until JIRA is restarted. You should further troubleshoot your memory issues using this guide.
- Check and ensure the Mail Queue Service is installed. Click *Admininstration* > *Services* to inspect that the service exists, and is set at a reasonable interval. This interval controls how frequently the mail queue is processed. You can flush the mail queue to send out pending messages immediately to your mail server.
- Inspect your Mail Queue under *Administration* > *Mail Queue*. See if you are given the option to *Bypass currently sending mail*. A stuck email or trackback ping can hold up the queue. See JRA-7903.

Email notifications are not being received by a specific group of people/person

- Check if the **Email me when I make changes** option in the user's profile is *true*. The default value for this property for all users can also be set under *Administration* > *User Defaults*
- Check the project permissions under the project's permission scheme. Users must have Browse Project permission for the project to which the issue belongs. If you're using Issue Security Levels, ensure they are members of any Issue security levels that have been applied to the issue.
- Inspect the Mail Queue under *Administration* > *Mail Queue* and see if notifications are being queued. You can adjust the Mail Queue Service interval to a longer period for troubleshooting purposes. Inspect the **Error Queue** tab for any errors.

Notifications aren't occurring after clicking through a transition/workflow step

- Notifications are triggered in a transition by the its post-function.
    - Inspect the post-function of the transition step you're investigating and ensure it's firing an event. This example shows the **Work Started On Issue** event.



    - Inspect your notification scheme and check that the event **Work Started On Issue** actually has people mapped there.
    - Ensure the notification scheme you just checked is actually assigned to your project.

Please see the documentation on JIRA Events, workflow Post-Functions, and Notification Schemes as they are all related.

The content of the email notification is wrong

- If the content of the notification refers to an invalid or non-existent issue, then the notification may be coming from another source. This situation can happen when you restore a XML backup of your production JIRA instance into a development/test server. The development/test JIRA server then begins sending out notifications in addition with your production JIRA instance. Please see this guide on Disabling email sending/receiving for a Development/Test JIRA instance.
- If users are receiving messages in HTML or Text and they wish to change this preference, have them change this property in their user profile under **Outgoing email format**.
- If the URL links inside the content of the notification points to the wrong site, check your **base URL** property under *Administration* > *General Configuration*.
- If the *FROM:* header of the mail being received is not what you've configured under *Administration* > *General Configuration* > *Email from*, check to see if the Project's Mail Project setting is set. It will override the global setting.

Issues created from e-mail do not generate notifications

- Please check your mail handler configuration and ensure that **notifyusers** is set in the handler parameters.

General Troubleshooting Tips

- Reviewing JIRA's logs will greatly help you narrow down the problem. Often, the problem exists in the mail server and a Google search of the error from the logs can help you identify the cause.
- The **Mail Queue** (under *Administration* > *Mail Queue*) can give you a general idea of how many emails are being generated.
- You can get more detailed logging under *Administration -> System -> Logging & Profiling*. Click Edit next to the com.atlassian package and change it from the WARN logging level to DEBUG. This change does not persist after a restart.
- For even more detailed logging (display the message headers, protocol details), see Logging email protocols.

Related Links

- Troubleshooting Issue Creation Via Email

Resolved issues appearing in Open issues filters

Symptoms

If you find that supposedly "Resolved" issues are appearing in an open issues filter, your customized workflow may not be properly configured.

JIRA regards an "open" issue to be one without a resolution. With a standard JIRA workflow, this means all statuses except Resolved and Closed. When Resolving or Closing an issue, you are presented with a transition screen containing the Resolution field, which you must set to complete that transition. See the default workflow as an example.

It is possible to reconfigure JIRA such that Resolved issues, for example, do not have a resolution. This can happen in two ways:

- The user creates a custom workflow, and doesn't prompt the user for a Resolution on the resolve screen.
- The user hides the Resolution field in the field configuration, so it never appears to users.

Resolution

The long-term solution is to fix the workflow to present the Resolution at every transition into a non-open status. If the Resolution field is hidden, it should be made visible.

Existing issues in Resolved or Closed that have no resolution can be fixed by reopening and reclosing with a resolution.

Here is the process, assuming issues in Resolved and Closed statuses without a resolution set:

1. Do a search for issues in status "Closed", with Resolution "unresolved", which will show affected issues.
2. On the right, you'll see the text "Bulk Change: all X issue(s)". Open that link twice, eg. the same page in two tabs or two browser windows.
3. In one page:
   o Click the checkbox to select all issues, and click "Next >>".
   o For Operation, choose "Transition Issues", and then choose "Reopen Issue"
   o Uncheck "Send mail for this update"
   o Click confirm.
4. Now in the second page (displaying that original set of issues):
   o Click the checkbox to select all issues, and click "Next >>".
   o Operation, choose "Transition Issues", and then choose "Close Issue"
   o Select a resolution (eg. "Fixed").
   o Uncheck "Send mail for this update"
   o Click confirm.

By doing this you have reopened and closed the issues, setting a resolution. The Closed issues should now no longer appear on your dashboard.

Repeat the same process, but selecting "Resolved" issues this time (and transitioning through Reopen and Resolve).

Parsing utf-7 emails

Some users report having problem parsing `unicode-1-1-utf-7` (aka `utf-7`) emails. JIRA breaks with a stacktrace like:

```
2007-01-31 12:54:59,176 JiraQuartzScheduler_Worker-2 ERROR [service.util.handler.CreateIssueHandler]
Could not create issue!
java.io.UnsupportedEncodingException: unicode-1-1-utf-7
at sun.nio.cs.StreamDecoder.forInputStreamReader(StreamDecoder.java:52)
at java.io.InputStreamReader.<init>(InputStreamReader.java:83)
at com.sun.mail.handlers.text_plain.getContent(text_plain.java:64)
at javax.activation.DataSourceDataContentHandler.getContent(DataHandler.java:774)
at javax.activation.DataHandler.getContent(DataHandler.java:521)
at javax.mail.internet.MimeBodyPart.getContent(MimeBodyPart.java:603)
at
com.atlassian.jira.service.util.handler.AbstractMessageHandler.handleMultipart(AbstractMessageHandler.jav
com.atlassian.jira.service.util.handler.AbstractMessageHandler.createAttachmentsForMessage(AbstractMessag
com.atlassian.jira.service.util.handler.CreateIssueHandler.handleMessage(CreateIssueHandler.java:201)
at
com.atlassian.jira.service.util.handler.CreateOrCommentHandler.handleMessage(CreateOrCommentHandler.java:
com.atlassian.jira.service.services.mail.MailFetcherService.run(MailFetcherService.java:190)
at com.atlassian.jira.service.JiraServiceContainerImpl.run(JiraServiceContainerImpl.java:67)
at com.atlassian.jira.service.ServiceRunner.execute(ServiceRunner.java:48)
at org.quartz.core.JobRunShell.run(JobRunShell.java:191)
at org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:516)
```

The solution

Hopefully one day, Sun will include support for this encoding natively (see this Sun bug report, but in the meanwhile you can install a library to get this working. Installation is quite simple:

1. Download the jutf7 jar from http://sourceforge.net/projects/jutf7
2. Copy the jar to your $JAVA_HOME/jre/lib/ext directory. **No other directory will do** - it has to be in this (lowest)classloader to be picked up.
3. Restart JIRA (or Confluence, or whatever is parsing the emails).

Adding custom content to the front page

Custom HTML content can be easily added to the dashboard by a JIRA administrator.

For example, to customise the text that appears on users' dashboards, click on General Configuration], click '**Edit Configuration**', and edit the **Introduction** field.

Note that look and feel can also be customised (e.g. add your organisation's logo and/or preferred colour scheme).

Additionally, the announcement banner is useful for sending broadcasts to all JIRA users.

Changing the default session timeout

To change the default session timeout (which is 60 minutes) you must edit the file `web.xml`. This file can be found in `<YOUR DEPLOYMENT>/WEB-INF/web.xml`. If you are deploying JIRA as a closed `.war` file you will need to unzip the .war, edit the file, and re-create the `.war` with exactly the same structure as it originally had.

The element you want to edit in the `web.xml` file is:

```
<source>
    <session-config>
        <session-timeout>60</session-timeout>
    </session-config>
</source>
```

The value within the **session-timeout** tag defines the amount of time the session will exist, in minutes.

Note that after editing the `web.xml` file you will need to restart JIRA for your change to take effect.

How is JIRA pronounced?

We pronounce it 'JEEra', based on the pronunciation of 'Kujira' (see What does JIRA mean?)

Translating JIRA

What translations of JIRA are currently available?

Currently, JIRA ships with a number of translations in the most commonly-requested languages.

As a JIRA administrator, you can choose the default language from the list of installed languages: see Choosing a Default Language for the latest list.

Individual users can also choose their preferred language from the same list: see Choosing a Language.

How are new translations of JIRA created?

JIRA's internationalisation is an ongoing effort, and we'd love your help. If you would like to translate JIRA into your language, the rest of this document describes how to do so.

- Where to begin?
- Translating into multi-byte languages
- Translating words
- Translating sentences
- Translating the javascript calendar popup tool
- Packaging the translation
- Installing a new translation of JIRA
- Submitting a translation

Where to begin?

In order to get JIRA to appear in your language you need to translate all the properties files located inside `language_default.jar`. The `language_default.jar` is found inside the `WEB-INF/lib` directory.

> **Note**
> The latest properties files for the next (unreleased) JIRA version are available from http://www.atlassian.com/software/jira/translations.

Translating JIRA's default English property files begins with extracting the contents of the `language_default.jar` into a temporary directory (using a standard Unzip utility of 'jar xvf' from the command-line). After extracting the jar, make a copy of each property file, giving it a new name that contains locale information. For example, if you are translating the `Dashboard.properties` file into French, save all changes to a separate `Dashboard_fr_FR.properties` file. That is, the format of the file name is:

```
<Default Name><Language Code><Country Code>.properties
```

> **Note**
> In some cases you might need to modify language_en_UK.jar - JRA-8266 in order to modify the default English translation.

See Translating into multi-byte languages (below) for a slight modification to this procedure, if applicable.

The "Language Code" is a valid ISO Language Code. These codes are the lower-case, two-letter codes as defined by ISO-639.

The "Country Code" is a valid ISO Country Code. These codes are the upper-case, two-letter codes as defined by ISO-3166.

> **Note**
> The original directory structure of the property files should be preserved. For example, when creating `com/atlassian/jira/web/action/Dashboard_fr_FR.properties` file, it must be placed in `com/atlassian/jira/web/action directory` (relative to the temporary directory you are working in).

Translating into multi-byte languages

Languages like Chinese and Russian are represented with native two-byte encodings like gb2312 or koi8. Java `.properties` files are assumed to be Unicode encoded as ISO-8859-1. Therefore entering Chinese or Russian characters directly into the `.properties` files will not work.

The solution is to create the translation as a separate file:

```
<Default Name><Language Code><Country Code>-native.txt
```

Once completed, these can be transformed into regular `.properties` files with the 'native2ascii' command. E.g. for Russian:

```
native2ascii _-encoding cp1251 JiraWebActionSupport_ru_RU-native.txt
```

Translating words

Properties files contain many "entries" for each of the words and sentences used in JIRA. The entry for the common word "Project" looks like this:

```
common.concepts.project = Project
```

To translate this into French (for example), you would replace the English word "Project" with "Projet", ie:

```
common.concepts.project = Projet
```

Note
Not all entries for a particular page in JIRA will appear in the natural properties file associated with that page. For example, on the "Create Issue" page, entries for "Project" and "Issue Type" cannot be found in the `CreateIssue.properties` file. Some words in JIRA appear in more than one place and have therefore been placed in a single properties file. This saves you from having to translate common words more than once. The name of this file is **`JiraWebActionSupport.properties`**. The location of the file is in **atlassian-jira/WEB-INF/lib/language_default.jar**. Once you find the jar unzip it. Once inside the jar file go to **language_default/com/atlassian/jira/web/action/JiraWebActionSupport.properties**.

Translating sentences

Some sentences in JIRA have certain words that are HTML links *OR* constitute database data. For example:

```
issue.operations.move = {0}Move{1} this issue to another project
```

OR

```
roadmap.issuesresolved = {0} of {1} issues have been resolved
```

In the first example, the word "Move" links to the "Move Issue" page. The HTML code for this link is passed in using {0} and {1}. In the second example, {0} is the number of resolved issues and {1} is the total number of issues. When translating sentences, ensure that you do not omit these elements. A French translation of the first example above might look like this:

```
issue.operations.move = {0}Déplacer{1} cette demande vers un autre projet
```

Note
Any HTML needs to be escaped, e.g.
use

```
&amp;lt;
```

not

```
<
```

Translating the javascript calendar popup tool

In addition to the JIRA-specific translation files, you may want to see if there is an existing translation for the ***javascript calendar utility*** that JIRA uses as a date picker. You can find these files in `atlassian-jira/includes/js/calendar/lang/`. Many languages already have a translation; it is a good idea to look over the translation and see if the months correspond to the correct months that java expects for a date in your locale. If you add a new `calendar_{LANG}.js` file, you will need to add an entry mapping the filename to your language code in the file **`atlassian-jira/WEB-INF/classes/calendar-language-util.properties`**. For example:

```
en=calendar_en.js
```

You will also need to add the javascript file as a downloadable web resource. Modify the file `system-webresources-plugin.xml` to include the following:

```
<web-resource key="calendar-{LANG}" i18n-name-key="admin.web.resources.plugin.calendar.{LANG}.name"
name="Calendar"
                  state='enabled'>
        <resource type="download" name="calendar-{LANG}.js" location=
"/includes/js/calendar/lang/calendar-{LANG}.js">
            <param name="source" value="webContextStatic"/>
        </resource>
    </web-resource>
```

Packaging the translation

The properties files for each locale are collected into language bundles. Once you have finished translating all the properties files, a language descriptor file has to be created. This file "tells" JIRA that a new language is available. The file **MUST** be called **`language-descriptor.xml`** and should be located at the root of the language bundle (i.e. directly in the temporary directory). The `language-descriptor.xml` tells JIRA which locale (language and country) the property files represent, and the JIRA version for which the translation was last updated. The `language-descriptor.xml` file looks like this:

```
<language>
    <locale>en_UK</locale>
    <version>2.6</version>
</language>
```

The `locale` tag must contain a proper Java locale name. The locale name is made up of the Language Code and Country Code (please see the Where to begin? section for more details).

- Set the value of the contents of the `locale` tag to the locale that your translation represents.
- Set the contents of the `version` tag to the version of JIRA for which you are doing the translation.

Create a jar ('`jar cvf language_<locale>.jar ...`') which contains all the translated property files (remember to preserve the directory structure) and the `language-descriptor.xml` file. Although not absolutely necessary, it is a good idea to call this file `language_<locale>.jar`, where `<locale>` is the contents of the locale tag in the `language-descriptor.xml` file. For example, for French call the file `language_fr_FR.jar`.

Installing a new translation of JIRA

To make JIRA aware of your translation, you will need to copy the jar file that you created in the previous step (see Packaging the translation into the `WEB-INF/lib` directory found under JIRA's web application directory.

If you would like JIRA to display a flag for your translation during the setup process, place a GIF image with the same name as the locale into the `/images/flags directory` under JIRA's web application directory. For example, for the French translation place a file called `fr_FR.gif` into the `/images/flags` directory.

- If you are installing JIRA for the first time, the first page of the Setup Wizard should now display your newly installed language.
- If you are already using JIRA, you can choose your newly installed language from the list of installed languages: see http://www.atlassian.com/software/jira/docs/latest/i18n.html.

Submitting a translation

If you would like your translation to be included in JIRA, please create an issue on jira.atlassian.com and attach the `jar` containing the properties files.

Reporting a translation issue

-
- Have you noticed an issue with an existing translation? Please file an issue!

How to ensure the Road Map tab is visible

If the Road Map project tab is not visible for a particular project, it may be for one of the following reasons:

- The Road Map Panel (roadmap-panel) module is disabled
- The "Fix Version/s" field is hidden via at the Field Configuration

So, if you are not seeing this tab, but you want to, ensure:

- The Road Map module is enabled under *Administration -> System -> Plugins -> Project Panels Plugin*
- The "Fix Version/s" field is not hidden under *Administration -> Issue Fields -> Field Configurations*

How to have long component version names display properly in the Issue Navigator

If you create a component or version name which is very long, it is not possibly to view it in the issue navigator. If you're finding this happening in your Jira instance, you can set the width of the component or version list to auto and wrap it in a element with fixed width in this file:

- WEB-INF/classes/templates/jira/issue/searchers/edit/project-constants-searcher-edit.vm

Here is a code sample:

```
...
<div style="width: 180px; overflow-x: scroll; border: 1px #ddddff solid;">
<select .. style="width: auto;">
...
</select>
</div>
...
```

Login problems

I have manually reset a user's password, but the user still cannot login

Check (in Admin -> Global Settings -> Global Permissions) that the user belongs to a group that has the **JIRA Users** permission.

The user cannot get past the login page. After clicking the "Log In" button, the login page just refreshes.

This usually occurs when JIRA cannot set a browser cookie. Ensure that cookies are allowed in the user's browser settings.

If you are using IE6, check that your server name does not have an underscore ("_") in it, as IE6 has a problem with this (see JRA-1624).

We already have users & groups defined elsewhere - can JIRA make use of these?

Yes. If you have users and groups defined elsewhere then you can either use an existing OSUser provider (such as LDAP or JDBC) or write your own if you have custom needs.

Associating a new screen with issue operations

When you spend some quality time creating a new screen with custom fields, you want to make people use it. This quick and dirty how-to helps make your screen implementation dreams come true!

1. The Screen

Create new screens to change the order of fields or add new custom fields to your display. Screens can be associated with issue operations (e.g. *Create New Issue*) or workflow transitions (e.g. *Start Progess*).

1. Create the screen you would like to use.
   Administration->Issue Fields->Screens

2. The Screen Scheme

A Screen Scheme allows you to hook basic Issue Operations (View, Edit, Create) to specified screens. You can either use the same screen for each operation or live on the bleeding edge and assign a different screen for each operation.

1. Create a screen scheme that uses this screen.
   Administration->Issue Fields->Screen Schemes->Add Screen Scheme
2. Once you have added the screen scheme, configure the screen scheme by clicking **Configure** next to the newly listed screen scheme.
3. Under the **Add Issue Operation To Screen Association** section, add each operation you would like to associate with this screen.

3. The Issue Type Screen Scheme (Enterprise Version Only)

Issue Type Screen Schemes allow you to specify which Screen Scheme is associated with which issue type (e.g. *Bug*)

1. Create a new Issue Type Screen Scheme
   Administration->Issue Fields->Issue Type Screen Scheme
2. Under the **Add Issue Type Screen Scheme** section, add a new Issue Type Screen Scheme that uses your Screen Scheme.
   - **Extra credit:** You can click **Configure** next to your newly added Issue Type Screen Scheme in case you'd like to associate different Screen Schemes with different Issue Types. In this case, using the default setting is good enough.
3. Associate the Issue Type Screen Scheme with your project
   Administration->Projects
4. Click on the name of the project you wish to modify.
5. Under the Project, click the **Select** link next to **Issue Type Screen Scheme**.
6. Select your newly created Issue Type Screen Scheme and associate it with the specified project.

Did it work?
Issues in the specified project should now be using your customized screen. Test they are by creating, editing and viewing issues. The screens should be shown for the issue operations you assigned to your screen in Step 2.

CVS ssh Jira Integration

CVS :ext: ssh Jira Integration

The following information is probably only relevant to Linux/Unix/OSX/Cygwin environments.

CVS_RSH environment variable

In order to use the :ext: method for connection to CVS, the CVS_RSH environment variable needs to be set in the environment that runs JIRA. It should be set to the path to the ssh binary.

put this in your profile

```
CVS_RSH=`which ssh`
export CVS_RSH
```

Problems Authorising when command line works

One user reported the following:

> *The problem was found to be the UsePAM directive in sshd_config on the cvs server(Debian-Sarge) - this needs to be disabled (which it wasn't) with the PasswordAuthentication enabled.*

JIRA as a Support System

This document shows how to set JIRA up as a support system:

- JIRA Advantages
- Feature Setup Instructions
  - Permissions
    - 1. Different Projects
    - 2. Issue Level Security (Enterprise only)
  - Post-Ticket Pages
  - Work Queues

    - Email Integration
    - Custom Landing Page
    - Support Knowledge Base
    - Log Phone Calls To Tickets
    - SLAs
    - Escalate Tickets
    - Change Status After Comment
- Example Scenario
    - Further Support Discussion
    - Related Best-Practice Discussions

JIRA Advantages

- By using one system for support and bug / feature tracking, you can link support issues to the bugs that they reference.
- JIRA is a very simple system to install and use - there is very little training required for support staff, or end users
- JIRA's configurable workflow adapts to your existing support processes

Feature Setup Instructions

Note that some terminology is different between the two systems - for example a support system typically uses the word 'ticket' where an issue tracking system may use the word 'issue'.

Permissions

A support system has different need for permissions than a bug tracking system. Typically as an end user you can only see issues that you, or your company has raised. The ways of doing this are:

1. Different Projects

At a very simple level, if you are supporting a very limited number of clients, you can set up a different project for each of your clients, with a different permission scheme for each project.

2. Issue Level Security (Enterprise only)

You can set up different security levels for each customer. This is similar to having different projects, but allows the support team to manage the issues in just one project.

You can set up the permission schemes so that only the reporter of an issue, and the support staff can see the issue. This means that each user can only see their own users, and is very suited to an internal help desk system, or any other support system with a large number of end users.

Post-Ticket Pages

If you have JIRA **Standard** or **Professional** and need to let customers raise issues, but not see each others', then it is possible to redirect users to a custom 'thank you' page after raising issues.

Work Queues

Often in support systems, the priority of an issue is not as important as the order in which the issues are raised. There may be a Service Level Agreement in place, which specifies that an issue must be responded to within a certain time.

The JIRA toolkit will show you whether the last commented was from a JIRA Administrator, or whether it was from a customer. This allows issues to be prioritised by the order in which they need a response.

Email Integration

JIRA can easily be set up to handle incoming email, and create new issues, or comment on existing issues. It also sends mail notifications to users when the issue has been updated.

When setup this way, the client can create and comment on an issue, without having access to JIRA.

For more information, see the documentation on Setting up email integration in JIRA — particularly the CreateOrCommentHandler.

Custom Landing Page

You are able to setup a supprt-friendly landing page explaining how to log tickets using the Improved HTML Plugin.

Support Knowledge Base

Please see Adding Knowledge Base Functionality To JIRA.

Log Phone Calls To Tickets

Please see Logging Phone Calls In JIRA.

SLAs

Most SLAs are very specific to a particular organisation, so it is difficult to ship a completely out-of-the-box solution with JIRA that will meet everyone's needs. However JIRA has 2 approaches that can be used separately or jointly to meet SLAs:

1. The most powerful approach is to write a Jelly script (sample available) which invokes a saved search (filter), and loops over the issues, adding a comment, transitioning them to a new state (e.g. "Requires Response"), or otherwise letting people know that action needs to be taken. This Jelly script would then be run periodically by a Jelly runner service. Atlassian uses this approach on https://support.atlassian.com, to automatically close issues that have not been replied to in X days. We have a filter returning issues in status "Waiting for Customer", updated from any time to 1 week ago (i.e. not touched in the last week), and these are transitioned to "Inactive", which triggers an email letting the customer know.
2. Create a search filter that finds all issues that meet a certain criteria. Save this filter and subscribe to it, either by email (through JIRA) or by subscribing to the filter's RSS feed in an RSS reader. This way JIRA will notify subscribers what issues are 'outstanding'. For more information on creating and saving filters and subscriptions please see this page. There is also a short video on Simple SLA with Filters.
3. If a Jelly script cannot do what you want, or JIRA's searching capabilities are not sufficient to match issues you want, you could write a custom service that locates issues that meet a certain criteria and then does something with matching issues. For example, a service could reassigns the issues to another team member (e.g. project's lead), increments priority, sends notifications, etc. For more information on JIRA services please see this page.

Escalate Tickets

For an example of code that uses JIRA's API to escalate issues please see: Simple Escalation.

Change Status After Comment

A user adding a comment via the JIRA UI can be prompted to change the issue status. The source is not yet available as this is currently a work in progress but please visit Adaptavist for updates.

Example Scenario

Here is an example scenario for a support environment within an organisation and suggestions on how to setup JIRA to fit this environment.

1. End-users: company workers place phone calls to the 'hot-line' team.
2. Hot-line: answer the end-users and open a ticket for every issue
3. 1st level Help Desk: analyse hot-line tickets, and close them if they are able to respond themselves. Otherwise they dispatch the ticket to one of three 2nd level help desk teams.
    a. Technical 2nd level help desk
    b. Functional 2nd level help desk
    c. Logistic 2nd level help desk

The best way to setup JIRA for the above environment is to create a separate JIRA project for each of the four support groups (one 1st level support team and three 2nd level support teams). It would also be useful to create a separate permission scheme for each project so that permissions can be managed for each project separately. For more information on permissions please see:
Managing Project Permissions

The hot-line team will create a new issue in the 1st level support team's dedicated project (referred to as 'hot-line' project from here on) for every call they receive. The way the hot-line project should be setup depends on whether the actual end users need to see JIRA issues. If yes, ensure that every member of this hot-line team has Modify Reporter permission so that they can set the 'reporter' of the issue as the actual end caller.

It is also possible to create a custom field of type User which can be used to track who (which member of the hot-line team) actually created the issue. The hot-line team member will have to populate this field with their username. For more information on custom fields please see:
Adding a Custom Field

You can then give the Browse Project permission in the hot-line project's permission scheme to the 'Reporter' role (please see the permission documentation referenced above for more details) and 2 user group . One user group will represent represents the hot-line team and the other the 1st level support team. This way, the end users can see issue created on their behalf, but not issue's created for other users. The hot-line group members and the 1st level support team will be able to see all issues in the project.

If the actual end users do not need to see the issues in JIRA it is probably better to not give the Modify Reporter permission to anyone for the hot-line project. The reporter field of the issue will then automatically default to the logged in user (i.e. the hot-line group member who is creating the issue). A custom field of type User can still be created and used to record on whose behalf the issue was created. The field will have to be populated manually during issue creation. This custom field can also be made 'required' so that it will have to be populated during issue creation.

The user group representing 1st level support team should be given the resolve and close issue permissions so that they can resolve/close issues once they are dealt with.

I also recommend setting the 'Assignable User' permission in the hot-line permission scheme to the user group representing the 1st level support team, so that issues can be assigned to them. The 'Assign Issue' permission can be given to the hot-line group so that its members can assign issues to specific 1st level support team members.

Alternatively, the 'Assign Issue' permission can be given to only the 'Project Lead'. The default assignee of the hot-line project can be set to 'Project Lead' or 'Unassigned' (if unassigned issues are enabled. Then the hot-line project's lead can go through all the issues assigned to him/herself or all Unassigned issues and assign them appropriately.

If the 1st level support team members cannot resolve an issue they should create a new issue in one of the other three projects (the technical support project, the functional support project, logistics support group project) to
indicate that the issue has been passed to the 2nd level support. For this to occur the 1st level support team must be given the 'Create Issue' permission in the permission schemes used by these projects.

The issues created in the 2nd level support projects should be linked to the issue in the hot-line project
using Issue Links:
[Configuring Issue Linking]

Each of the 3 support projects can be setup as required by each team, in terms of their permissions, notifications, workflows, etc.

If all internal users are stored in a LDAP directory, please take note of JIRA's LDAP integration:
Integrating JIRA with LDAP

JIRA's customisable workflow can also be very useful:
Configuring Workflow

The workflow can be customised for each project (in JIRA Enterprise), and can be used to better reflect the business process of each support team in JIRA. For example, if issues can only have 2 stages (Open and Closed) then it is far better to create and use a custom workflow rather than use the JIRA's default workflow.

Using JIRA's flexible plugin system it is also possible to extend JIRA's functionality in regards to workflow. One place where this can become useful, is when closing issues in the hot-line project that have linked issues in one or more of the 2nd level support projects. It is possible to write a custom Workflow Condition that will look at all the linked issues and only allow an issue to be Closed when the linked issues are also closed. This will ensure, that the issues in the hot-line project
are only closed when the linked issues are handled by the respective 2nd level support team. For more information on creating custom workflow elements (e.g. Workflow Conditions) please see:
How to create Custom Workflow Elements for JIRA 3

If one of the support teams also has an existing support system in place that they would like to continue using, it should be possible to integrate that system with JIRA. JIRA has a number of extension points that can be used to communicate (and hence integrate) with external systems:
Extending JIRA

By default, JIRA related issue links do not affect workflow, so users can close issues even if other open issues are listed as blocking it. You can enforce the rule that all blocking issues must be resolved before you can resolve the parent issue using the custom 'blockingLinksClosed Condition' workflow plugin.

Further Support Discussion

- How Atlassian Uses JIRA For Support
- Example Helpdesk/Support System Project Configuration

Related Best-Practice Discussions

JIRA as a Support System

Confluence UI Guidelines

Using JIRA for Agile Development

Confluence UI Guidelines

Confluence UI Guidelines

This document is a work in progress. Feel free to add any comments at the bottom.

Jelly Escalation

Below are the 2 jelly scripts used by Jira's support system to automatically close issue after a certain period. These jelly scripts are then run with the built in Jelly Service.

Make an issue inactive

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:core="jelly:core"
xmlns:log="jelly:log" >
<jira:Login username="atlassiansupport" password="[your password]">
    <log:warn>Running Inactivate issues service</log:warn>
    <!-- Properties for the script -->
    <core:set var="comment">This issue has not been updated for 5 business days.

If you have an update, please use "Add Comments For Atlassian" action to let us know. If you need more
time to gather information please let us know and we will 'freeze' this issue. If you have no other
questions, please Close this issue.

If no update is received in the next 5 business days, this issue will be automatically closed.

Thank you,

  The Atlassian Support Team</core:set>
    <core:set var="workflowStep" value="Mark Inactive" />
    <core:set var="workflowUser" value="atlassiansupport" />
    <core:set var="filter7Days" value="11505" />

    <!-- Run the SearchRequestFilter -->
    <jira:RunSearchRequest filterid="${filter7Days}" var="issues" />

    <core:forEach var="issue" items="${issues}">
        <log:warn>Inactivating issue ${issue.key}</log:warn>
        <jira:TransitionWorkflow key="${issue.key}" user="${workflowUser}" workflowAction=
"${workflowStep}" comment="${comment}"/>
    </core:forEach>
</jira:Login>
</JiraJelly>
```

Close an issue

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:core="jelly:core"
xmlns:log="jelly:log" >
<jira:Login username="atlassiansupport" password="[your password]">
    <log:info>Running Close issues service</log:info>
    <!-- Properties for the script -->
    <core:set var="comment">This issue has not been updated for 10 business days and will be Closed.

If this issue has not been completed please reopen this issue and we will do our best to help you as
soon as we can.

Thank you,

  The Atlassian Support Team</core:set>
    <core:set var="workflowStep" value="711" />
    <core:set var="workflowUser" value="atlassiansupport" />
    <core:set var="filter7Days" value="11509" />

    <!-- Run the SearchRequestFilter -->
    <jira:RunSearchRequest filterid="${filter7Days}" var="issues" />

    <!-- Iterate over the issues -->
    <core:forEach var="issue" items="${issues}">
        <log:warn>Closing inactive issue ${issue.key}</log:warn>
        <jira:TransitionWorkflow key="${issue.key}" user="${workflowUser}" workflowAction=
"${workflowStep}" comment="${comment}" resolution="Customer Timeout"/>
    </core:forEach>
</jira:Login>
</JiraJelly>
```

For more helpings of Jelly, see Jelly Examples.

Simple Escalation

Here is a piece of code that performs simple escalation. The code finds all issues that have been in the 'Open' status for longer than 24 hours and increases the priority of these issues (if there is a higher priority). This code could be used in a JIRA service so that it is performed periodically.

Please note that the code assumes that all the issues use the default workflow. Hence it also assumes that the the step id it should search for in the OSCurrentstep table is 1. If your issues are using a different workflow you will need to see what status id to search for.

```
EntityCondition con = new EntityExpr(
        new EntityExpr("stepId", EntityOperator.EQUALS, new Long(1)),
        EntityOperator.AND,
        new EntityExpr("startDate", EntityOperator.LESS_THAN_EQUAL_TO, new Timestamp(System
.currentTimeMillis() - 24*3600*1000))
);

List steps = CoreFactory.getGenericDelegator().findByCondition("OSCurrentStep", con, null, null);
for (Iterator iterator = steps.iterator(); iterator.hasNext();)
{
    GenericValue stepGV = (GenericValue) iterator.next();
    IssueManager issueManager = ComponentManager.getInstance().getIssueManager();
    GenericValue issueGV = issueManager.getIssueByWorkflow(stepGV.getLong("entryId"));

    // Increase priority
ConstantsManager constantsManager = ComponentManager.getInstance().getConstantsManager();
    GenericValue priority =
constantsManager.getPriority(issueGV.getString(IssueFieldConstants.PRIORITY));
    Collection priorities = constantsManager.getPriorities();
    GenericValue higherPriority = null;
    for (Iterator iterator1 = priorities.iterator(); iterator1.hasNext();)
    {
        GenericValue priorityGV = (GenericValue) iterator1.next();
        if (priorityGV.getString("id").equals(priority.getString("id")))
        {
            if (higherPriority != null)
            {
                // Update issue
issueGV.set(IssueFieldConstants.PRIORITY, higherPriority.getString("id"));
                // Save issue to database and fire an event
GenericValue originalIssue = issueManager.getIssue(issueGV.getLong("id"));
                User updater = UserUtils.getUser("admin");
                IssueUpdateBean issueUpdateBean = new IssueUpdateBean(issueGV, originalIssue,
IssueEventType.ISSUE_UPDATED, updater);
                IssueUpdater issueUpdater = ComponentManager.getInstance().getIssueUpdater();
                issueUpdater.doUpdate(issueUpdateBean, true);

            }

            break;
        }
        else
        {
            higherPriority = priorityGV;
        }
    }
}
```

The above code will make change items of updated issues appear as if they have been performed by the "admin" user. You may wish to create a special user for this task.

Changing the default attachment size limit

To change the default size limit for attachments, see the Enabling File Attachments page.

Importing data

To import issue data from CSV (Comma-Separated Value), Bugzilla, FogBugz or Mantis, please see the documentation:

- CSV

- Bugzilla
- FogBugz
- Mantis

For other types of import, please contact us as we may have done it before. See also JIRA's Jelly support — Jelly is a scriptable interface to JIRA that is useful for importing data.

Tips of the Trade

Below are some links to external blog posts and articles containing technical tips and instructions on setting up and using JIRA. This page presents an opportunity for customers and community authors to share information and experiences.

The references here are links to technical 'how to' guides written by bloggers who use JIRA. For feature tours, solution tours and other information about bug and issue trackers, please refer to the Atlassian website and to our evaluator resources.

Please be aware that these are external blogs and articles.
Most of the links point to external sites, and some of the information is relevant to a specific release of JIRA. Atlassian provides these links because the information is useful and relevant at the time it was written. Please check carefully whether the information is still relevant when you read it, and whether it is relevant to your version of JIRA. Unless explicitly stated, Atlassian does not offer support for third-party extensions or plugins. The information in the linked blog posts has not been tested or reviewed by Atlassian. We recommend that you test all solutions on a **test** server before trying it on your production site.

On this page:

- 'Send reminder on' custom field for Jira
- User Activity Statistics
- Showing custom fields in 'sub-task' columns
- Using User Properties
- Making it easier to maintain JIRA workflows
- Jira Workflow Report Update
- NetBeans 6.7 RC1 - and JIRA support (beta)
- JIRA To Omnifocus Script
- Update JIRA-issues with OmniFocus
- Git branches to handle contributor patches
- Setting issue security level by issue type
- Unique Issue ID Across Projects

Administration

'Send reminder on' custom field for Jira

- By: Sam Haldane, on blog 'blog.samhaldane.com'
- About: How to set up JIRA to send issue reminders to users on specified dates
- Date: 17 August 2009
- Related documentation: Adding a Custom Field, Saving Searches ('Issue Filters')

User Activity Statistics

- By: Zaccary Craven, on blog 'Tips and tricks for JIRA administrators'
- About: How to show a list of all usernames along with the number of times each user has created a comment
- Date: 19 January 2009
- Related documentation: Adding the Filter Statistics Gadget

Showing custom fields in 'sub-task' columns

- By: Zaccary Craven, on blog 'Tips and tricks for JIRA administrators'
- About: How to show the values of subtask custom fields on your issue screens
- Date: 8 December 2008
- Related documentation: Custom fields

Using User Properties

- By: Matt Doar, on blog 'Consulting Toolsmiths'
- About: How to add, display and filter the user properties with the JIRA Toolkit plugin
- Date: 20 February 2008
- Related documentation: Managing Users

Workflow

Making it easier to maintain JIRA workflows

- By: Matt Doar, on blog 'Consulting Toolsmiths'
- About: How to display the name of the screen used by each transition in a workflow in one place
- Date: 1 July 2009
- Related documentation: Configuring Workflow

Jira Workflow Report Update

- By: Jamie Echlin, on the 'onresolve team blog'
- About: Visualising JIRA workflows, with hints about a common problem when defining resolutions in JIRA workflows
- Date: 19 December 2008
- Related documentation: Configuring Workflow

Integration with Other Tools

NetBeans 6.7 RC1 - and JIRA support (beta)

- By: Fabrizio Giudici, on 'Fabrizio Giudici's Blog'
- About: How to get JIRA integration in NetBeans 6.7 RC1
- Date: 1 June 2009

JIRA To Omnifocus Script

- By: David Martinez, on blog 'Hackerdude'
- About: A script that logs into JIRA and creates OmniFocus tasks for each of the JIRA items that are assigned to you, so they sync to your Omnifocus for iPhone, you only have to keep track of one inbox, etc.
- Date: 4 March 2009

Update JIRA-issues with OmniFocus

- By: Alain Petignat, on blog 'sequenz'
- About: Updating JIRA fields (time estimate, due date and assignee) directly from OmniFocus
- Date: 12 April 2009

Development

Git branches to handle contributor patches

- By: Julien Ponge, on "JPz'log"
- About: Managing patches in pending state via Git, SVN and JIRA
- Date: 4 December 2008

Setting issue security level by issue type

- By: Jamie Echlin, on the 'onresolve team blog'
- About: Two ways to set JIRA issue security levels by issue type
- Date: 5 August 2008
- Related documentation: Configuring Issue Level Security

Unique Issue ID Across Projects

- By: Surya Suravarapu, on 'Surya Suravarapu's Blog'
- About: A plugin that allows you to have unique issue IDs across all your JIRA projects
- Date: 14 July 2009
- Related documentation:
  - Changing the Issue Key format
  - Change History

Have you written a technical tip for JIRA?
Add a comment to this page, linking to your blog post or article. We will include it if the content fits the requirements of this page.

Feedback?
Your first port of call should be the author of the linked blog post. If you want to let us know how useful (or otherwise) a linked post is, please add a comment to this page.

Other Sources of Information

JIRA documentation
Evaluator resources
Atlassian website
Atlassian forums
Atlassian blog
JIRA plugins

JIRA Installation and Upgrade Guide

The pages listed below contain information on installing and upgrading JIRA:

- JIRA Requirements
    - Installing Java
- Installing JIRA Standalone
    - Installing JIRA Standalone on Windows
    - Installing JIRA Standalone on Mac OS
    - Installing JIRA Standalone on Unix or Linux
    - Configuring JIRA Standalone
        - Running JIRA over SSL or HTTPS
        - Integrating JIRA with a Web Server
        - Running JIRA Standalone as a Service
        - Installing Confluence into JIRA Standalone
        - Changing JIRA Standalone's port
- Installing JIRA WAR-EAR
    - Installing JIRA on Tomcat 6.0
    - Installing JIRA on Tomcat 5.5
    - Installing JIRA on Oracle WebLogic
    - Installing JIRA on IBM Websphere 6.x
    - JIRA WAR-EAR Configuration Overview
        - Configuring the Entity Engine for JIRA
    - Switching Application Servers
- Running the Setup Wizard
- Connecting JIRA to a Database
    - Recommended Databases
    - Connecting JIRA to SQL Server 2005
    - Connecting JIRA to MySQL
    - Connecting JIRA to PostgreSQL
    - Connecting JIRA to Oracle
    - Connecting JIRA to DB2
    - Connecting JIRA to Firebird
    - Connecting JIRA to Sybase
    - Connecting JIRA to HSQLDB
    - Switching Databases
- Upgrading JIRA
    - Disabling Auto-Export
- Important Directories and Files
    - JIRA Home Directory
    - JIRA Installation Directory
    - Setting your JIRA Home Directory
- JIRA Releases
    - Production Releases
        - All JIRA Release Notes (version 3.x and later)
        - All JIRA Upgrade Guides (version 3.x and later)
        - JIRA 3.13 Release Notes
        - JIRA 3.12 Release Notes
        - JIRA 3.11 Release Notes
        - JIRA 3.10 Release Notes
        - JIRA 3.9 Release Notes
        - JIRA 3.8 Release Notes
        - JIRA 3.7 Release Notes
        - JIRA 3.6 Release Notes
        - JIRA 3.5 Release Notes
        - JIRA 3.4 and 3.4.1 Release Notes

        - JIRA 3.3 Release Notes
        - JIRA 3.2 Release Notes
        - JIRA 3.1 Release Notes
        - JIRA 3.0 Release Notes
        - JIRA 2.6 Release Notes
    - JIRA Release Summary
    - Beta Releases
        - JIRA 4.0 Beta 2 Release Notes
        - JIRA 4.0 Beta 1 Release Notes
    - EAP Releases
        - JIRA 3.11 EAP Release Notes
    - Security Advisories
        - JIRA Security Advisory 2007-12-24
        - JIRA Security Advisory 2008-02-21
        - JIRA Security Advisory 2008-08-26
        - JIRA Security Advisory 2008-10-29
        - JIRA Security Advisory 2008-12-09
        - JIRA Security Advisory 2009-04-02

JIRA Requirements

JIRA is a 'web application', meaning it runs centrally on a server, and users interact with it through web browsers from any computer.

- JIRA Client Requirements
    - 1. Browser
- JIRA Server Requirements
    - 1. Hardware
    - 2. Operating System
    - 3. Java
    - 4. Application Server
    - 5. Database

JIRA Client Requirements

1. Browser

The following browsers are recommended for use with JIRA 4:

- Internet Explorer 7 and 8 *
- Firefox 3.x
- Safari 4

** All of the main functionality will work in IE 6; however, some of the visual effects will be missing, as IE 6 is becoming quite dated and does not support web standards that are in common use today.*

To get the full experience JIRA has to offer, you should enable Javascript in your browser. If Javascript is disabled you will still be able to use all the main functionality of JIRA, but some features will not work (e.g. adding issue types per project; the preview for the wiki renderer).

JIRA Server Requirements

1. Hardware

During evaluation, JIRA will run well on any reasonably fast workstation computer (e.g. 1.5+ Ghz processor). Memory requirements depend on how many projects and issues you will store, but 256MB is enough for most evaluation purposes. (Note: most people start by installing JIRA Standalone on their local computer. It is easy to move JIRA to a production server later).

The hardware required to run JIRA in production depends mainly on the number of issues and users that your installation will have, as well as the maximum number of concurrent requests that the system will experience during peak hours.

- If you are planning to have a small number of projects (10-20) with 1,000 to 5,000 issues in total and about 100-200 users, a recent server (2.8+GHz CPU) with 256-512MB of available RAM should cater for your needs.
- If you are planning for a greater number of issues and users, adding more memory will help. We have reports that allocating 1GB of RAM to JIRA is sufficient for 100,000 issues.
- If your system will experience a large number of concurrent requests, running JIRA on a multi-CPU machine will increase the concurrency of processing the requests, and therefore speed up the response time.

For reference, jira.atlassian.com has over 33,000 issues and over 30,000 user accounts. The system runs on a 64bit Quad processor. The server has 4 GB of memory with 1 GB dedicated to JIRA.

Please note that performance heavily depends on your usage pattern. As a general rule, if you wish to store more than 200,000 issues in JIRA we recommend that you set up separate instances of JIRA on different physical machines and split your projects and issues between the instances. You can follow the instructions on splitting a JIRA instance, if you need to convert an existing JIRA instance into multiple instances.

We would appreciate it if you let us know what hardware configuration works for you. Please create a ticket in our support system with your hardware specification and mention the number of users and issues in your JIRA installation.

> JIRA requires access to a local disk for certain functionality (e.g. if JIRA does not have read and write access to disk, searching and attachments will not work).

> While some of our customers run JIRA on SPARC-based hardware, Atlassian only officially supports JIRA running on x86 hardware and 64-bit derivatives of x86 hardware.

2. Operating System

JIRA runs on any operating system that supports Java (see below). If you're undecided, see our thoughts on OS choice.

Please note:

- If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
  ```
  $ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
  ```
- If you are using IBM's **i5 OS:** JIRA Standalone (Tomcat) will not run due to startup script problems.

3. Java

JIRA requires the **Java Developers Kit (JDK) version 5 (1.5) or later** to be installed on the server. Please see Installing Java.

Currently Sun's JDK is available for Windows (32 bit + 64 bit), Solaris (SPARC + 32 bit + 64 bit) and Linux Platforms (32 bit + 64 bit). If you are a Mac OSX user, Apple also provides a specific JDK that has been optimised for its hardware and OS.

If you are running the Sun JRE version 6 (1.6), please ensure that you are running Update 10 or higher.

4. Application Server

JIRA is a web application, so requires an application server.

- **JIRA Standalone** ships pre-configured with the Apache Tomcat application server, which is a stable, light weight and fast performing server.
- **JIRA WAR/EAR** can be installed into any of the supported application servers (provided they are compatible with your chosen operating system and JDK). Note that manual configuration will be required.

5. Database

JIRA requires a relational database for storage of issue data. JIRA supports most relational database servers, so we suggest using the one that you are most comfortable administering.

- **JIRA Standalone** ships pre-configured with the HSQLDB database, which is suitable for evaluation purposes. HSQLDB is prone to database corruption. Before deploying to a production environment, we **strongly recommend** that you connect JIRA Standalone to an enterprise database (see the list of supported databases).
- **JIRA WAR/EAR** can be connected to any of the supported databases, provided they are compatible with your application server.

Next Step

Install JIRA Standalone (recommended); or Install JIRA WAR/EAR.

Installing Java

This page contains instructions for installing Java and setting JAVA_HOME.

> You can skip this page if you are using the Windows Installer.

1. Installing Java

JIRA requires a Java Development Kit (JDK) version 5 (1.5) or above to run. The JDK may be obtained from Sun's website (get the 'offline' edition if you're using Windows).

If you are running the Sun JRE version 6 (1.6), please ensure that you are running Update 10 or higher.

Linux note: Linux distributions frequently have an open-source implementation of Java called GCJ installed. Do not use this - it is incomplete and will cause JIRA to fail in obscure ways. You can test whether you have the correct Sun Java by running `java -version` :

```
~$ java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build 1.6.0-b105)
Java HotSpot(TM) Client VM (build 1.6.0-b105, mixed mode, sharing)
```

On recent Linux distributions, Sun's Java can be installed with a command like `sudo apt-get install sun-java6-jdk` (for Ubuntu).

Linux note: On recent X.org-based distros (eg. FC4+) to avoid getting errors like:

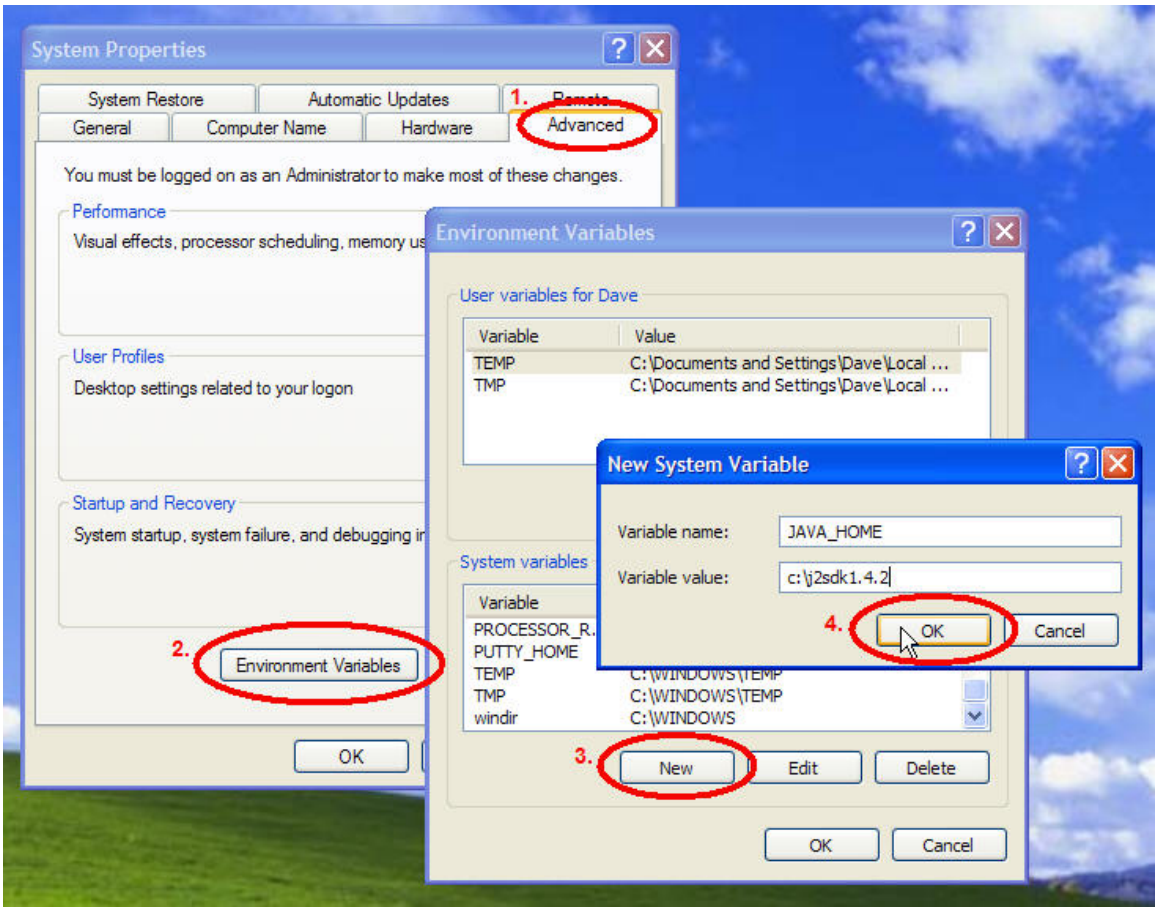
```
java.lang.UnsatisfiedLinkError: /opt/j2sdk1.4.2_11/jre/lib/i386/libawt.so: libXp.so.6: cannot open
        shared object file: No such file or directory
```

you will need to install the **xorg-x11-deprecated-libs** package (Fedora) or equivalent (check Google).

2. Setting JAVA_HOME

Once the JDK is installed, you will need to set the **JAVA_HOME** environment variable, pointing to the root directory of the JDK. Some JDK installers set this automatically (check by typing ' `echo %JAVA_HOME%` ' in a DOS prompt, or ' `echo $JAVA_HOME` ' in a shell). If it is unset, set it by hand in the Control Panel.

1. Right click on the **My Computer** icon on your desktop and select properties
2. Click the **Advanced** Tab
3. Click the **Environment Variables** button
4. Click **New**
5. Enter **JAVA_HOME** as the variable name and the directory where you installed Java.



6. Restart your computer.

3. Confirming that Java works

When the above steps have been done correctly, it should be possible to open a DOS prompt and type `%JAVA_HOME%\bin\javac` , and see this printed:

```
Usage: javac <options> <source files>
where possible options include:
...
```

If, later on when you try to start JIRA, you get the error **Windows cannot find '-Xms128m',** it is because you have not correctly set JAVA_HOME.

Next Step

Install JIRA Standalone

Installing JIRA Standalone

This Installation Guide applies if you are installing JIRA for the first time. If you are upgrading JIRA, please refer to the Upgrade Guide.

Which 'Distribution' Should You Choose?

JIRA is available in two 'distributions':

Standalone distribution	WAR/EAR distribution
Pre-packaged with Tomcat application server	Deploys into an existing application server

Pre-packaged with HSQLDB database, but can be easily connected to any supported database	Connects to any supported database
Requires virtually no setup	Requires manual configuration
Recommended for all users	Suitable only for system administrators

The **Standalone distribution is recommended,** even for organisations with an existing application server environment. Please note however, that the HSQLDB database pre-packaged with the Standalone distribution is suitable for evaluation purposes only.

To install JIRA Standalone, follow the instructions for your operating system:

- Installing JIRA Standalone on Windows
- Installing JIRA Standalone on Mac OS
- Installing JIRA Standalone on Unix or Linux

If you are installing JIRA 4.0 Beta:
The **only** plugin that is compatible with JIRA 4.0 Beta is the latest JIRA Toolkit. Do not install any other plugins.

Installing JIRA Standalone on Windows

This Installation Guide applies if you are installing JIRA for the first time. If you are upgrading JIRA, please refer to the Upgrade Guide.

To install JIRA Standalone on Windows, follow one of these methods:

- Using the Windows Installer (recommended)
  - 1. Download and Install JIRA Standalone
  - 2. Start JIRA
  - 3. Run the Setup Wizard
  - Next Steps
- Using the Windows Archive (ZIP File)
  - Before you begin
  - 1. Download and Install JIRA Standalone (ZIP file)
  - 2. Set JIRA Home (ZIP file)
  - 3. Start JIRA (ZIP file)
  - 4. Run the Setup Wizard (ZIP file)
  - Next Steps (ZIP file)

Using the Windows Installer (recommended)

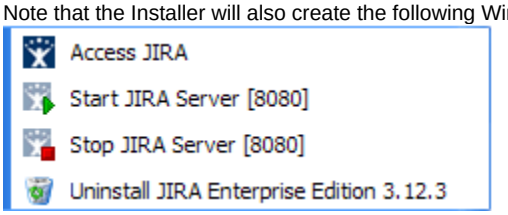
1. Download and Install JIRA Standalone

1. Download the JIRA Standalone **Windows Installer (.EXE)** file here.
2. Run the **.EXE** file.
3. You will be prompted for input (such as the directory in which to install JIRA and the JIRA Home directory). Note that installing the JIRA files in certain directories, for example the default "C:\Program Files", may require you to have local administrator rights on your PC.
4. If you have local administrator rights on your PC, you will be prompted to install JIRA as a *service* (recommended). You can also do this manually later, as described in Configuring JIRA Standalone to run as a Service.
5. You may be prompted to choose the *port* that JIRA runs on. (This will happen if you have already installed JIRA and it is running, or if you have some other existing network service running on JIRA's default ports.) If you are not sure which port number to choose, try new numbers one by one, starting with '8080' and incrementing by one. Keep going until you find an available port number.

2. Start JIRA

JIRA will start automatically when the Installer finishes, if you have selected the option to launch JIRA at the end of the Installer wizard.

Note that the Installer will also create the following Windows 'Start' menu shortcuts, which you can also use to start and stop JIRA:



- **Access JIRA** — this shortcut opens a web browser window to access your JIRA application (via http://localhost:8080). Please note that

your JIRA server must be running to access the JIRA application via your browser.
- **Start JIRA Server** — this shortcut starts the JIRA instance on your application server, so that you can access it with your browser.
- **Stop JIRA Server** — this shortcut stops the JIRA instance on your application server. You will not be able to access the JIRA application via your browser, if a JIRA instance is not running on your application server.

To access JIRA, go to your web browser and type this address: http://localhost:8080. Alternatively, you can use the 'Access JIRA' shortcut in the 'Start' menu.

If JIRA does not appear, you may need to change the port that JIRA runs on.

3. Run the Setup Wizard

See Running the Setup Wizard.

Next Steps

- We highly recommend that you configure JIRA Standalone to run as a service. If you do not do this, you will need to start the JIRA server manually (via the 'Start' menu shortcut or using the `bin\startup.bat`file) every time you restart your computer. NOTE: If you have Administrator rights to your PC, then the Windows Installer you may have already done this as part of the installation process
- By default, JIRA Standalone uses the standard Tomcat port (i.e. 8080). If you need another application to run on that port, either now or in the future, please see Changing JIRA Standalone's port.
- As part of its installation process, JIRA Standalone automatically installs, configures and connects itself to an HSQLDB database. This is fine for evaluation purposes, however HSQLDB is prone to data corruption. For production installations, we strongly recommend that you connect JIRA to an external database.

Using the Windows Archive (ZIP File)

Before you begin

Please ensure that you have installed Java and set JAVA_HOME — unless you are using the **Windows Installer,** which will do this for you automatically.

1. Download and Install JIRA Standalone (ZIP file)

1. Download the JIRA Standalone **Windows Archive (.ZIP)** file here.
2. Unzip the downloaded file.

   Avoid the Windows XP built-in unzip tool! The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

3. Extract all the files to **`'c:'`** or another **short path** to avoid Windows path size limitations.
4. If you are installing **JIRA 3.2.2 or earlier** and use **WinZIP version 9**, please manually create a `temp` directory in the root directory (alongside bin\, conf\ etc), as WinZIP doesn't extract this (resulting in index errors).

2. Set JIRA Home (ZIP file)

To specify the location of your JIRA Home Directory:

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

3. Start JIRA (ZIP file)

Run `bin\startup.bat` (or if that immediately exits, try `bin\catalina.bat run`). Note: assuming you unzipped to **`'c:'`,** the file is located here: **`c:\atlassian-jira-enterprise`**`-X.X.X-standalone\bin\startup.bat` (`'enterprise'` or `'professional'` or `'standard'`). JIRA will be launched in a black 'Tomcat' window (do not close this window). Wait until the following message appears:

```
************************************************
         You can now access JIRA through your web browser.
        ************************************************
```

To access JIRA, go to your web browser and type this address: http://localhost:8080.

If JIRA does not appear, you may need to change the port that JIRA runs on.

4. Run the Setup Wizard (ZIP file)

See Running the Setup Wizard.

Next Steps (ZIP file)

- We highly recommend that you configure JIRA Standalone to run as a service. If you do not do this, you will need to start the JIRA server manually (using the `bin\startup.bat` file) every time you restart your computer. Note that once you configure JIRA to run as a service, you will start and stop JIRA via the Services control panel in Windows (instead of the method described in "3. Starting JIRA", above), and there will be no black 'Tomcat' window.
- By default, JIRA Standalone uses the standard Tomcat port (i.e. 8080). If you need another application to run on that port, either now or in the future, please see Changing JIRA Standalone's port.
- As part of its installation process, JIRA Standalone automatically installs, configures and connects itself to an HSQLDB database. This is fine for evaluation purposes, however HSQLDB is prone to data corruption. For production installations, we strongly recommend that you connect JIRA to an external database.

Installing JIRA Standalone on Mac OS

To install JIRA Standalone on Mac OS, follow these steps:

- Before you begin
- 1. Download and Install JIRA Standalone
- 2. Set JIRA Home
- 3. Start JIRA
- 4. Run the Setup Wizard
- Next Steps

Before you begin

Please ensure that you have installed Java and set JAVA_HOME.

1. Download and Install JIRA Standalone

1. Download the JIRA Standalone **TAR (.GZ)** file here.
2. Unzip the downloaded file.

If something goes wrong, please verify that Java is installed correctly. If the problem persists, please contact us — we're happy to help.

2. Set JIRA Home

To specify the location of your JIRA Home Directory:

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

3. Start JIRA

Run `bin/startup.sh` to start JIRA.

JIRA will be launched in a black 'Tomcat' window (do not close this window). Wait until the following message appears:

```
************************************************
          You can now access JIRA through your web browser.
         ************************************************
```

To access JIRA, go to your web browser and type this address: http://localhost:8080.

If JIRA does not appear, you may need to change the port that JIRA runs on.

4. Run the Setup Wizard

See Running the Setup Wizard.

Next Steps

- By default, JIRA Standalone uses the standard Tomcat port (i.e. 8080). If you need another application to run on that port, either now or in the future, please see Changing JIRA Standalone's port.
- As part of its installation process, JIRA Standalone automatically installs, configures and connects itself to an HSQLDB database. This is fine for evaluation purposes, however HSQLDB is prone to data corruption. For production installations, we strongly recommend that you connect JIRA to an external database.

Installing JIRA Standalone on Unix or Linux

To install JIRA Standalone on UNIX/Linux, follow these steps:

- Before you begin
- 1. Download and Install JIRA Standalone
- 2. Set JIRA Home
- 3. Start JIRA
- 4. Run the Setup Wizard
- Next Steps

Before you begin

Please ensure that you have installed Java and set JAVA_HOME.

If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
`$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira`

1. Download and Install JIRA Standalone

1. Download the JIRA Standalone TAR (.GZ) file here.
2. Unzip the downloaded file.

   Avoid the Solaris default tar utility! On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

If something goes wrong, please verify that Java is installed correctly. If the problem persists, please contact us — we're happy to help.

2. Set JIRA Home

To specify the location of your JIRA Home Directory:

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired

location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

3. Start JIRA

Run `bin/startup.sh`. Wait until the following message appears:

```
************************************************
         You can now access JIRA through your web browser.
        ************************************************
```

To access JIRA, go to your web browser and type this address: http://localhost:8080.

If JIRA does not appear, you may need to change the port that JIRA runs on.

Note: Logs will be written to `logs/catalina.out`.

4. Run the Setup Wizard

See Running the Setup Wizard.

Next Steps

- By default, JIRA Standalone uses the standard Tomcat port (i.e. 8080). If you need another application to run on that port, either now or in the future, please see Changing JIRA Standalone's port.
- As part of its installation process, JIRA Standalone automatically installs, configures and connects itself to an HSQLDB database. This is fine for evaluation purposes, however HSQLDB is prone to data corruption. For production installations, we strongly recommend that you connect JIRA to an external database.

Configuring JIRA Standalone

The pages listed below contain information on configuring JIRA Standalone:

- Running JIRA over SSL or HTTPS
- Integrating JIRA with a Web Server
- Running JIRA Standalone as a Service
- Installing Confluence into JIRA Standalone
- Changing JIRA Standalone's port

Running JIRA over SSL or HTTPS

When web applications are being accessed across the internet, there is always the possibility of usernames and passwords being intercepted by intermediaries (e.g. between your computer and the ISP/company). It is often a good idea to enable access via HTTPS (HTTP over SSL), and require its use for pages where passwords are sent. Note, however, that using HTTPS may result in slower performance.
In some cases where issue data is sensitive, all pages should be accessed via HTTPS.

The process of enabling SSL access is specific to each app server, but specifying which pages to require protection for is generic. Here we describe the process for Tomcat, bundled with JIRA Standalone.

JIRA 3.5.3 has a bug which causes a popup warning to appear for IE users when using SSL. This has been fixed in 3.6 and above

On this page:

- Enable SSL access
  - Tomcat (JIRA Standalone)

    - Importing from a CA issued Certificate
- Requiring HTTPS for certain pages
- SSL + Apache + IE problems
- Troubleshooting
    - Can't find the keystore
    - Incorrect password
    - Passwords don't match
    - Wrong certificate

Enable SSL access

Tomcat (JIRA Standalone)

Edit `conf/server.xml`, and at the bottom before the </Service> tag, add this section (or uncomment it where you find it) in Tomcat 5.5 (JIRA Standalone >= 3.3):

```
<Connector port="8443" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true" useBodyEncodingForURI="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
```

or the following if using Tomcat 4.1.x (JIRA Standalone <= 3.2.3):

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
port="8443" minProcessors="5" maxProcessors="75"
enableLookups="true"
acceptCount="100" debug="0" scheme="https" secure="true"
useURIValidationHack="false" disableUploadTimeout="true">
<Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
clientAuth="false" protocol="TLS" />
</Connector>
```

This enables SSL access on port 8443 (the default for https is 443, but just as Tomcat uses 8080 instead of 80 to avoid conflicts, 8443 is used instead of 443 here).

Now create a SSL key for Tomcat to send to connecting clients. You can create a self-signed key for testing purposes with one of the following commands:

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA (Windows)
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA  (Unix)
```

The keytool utility will prompt you for two passwords: the keystore password and the key password for Tomcat. You must use the same value for both passwords, and the value must be either:

1. "changeit" (this is the default value Tomcat expects), or
2. if you use a value other than "changeit", you must also specify it in `conf/server.xml`. You must add the following attribute to the Connector tag described above:

    ```
    keystorePass="<password value>"
    ```

To add a CA-issued key pair, see the Tomcat documentation.

IE7 on Vista Issue
If your clients will access JIRA from Internet Explorer 7 on Vista, please ensure that you specify the **-keyalg RSA** flag. By default the SHA1 algorithm is used, which results in 'Internet Explorer cannot display the webpage' errors on IE7 on Vista. Apparently on JDK 1.6 you also need to specify the **-sigalg MD5withRSA** flag since **-keyalg RSA** will still result in SHA1 being used (see this blog for more information).

Once Tomcat is restarted, you should be able to access JIRA on **https**://localhost:**8443**/. Try this before continuing.

Importing from a CA issued Certificate

When using CA certificates, you also need import the certificate using the `keytool` command, rather than generating a self-signed key. Something like the command below.

```
keytool -import -alias tomcat -file certificate.cer -keystore some/path/to/file -storepass
something.secure
```

The `-file` is your certificate and the `-keystore` is an optional destination, but it'll guarantee that you know where your keystore is. By default, the keystore is placed in your user home directory. You can get more information on the keytool at the keytool documentation.

You'd then need to edit the `server.xml` as per the Tomcat Docs "Edit the Tomcat Configuration File" section. Basically, you'll need to add the `keystoreFile` and `keystorePass` to the SSL Connector definition to match your keystore settings.

Requiring HTTPS for certain pages

Although HTTPS is now activated and available, the old HTTP URLs (http://localhost:8080) are still available. In most situations one wants these URLs to continue working, but for some to redirect to their https equivalent. This is done by editing `WEB-INF/web.xml`, and adding the following section at the end of the file, before the closing `</web-app>`:

```
<security-constraint>
 <web-resource-collection>
  <web-resource-name>all-except-attachments</web-resource-name>
  <url-pattern>*.js</url-pattern>
  <url-pattern>*.jsp</url-pattern>
  <url-pattern>*.jspa</url-pattern>
  <url-pattern>*.css</url-pattern>
  <url-pattern>/browse/*</url-pattern>
 </web-resource-collection>
 <user-data-constraint>
  <transport-guarantee>CONFIDENTIAL</transport-guarantee>
 </user-data-constraint>
</security-constraint>
```

This means that all URLs *except attachments* are redirected from HTTP to HTTPS. IE has a bug which prevents attachments like .doc files being viewed via HTTPS if SSL protection is forced in `web.xml`.

Once this change is made, restart JIRA and access http://localhost:8080. You should be redirected to https://localhost:8443/secure/Dashboard.jspa. The port it redirects to is determined by the `redirectPort` value you specify in the `server.xml` file in the HTTP Connector stanza.

(!)There does not seem to be an easy way to make subsequent pages revert to HTTP after logging in via HTTPS - see JRA-7250

SSL + Apache + IE problems

Some people have reported errors when uploading attachments over SSL using IE. This is due to an IE bug, and can be fixed in Apache by setting:

```
BrowserMatch ".MSIE." \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
```

Google has plenty more on this.

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by keytool, as described above.

When you enter "https://localhost:8443" in your browser, if you get a message such as "Cannot establish a connection to the server at localhost:8443", look for error messages in your logs/catalina.out log file. Here are some possible errors with explanations:

Can't find the keystore

```
java.io.FileNotFoundException: /home/user/.keystore (No such file or directory)
```

This indicates that Tomcat cannot find the keystore. The keytool utility creates the keystore as a file called `.keystore` in the current user's home directory. For Unix/Linux the home directory is likely to be `/home/<username>`. For Windows it is likely to be `C:\Documents And Settings\<UserName>`.

Make sure you are run JIRA as the same user who created the keystore. If this is not the case, or if you are running JIRA on Windows as a service, you will need to specify where the keystore file is in `conf/server.xml`. Add the following attribute to the connector tag you uncommented:

```
keystoreFile="<location of keystore file>"
```

Incorrect password

```
java.io.IOException: Keystore was tampered with, or password was incorrect
```

You used a different password than "changeit". You must either use "changeit" for both the keystore password and for the key password for Tomcat, or if you want to use a different password, you must specify it using the `keystorePass` attribute of the Connector tag, as described above.

Passwords don't match

```
java.io.IOException: Cannot recover key
```

You specified a different value for the keystore password and the key password for Tomcat. Both passwords must be the same.

Wrong certificate

```
javax.net.ssl.SSLException: No available certificate corresponds to the SSL cipher suites which are
enabled.
```

If the Keystore has more than one certificate, Tomcat will use the first returned unless otherwise specified in the SSL Connector in `conf/server.xml`.

Add the `keyAlias` attribute to the Connector tag you uncommented, with the relevant alias, for example:

```
<Connector port="8443" maxHttpHeaderSize="8192"
               maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
               enableLookups="false" disableUploadTimeout="true" useBodyEncodingForURI="true"
               acceptCount="100" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS"
               keystoreFile="/opt/local/.keystore"
               keystorePass="removed"
               keyAlias="tomcat"/>
```

Integrating JIRA with a Web Server

The following pages contain information on integrating JIRA with a web server.

- Integrating JIRA with IIS
- Integrating JIRA with Apache

Integrating JIRA with IIS

This page describes how to configure Microsoft's IIS web server and JIRA such that IIS forwards requests on to JIRA, and responses back to the user. This is useful if you already have IIS running serving web pages (e.g. http://mycompany.com), and wish to integrate JIRA as just another URL (e.g. http://mycompany.com/jira).

JIRA is written in Java, and needs a Java Application Server (servlet container) to run. As IIS does not provide services of a Java Application Server, it is not possible to deploy JIRA directly into IIS. It is possible, however, to configure IIS to proxy requests for JIRA to an application server where JIRA is deployed. Therefore, if your main website is running in IIS, it is possible to integrate JIRA into this website.

If you need to integrate JIRA with IIS, JIRA needs to be deployed into a Java Application Server that provides IIS integration capability. Apache Tomcat is one of these application servers. The Standalone distribution of JIRA ships with Apache Tomcat, so if you are using the Standalone distribution or have deployed JIRA WAR into your own Tomcat instance, please follow this document.

If you are not using the Standalone distribution or Apache Tomcat, please consult your application server's documentation to determine whether it is possible to integrate with IIS and how to achieve it.

To integrate JIRA Standalone with IIS you will need to:

1. Configure JIRA and test that it works on its own
2. Configure Tomcat to accept proxied requests from IIS
3. Configure IIS to forward JIRA requests to Tomcat
4. *(Optional)* Configure IIS to forward Confluence requests to Tomcat (if you are using both Confluence and JIRA).

1. Configure JIRA

1. Follow the JIRA Standalone install guide to install and configure the Standalone distribution of JIRA; or deploy the EAR-WAR distribution into Apache Tomcat. Note that JIRA can be installed on the same machine as IIS, but this is not necessary.
2. Change the context path of the JIRA web application:
   To allow IIS to proxy requests to JIRA, JIRA web application must be deployed with a context path (e.g. the **/jira** in http://localhost:8080*/jira*) in Tomcat. The context path **must** be set to the path in the URL that IIS will use to proxy requests. For example, if your website is running with address `www.example.com` in IIS, and you would like to make JIRA available under `www.example.com/jira`, you will need to set JIRA's context path to "`/jira`" in Tomcat.
   To do this, edit the `conf/server.xml` file (if you are using JIRA Standalone) or the `jira.xml` file (if you are using the EAR-WAR distribution of JIRA). Change the `path` attribute of the `Context` element to "`/jira`". For example, in JIRA Standalone 3.3 and later the `Context` element would look like:

   ```
   <Context path="/jira" docBase="${catalina.home}/atlassian-jira"
              swallowOutput="true" reloadable="false">
   ```

3. Restart JIRA after changing the context path.
4. Turn JIRA's GZip compression **OFF** (since there will be no benefit from GZip compression once proxying is implemented).
5. Test that JIRA works correctly by pointing your web browser directly at Tomcat (e.g. http://localhost:8080/jira) and going through JIRA's Setup Wizard. If you have completed the Setup Wizard previously, try creating an issue or editing one. Please ensure that no errors occur.

2. Configure Tomcat to accept proxied requests

1. Enable **AJP/1.3 Connector** in Tomcat: To allow Tomcat to accept requests for JIRA from IIS, edit the `conf/server.xml` file and ensure that the **AJP/1.3 Connector** is enabled (i.e. *not* commented out).To enable the AJP/1.3 Connector in JIRA Standalone, Tomcat 5.5.x or Tomcat 5.0.x, remove the comment symbols (**<!-** ~~and~~ **->**) around the following section in the `conf/server.xml` file:

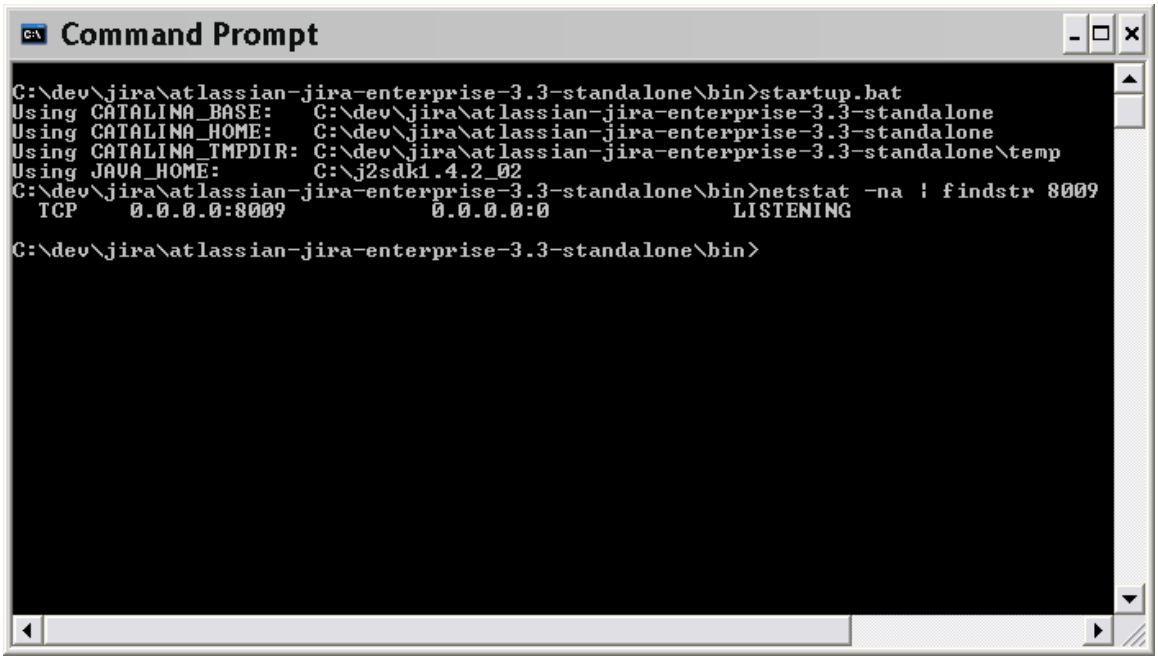
   ```
   <Connector port="8009"
              enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
   ```

   If you are using JIRA Standalone 3.2 or earlier or running JIRA in Tomcat 4.1.x the AJP/1.3 Connector definition in the `conf/server.xml` file looks like:

   ```
   <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
              port="8009" minProcessors="5" maxProcessors="75"
              enableLookups="true" redirectPort="8443"
              acceptCount="10" debug="0" connectionTimeout="0"
              useURIValidationHack="false"
              protocolHandlerClassName="org.apache.jk.server.JkCoyoteHandler"/>
   ```

   The above example configures Tomcat to listen for proxied IIS requests on port `8009`. If this port is already in use on the machine where JIRA is running, please change to another port.

2. Restart Tomcat and ensure that no errors regarding used ports appear in the logs or in the Tomcat Console.
3. Ensure that the AJP Connector is listening on the specified port (8009 by default). One way to do this is to use the "`netstat -na`" command in the command window and see if port 8009 is listed in the output:



3. Configure IIS to forward requests to JIRA

On the machine where IIS is deployed:

1. Download the ISAPI Redirect DLL from the apache site. When downloading, choose the version of Windows that IIS is running on (either win32 or win64), and then choose the latest available jk version.

   > The file to download is named **isapi_redirect_X.X.X.dll,** where 'X.X.X' is the version number. You will need to remove the version number from the DLL file (i.e. it needs to be named isapi_redirect.dll).

2. Place the DLL and the associated properties files in an installation directory. For the purpose of this document, we will assume the directory is `C:\tomcat_iis_connector`. Place the **isapi_redirect.dll** in this directory. Then download the isapi_redirect.properties file and place this in the same directory as the **isapi_redirect.dll** file.
3. Create a directory called 'conf' in your installation directory (`C:\tomcat_iis_connector\conf`). Download the files uriworkermap.properties and workers.properties.minimal and place them in the `C:\tomcat_iis_connector\conf` directory.
4. Create a directory called 'logs' (`C:\tomcat_iis_connector\logs`). This is where the logs associated with the **isapi_redirect.dll** execution will be placed.
5. In the "`C:\tomcat_iis_connector`" directory you may need to modify the `isapi_redirect.properties` file. The `isapi_redirect.properties` file tells the connector where to find its configuration files and where the DLL can be found in relation to the IIS server. There are 5 properties in this file:
   a. extension_uri — the path to the virtual directory that contains the **isapi_redirect.dll**
   b. log_file — the path to write the log file to
   c. log_level — the level at which the logs should be generated
   d. worker_file — the path to your `workers.properties.minimal` file in your installation
   e. worker_mount_file — the path to your `uriworkermap.propertiesl` file in your installation.
   If you are installing the connector in `C:\tomcat_iis_connector` and you follow the instructions below about setting up the virtual directory for the **isapi_redirect.dll,** then you should not have to change any properties in the provided file.
6. In the "`C:\tomcat_iis_connector\conf`" directory you may need to modify the `uriworkermap.properties` and the `workers.properties.minimal` files.

   > The provided files contain the changes mentioned here and should work if you completely follow this document. **If you have deviated from this document, then you will need to modify these files as described below.**

   The `workers.properties.minimal` file tells IIS where (IP address and port) Tomcat is running. The `uriworkermap.properties` tells IIS what requests to proxy to Tomcat.
   To edit these files:

   a. Edit the `uriworkermap.properties` and ensure that it contains the following mapping for JIRA. You do not need any other mappings.

```
/jira/*=worker1
```

The mapping (e.g. `/jira/`**) *must** be the same as the context path that JIRA has been deployed with in Tomcat as described in the Configure JIRA section of this document.

b. Edit the `workers.properties.minimal` file and modify the `worker.ajp13w.host` property if necessary. This property should be set to the host name or the IP address of the machine where Tomcat (with JIRA) is running. If Tomcat is running on the same machine as IIS then you can leave the property set to `localhost`. If you have specified a host name as the value of this property, please ensure that the IIS machine can correctly resolve it to the appropriate IP address.
c. If you have modified the port for the AJP Connector you will need to modify the `worker.ajp13w.port` property. Here is an example of the file with Tomcat running on the same machine as IIS and using the default port (8009) for AJP:

```
worker.list=worker1

#
# Defining a worker named worker1 and of type ajp13.
# Note that the name and the type do not have to match.
#
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```

7. Open **Control Panel,** then **Administrative Tools** and open **Internet Information Services.**
8. **IIS 7.0 only:** If you are using **IIS 7.0,** you will need to install two required service roles, ISAPI Extensions and ISAPI Filters:
    a. Navigate to Start Menu > All Programs > Administration Tools > Service Manager.
    b. Select 'Web Server (IIS)' in Server Manager > Roles.
    c. Click 'Add Role Services' and follow the Wizard.
9. Add an **ISAPI Filter** to IIS, as described below:
    - **IIS 6.0 or earlier:**
        a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and click on **Properties.**
        b. Click the **ISAPI Filters** tab.
        c. Check if there is a Filter that points to the `isapi_redirect.dll` file and that it is in the right location. If not, click **Add** and create one. Enter `tomcat` as the Filter Name and enter the location of the `isapi_redirect.dll` file for the executable.
        d. Click **Apply** and then **OK.**
    - **IIS 7.0:**
        a. Click the **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and click on **ISAPI Filters.**
        b. Click the **ISAPI Filters** icon.
        c. Check if there is a Filter that points to the `isapi_redirect.dll` file and that it is in the right location. If not, click **Add** and create one. Enter `tomcat` as the Filter Name and enter the location of the `isapi_redirect.dll` file.
        d. Click **OK.**
10. Create a **virtual directory** for JIRA in IIS.
    a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), choose **New** and then **Virtual Directory.**
    b. Go through the creation wizard. Set the `alias` as the value of the Context Path (without slashes) that was set in the Configure JIRA section of this document (see above). In our example this is `jira` .
    c. This can point to any directory.
    d. Complete the wizard.

The reason for creating a virtual directory is so that requests without the trailing slash still work. For example, if you are deploying JIRA under `http://www.example.com/jira/` without the virtual directory, then requests to `http://www.example.com/jira` will fail.

11. Create a **virtual directory** for access to the **isapi_redirect.dll** in IIS, as described below:
    - **IIS 6.0 or earlier:**
        a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), choose **New** and then **Virtual Directory.**
        b. Go through the creation wizard. Set the `alias` to be `jakarta` .
        c. This must point to the directory in which the **isapi_redirect.dll** is installed. In our example this is `C:\tomcat_iis_connector`.
        d. Complete the wizard, making sure that you grant the 'Execute' permission for the **Virtual Directory** by checking the 'Execute' checkbox.

- **IIS 7.0:**
    a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and choose **Add Virtual Directory.**
    b. Set the `alias` to be `jakarta` .
    c. **Physical Path** must point to the directory in which the **isapi_redirect.dll** is installed. In our example this is `C:\tomcat_iis_connector`.
    d. Click the 'jakarta' Virtual Directory and double-click 'Handler Mappings'.
    e. Click 'Edit Feature Permissions' in the Action panel on the right-hand side.
    f. Check the 'Execute' permission checkbox.

This Virtual Directory is needed for the connector to work. The alias that you give the directory needs to be the same as the path set in the `isapi_redirect.properties` file, `extension_uri` property. In our example this value is: `/jakarta/isapi_redirect.dll`.

12. If using IIS 6.0 or 7.0, you will need to add the dll as a **Web Service Extension,** as described below.
    - **IIS 6.0:**
        a. Right-click on **Web Service Extensions** and choose **Add a new Web Service Extension...**
        b. Enter `tomcat` for the **Extension Name** and then add the `isapi_redirect.dll` file to the required files.
        c. Select the **Set extension status to Allowed** check-box, then click **OK.**
    - **IIS 7.0:**
        a. Navigate to the servers and highlight your server.
        b. Navigate to 'ISAPI and CGI Restrictions'.
        c. Add and allow the isapi_redirect.dll extension.
13. You will need to restart the IIS Service. To do this, browse to **Control Panel,** click **Administrative Tools,** click on **Services,** find the IIS Admin Service and click **restart.**
14. You are done! To test the configuration, point your web browser at IIS and append JIRA's context path to the URL. For example, if your website is running under the address of `http://www.example.com` and you have deployed JIRA with the context path of `jira`, point your browser at `http://www.example.com/jira`.

4. Configure IIS to forward requests to Confluence as well as JIRA

You can configure IIS so that it forwards requests to both JIRA and Confluence.

The following instructions describe how to forward from IIS to separate instances of JIRA and Confluence, running in separate Tomcat servers. The instructions assume that you have already set up IIS to forward to JIRA as described in section 3 above. The instructions also assume that you have already installed Confluence as per the Confluence Installation Guide.

The instructions describe how to make JIRA available under `www.example.com/jira` as described above, and Confluence available under `www.example.com/confluence`.

1. If JIRA and Confluence are running on the same machine, ensure that Confluence is listening on a different port to JIRA:Edit the `conf/server.xml` file (if you are using Confluence Standalone) or the `jira.xml` file (if you are using the EAR-WAR distribution of Confluence). At the top of the file, change the `port` attribute of the `Server` element to a different port to the value for JIRA. For example, change it from `8005` to `8006`.
   Still in the `Server` element, Change the `port` attribute of the `Connector` sub-element to a different port to the value for JIRA. For example, change it from `8080` to `8090` .
2. Change the Confluence context path:Edit the `conf/server.xml` file (if you are using Confluence Standalone) or the `jira.xml` file (if you are using the EAR-WAR distribution of Confluence). Change the `path` attribute of the `Context` element to "`/confluence`".
3. Restart Confluence after changing the ports and the context path, and test that Confluence works correctly by pointing your web browser at http://localhost:8090/confluence.
4. Configure Confluence to accept proxied requests:Remove the comments around the AJP/1.3 Connector section in the Confluence `conf/server.xml` or `jira.xml` file and change the `port` attribute to a value different to the value for JIRA. For example, change it from `8009` to `8010`.
5. Restart Confluence and ensure that no errors regarding used ports appear in the logs or in the Tomcat console.
6. Edit the `uriworkermap.properties` file and add the following mapping:

```
/confluence/*=worker2
```

   The file should now contain the following mappings:

```
/jira/*=worker1
                    /confluence/*=worker2
```

7. Edit the `workers.properties.minimal` file:
   Change the line starting with `worker.list` to the following:

```
worker.list=worker1,worker2
```

Add the following lines to the end of the file (assuming the host is on the same machine as IIS and you changed the AJP/1.3 Connector port for Confluence to `8010`):

```
worker.worker2.type=ajp13
                    worker.worker2.host=localhost
                    worker.worker2.port=8010
```

The `workers.properties.minimal` file should now look like the following:

```
worker.list=worker1,worker2

#
# Defining a worker named worker1 and of type ajp13.
# Note that the name and the type do not have to match.
#
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009

worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8010
```

8. Create a **virtual directory** for Confluence in IIS. Set the `alias` to `confluence`. It can point to any directory.
9. Restart the IIS Service.
10. You are done! Confluence should now be available under `www.example.com/confluence`, and JIRA should still be available under `www.example.com/jira`.

Troubleshooting

- ***Whenever I go to JIRA in my browser, a login panel pops up. I enter a valid username and password for JIRA, but the panel pops up again.***Make sure that you have Anonymous Access set on the `jira` virtual directory in IIS. It will be set to that if you have followed the above instructions.To check this:
    1. In **'Internet Information Services',** right click the `jira` virtual directory and choose **'Properties'.**
    2. Click the **'Directory Security'** tab.
    3. Click the **'Edit...'** button in the **'Anonymous access and authentication control'** section.
    4. Make sure that the **'Anonymous access'** tick box is selected, and make sure that nothing is selected in the **'Authenticated access'** section. Do not select **'Basic authentication'.** Do not select **'Integrated Windows authentication'.**

- ***Whenever I go to JIRA in Internet Explorer, a login panel pops up. I enter a valid username and password for JIRA, but the panel pops up again. This doesn't happen, however, in another browser such as Firefox or Safari. I can successfully log in to JIRA in those browsers.***Make sure that you have Internet Explorer's **User Authentication** set to **Anonymous login.** To check this:
    1. In Internet Explorer, click the **'Tools'** menu and select **'Internet Options'.**
    2. Click the **'Security'** tab.
    3. Select the security zone that the JIRA server is in.
    4. Click the **'Custom level...'** button.
    5. Scroll right down to the bottom to the **'User Authentication'** section.
    6. Select **'Anonymous logon'** (if it is not already selected).
    7. Click the **'OK'** button on this screen, and again on the next screen.
    8. Restart Internet Explorer.

- ***When I try to navigate to my JIRA instance at http://localhost/jira in my browser, it prompts me to download a file with nonsensical information, rather than showing me my JIRA instance.***Make sure that you have granted the 'Execute' permission to your Virtual Directory for JIRA in IIS. See step 11 of the '3. Configure IIS to forward requests to JIRA' section in this document for detailed instructions.

Integrating JIRA with Apache

This page describes how to integrate an Apache web server with JIRA (via mod_proxy), such that Apache forwards requests on to JIRA, and responses back to the user. This is useful if you already have Apache serving web pages on port 80 (e.g. http://mycompany.com), and wish to

integrate JIRA as just another URL (eg. http://mycompany.com/jira).

This documentation describes a straightforward implementation of mod_proxy. If you require a more complex solution, refer to Apache HTTP Server Version Documentation and, if necessary, consult with someone in your organisation who is knowledgeable in the configuration of Apache.

- Step 1: Configure JIRA's application server
- Step 2: Configure Apache
  - Enable mod_proxy and mod_proxy_http
  - Configure mod_proxy
- Configuring SSL
- Troubleshooting
- Further Information

Step 1: Configure JIRA's application server

Here we assume you are using the JIRA Standalone distribution, which comes with Tomcat 5.5. First, we need to edit Tomcat's **conf/server.xml** file, and set the context path:

```
<Server port="8005" shutdown="SHUTDOWN">

        <Context path="/jira" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
             ....
```

Here we have set the context path to **/jira,** assuming JIRA will be running on http://mycompany.com/jira/.

Restart Tomcat, and ensure you can still access JIRA normally (eg. at `http://localhost:8080/jira/`).

Note: if you want Tomcat responsible for all URLs, specify a blank context path with `path=""` -- *not* `path="/"` .

Turn JIRA's GZip compression **OFF** (since there will be no benefit from GZip compression once proxying is implemented).

Step 2: Configure Apache

Assuming an Apache 2 installation, the following needs to be done:

Enable mod_proxy and mod_proxy_http

The exact steps will be specific to your operating system. Refer to the Apache documentation for your operating system. On Debian/Ubuntu it is done as follows:

```
teacup:/etc/apache2# a2enmod proxy_http
Enabling proxy as a dependency
Module proxy installed; run /etc/init.d/apache2 force-reload to enable.
Module proxy_http installed; run /etc/init.d/apache2 force-reload to enable.
teacup:/etc/apache2#
```

Configure mod_proxy

Here we create a config snippet for JIRA, in **sites-available/jira-mod_proxy:**

```
teacup:/etc/apache2# cd sites-available

teacup:/etc/apache2/sites-available# cat > jira-mod_proxy

<Proxy *>
Order deny,allow
Allow from all
</Proxy>

ProxyRequests       Off
ProxyPreserveHost On
ProxyPass           /jira       http://localhost:8080/jira
ProxyPassReverse    /jira       http://localhost:8080/jira

teacup:/etc/apache2/sites-available#
teacup:/etc/apache2/sites-available# a2ensite jira-mod_proxy
Site jira-mod_proxy installed; run /etc/init.d/apache2 reload to enable.

teacup:/etc/apache2/sites-available# /etc/init.d/apache2 reload
Reloading apache 2.0 configuration....
teacup:/etc/apache2/sites-available#
```

JIRA should now be integrated with Apache. You should be able to view JIRA at `http://localhost/jira` (i.e. on port 80).

Notes:

- The path '/jira' **must** be the same as the context path in Tomcat's conf/server.xml
- The **ProxyPreserveHost** directive allows Tomcat to know its public hostname and port. Without this, JIRA would redirect the public URL (e.g. `http://mycompany.com/jira/`) to `http://localhost:8080/jira/secure/Dashboard.jspa` .

If the links for Printable Version, RSS feeds, Word export and Excel export have incorrect URLs, starting with `localhost:8080/jira` instead of `http://mycompany.com/jira` , ensure that **ProxyPreserveHost** is set to **On.**

ProxyPreserveHost is only available on Apache 2. For Apache 1.1-1.3.x, you should instead specify **proxyName** and **proxyPort** attributes in Tomcat as follows:

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true"
      proxyName="mycompany.com" proxyPort="80" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="/jira" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            ....
```

If you are using Apache 1.x, make sure you **don't** use caching (CacheRoot directive).

Some users have reported problems with user sessions being hijacked when the mod_cache module is enabled. If you have such problems, disable the **mod_cache** module. Note that this module is enabled by default in some Apache 2 distributions.

Configuring SSL

If you want to use https (e.g. `https://mycompany.com/jira/`), then:

- Configure Tomcat to serve HTTP traffic on port 8443. There are instructions here. The end result should be that `https://localhost:8443/jira/` works.
- In the Apache config (`/etc/apache2/sites-available/jira-mod_proxy`), ensure you have **SSLProxyEngine on** specified, and redirect `/jira` to `https://localhost:8443/jira`:

```
<Proxy *>
Order deny,allow
Allow from all
</Proxy>

SSLProxyEngine on
ProxyRequests       Off
ProxyPreserveHost On
ProxyPass           /jira       https://localhost:8443/jira
ProxyPassReverse    /jira       https://localhost:8443/jira
```

- Please ensure that the ProxyPass and ProxyPassReverse directives do not include a trailing '/'. There have been reports that this may cause problems in JIRA 3.7 and above when serving static resources (javascript and css).

Troubleshooting

- On **Fedora Core 4,** people have reported 'permission denied' errors when trying to get mod_proxy (and mod_jk) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.

- If you are on Macintosh OS X, please disable **webperfcache,** which proxies port 80 by default. A user reported this as the likely cause of JIRA session problems, in the form of users' identities becoming mixed up:

```
The OSX Servers enable webperfcache by default for Virtual Hosts, which for static content
        would be great, but for dynamic sites (which ALL of ours are) it is Evil and causes
many
        issues.  Of note recently was the jira session issue. Also see :-
         http:
//developer.apple.com/documentation/Darwin/Reference/ManPages/man8/webperfcache.8.html
http://www.wodeveloper.com/omniLists/webobjects-admin/2005/January/msg00009.html
Unfortunately even if you disable webperfcache for a site, if there is a single site enabled
        then all sites will still proxy through webperfcache with resulting session problems.
```

- In general, if you are having problems:
    1. Ensure that JIRA works as expected when running directly from Tomcat on http://localhost:8080/jira
    2. Watch the log files (usually in /var/log/httpd/ or /var/log/apache2/). Check that you have a **LogLevel** directive in your httpd.conf, and turn up logging **('LogLevel debug')** to get more info.
    3. Check out the Knowledge Base .

Further Information

For more advanced mod_webapp configurations (eg. SSL), see this mod_proxy guide.

Running JIRA Standalone as a Service

For long-term use, JIRA should be configured to automatically restart when the operating system restarts. For Windows servers, this means configuring JIRA to run as a **Windows service.**

Running JIRA as a Windows service has other advantages. When started manually a console window opens, and there is a risk of someone accidentally shutting down JIRA by closing this. Also, the JIRA logs are properly managed by the Windows service (reliably found in `logs\stdout*.log` in your JIRA directory, and rotated daily).

There are two ways to install JIRA as a service: via the installer, and manually.

On this page:

- Installing JIRA as a Service
    - Installing as a Service with the Installer
    - Installing as a Service manually
- Removing the JIRA service
- Changing the Windows user that the JIRA service uses
- Specifying the startup order of multiple services
- Locating the name of a service
- Troubleshooting

Installing JIRA as a Service

Installing as a Service with the Installer

The easiest way to get JIRA installed as a Windows service is by clicking the '**Install JIRA as Service**' checkbox when running the Windows installer:

You will need Administrator rights to your PC for this to work.

Installing as a Service manually

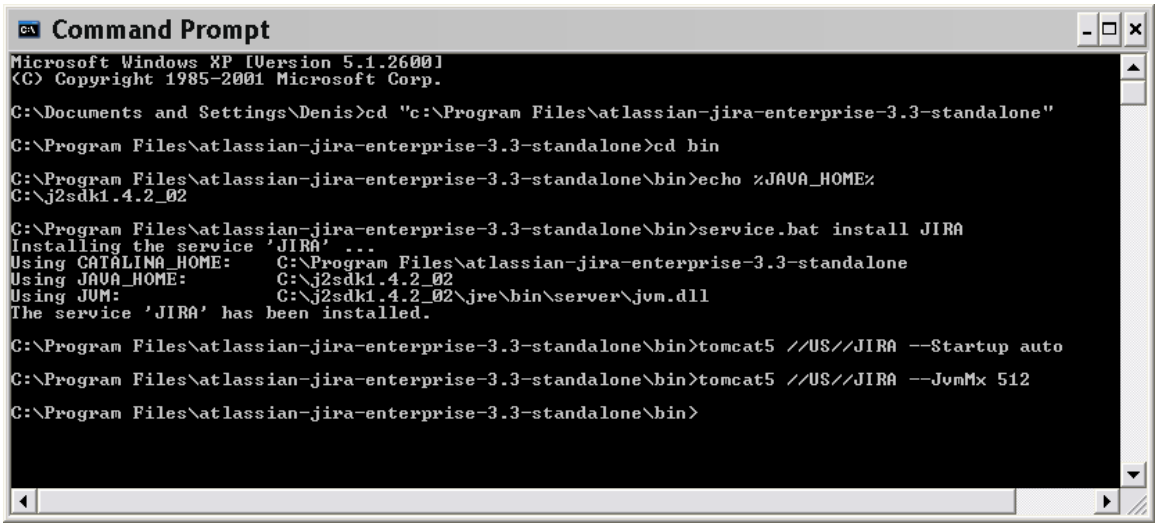
If you didn't use the Windows installer, or for some other reason have JIRA Standalone and now want it to run as a service, follow these instructions. They apply only to JIRA running in the Apache Tomcat application server (e.g. the JIRA "Standalone" distribution).

> If you are running 64-bit Windows, please note that Apache Tomcat cannot run as a Windows service if you are using a 64 bit JDK (see JRA-12965). Please ensure that you are using a 32 bit JDK (if you used the Windows Installer, a 32 bit JDK has already been installed for you).

1. Open a DOS prompt,
2. 'cd' to the JIRA Standalone directory, and then the *bin* subdirectory
3. If a directory in the path has spaces (e.g. 'C:\Program Files\..'), please convert it to its eight-character equivalent (e.g. c:\Progra~1\..).
4. Ensure the **JAVA_HOME** variable is set to the JDKbase directory, with `echo %JAVA_HOME%`.
5. Run the following command:

```
service.bat install JIRA
```

Here is a screenshot of the process:



JIRA should now be installed as a service.

6. In addition, to have the JIRA service start automatically when the server starts, run:

```
tomcat5 //US//JIRA --Startup auto
```

The JIRA service will automatically start up the next time the server reboots. The JIRA service can be manually started with the command **'net start JIRA'** and stopped with **'net stop JIRA'.**

To see what parameters the JIRA service is starting with, Go to Start -> Run, and run "regedt32.exe". There should be an entry at HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Procrun 2.0 -> JIRA.

7. Additional JIRA Standalone setup options (optional):
   - To increase the maximum memory JIRA can use (the default will already be 256MB), run:

     ```
     tomcat5 //US//service_name --JvmMx 512
     ```

     where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

   - To *add* JVM parameter, for example pass a parameter to enable JIRA's Jelly support, run:

     ```
     tomcat5 //US//service_name ++JvmOptions="-Djira.jelly.on=true"
     ```

     where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

   - If you are running JIRA and Confluence in the same JVM, increase the MaxPermSize size to 128 MB:

     ```
     tomcat5 //US//service_name ++JvmOptions="-XX:MaxPermSize=128m"
     ```

     where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

   - Occasionally, it may be useful to view JIRA's Garbage Collection information. This is especially true when investigating memory issues.To turn on the Verbose GC (garbage collection) logging, execute the following command in the command prompt:

     ```
     tomcat5 //US//service_name ++JvmOptions="-Xloggc:path\to\logs\atlassian-gc.log"
     ```

     where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.
     The path (denoted by **\path\to)** refers to the directory in which JIRA is currently installed. For example:

     ```
     tomcat5 //US//service_name ++JvmOptions="-Xloggc:c:\jira\logs\atlassian-gc.log"
     ```

     where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

See the Tomcat documentation for further service options.

Removing the JIRA service

If JIRA was installed through the Windows installer, go to the '**Control Panel**' in Windows, click '**Add or Remove Programs**' and remove JIRA. This will remove the service too.

If you installed the service manually (see above) it can be uninstalled with:

```
service.bat remove JIRA
```

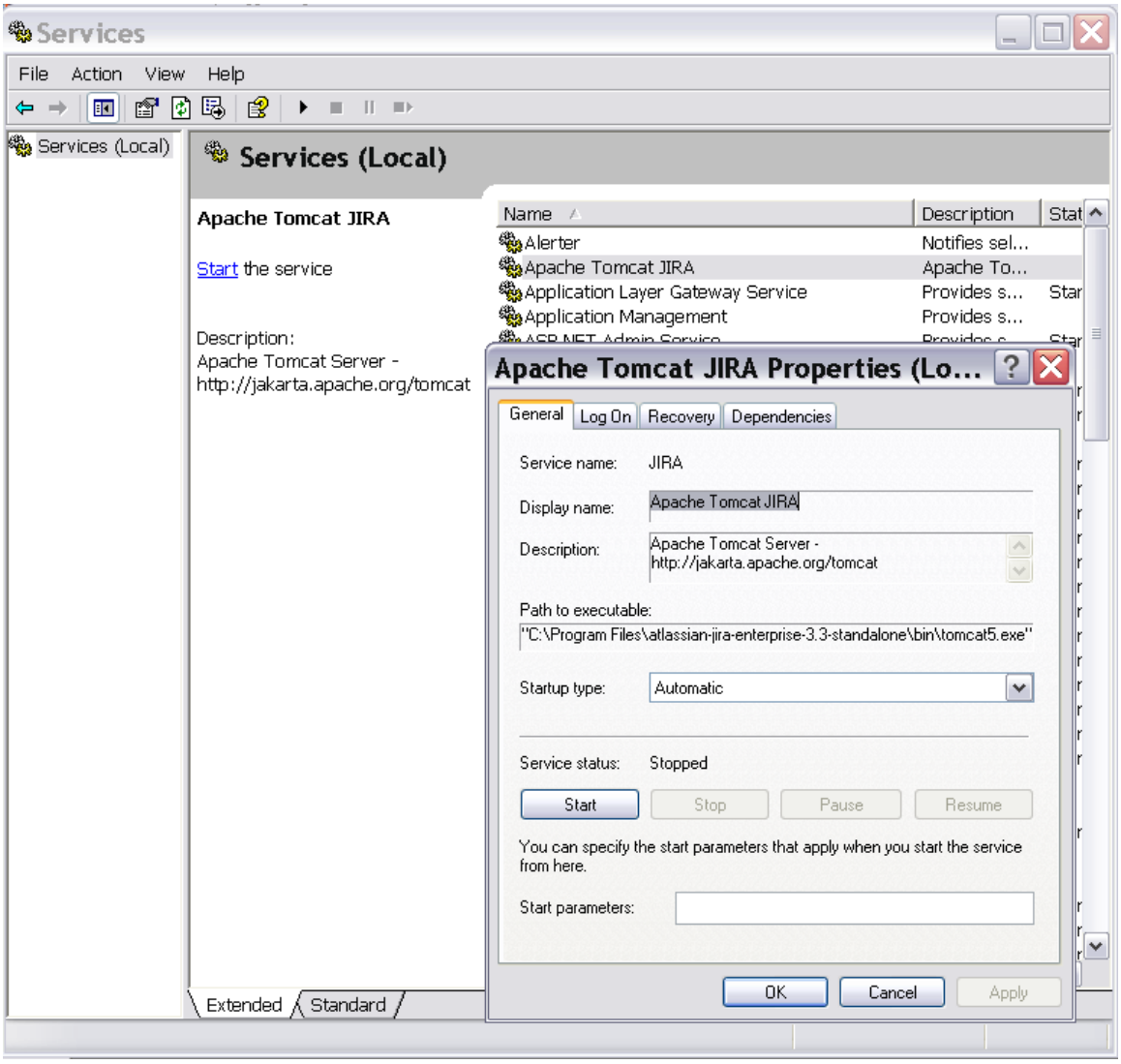
Alternatively, if the above does not work, use `tomcat5 //DS//JIRA`.

Changing the Windows user that the JIRA service uses

If you are using mapped network drives for JIRA's backup directory, attachments directory, index directoy or the %CATALINA_HOME%* directory, you need to ensure that JIRA can write to these drives. That is, these directories all need to be writeable by the user which the JIRA service is running as. This may mean that you need to change the Windows user that the JIRA server uses.

Also note that you must also specify these network drives by UNC and not letter mappings, e.g. **\\backupserver\jira** not **z:\jira**

To change the Windows user that the JIRA service uses, navigate to the service in Windows, i.e. **'Control Panel' -> 'Administrative Tools' -> 'Services'**. Locate the Apache Tomcat service, right-click and view the 'Preferences'.



Go to the '**Log On**' tab and change the user as desired.

Specifying the startup order of multiple services

If you have services that depend on each other, it is important that they are started in the correct order. Common examples include:

- If you are running both JIRA and Crowd, it is important to start Crowd first, so that Crowd is running before people try to login to JIRA.
- If the database JIRA connects to is hosted on the same server as JIRA, and is started via a Windows service, the JIRA service will only start successfully if the database service has already started first.

To set up start up dependency rules, open a command prompt and enter the following command:
`C:\Documents and Settings\Developer>sc config [JIRA service] depend=[database service]`
Please note the space character after `'depend='`.

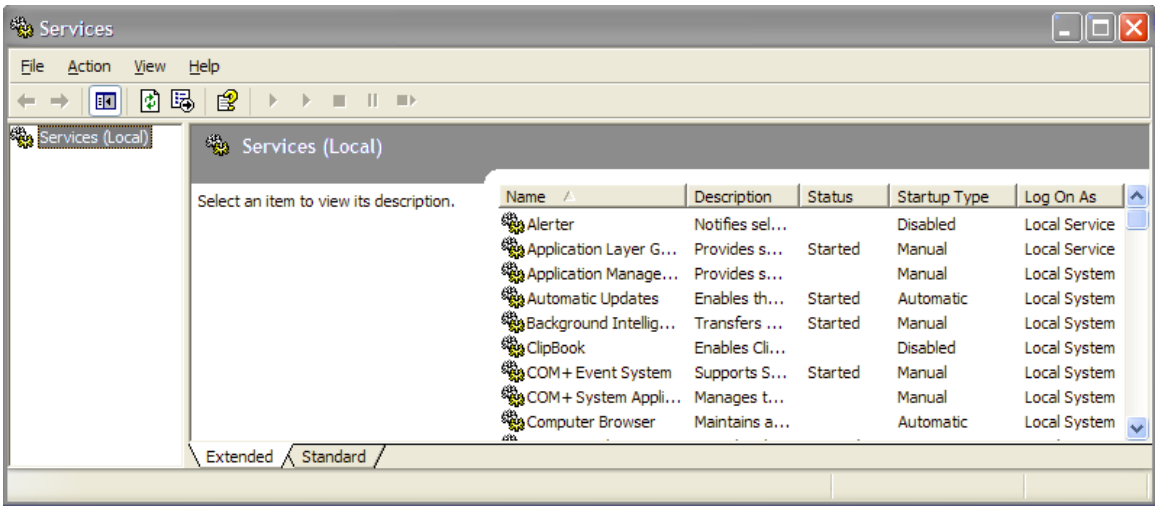
- **[JIRA service]** is the name of the JIRA service you are running, e.g. JIRA051007111904.
- **[database service]** is the name of the database service you are running, e.g. MSSQLSERVER.

If you wish, you can also set up dependency rules by editing the system registry. Please see http://support.microsoft.com/kb/193888 for details on how to do this.

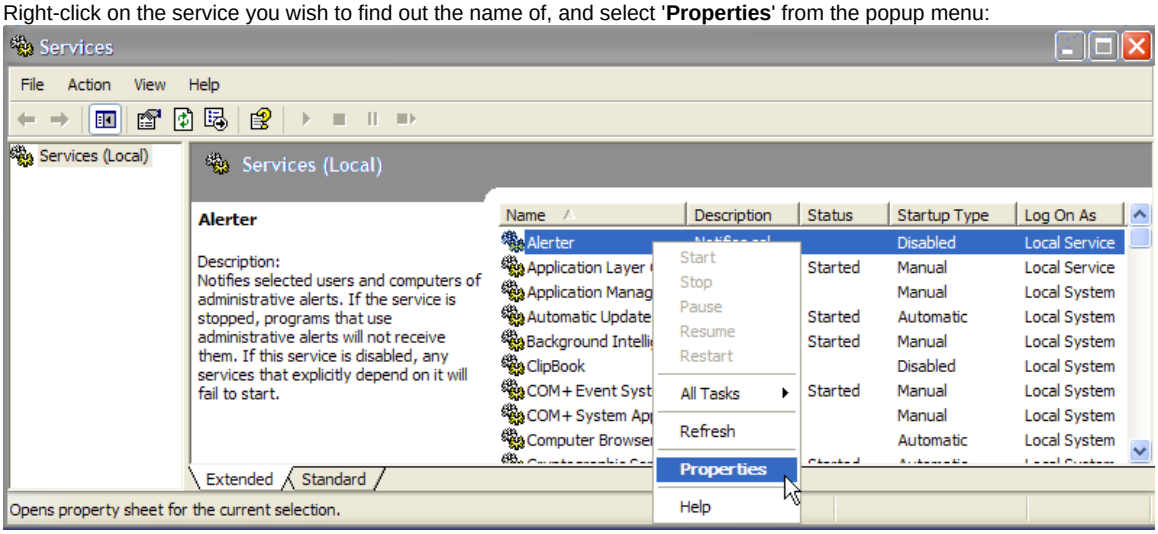
Locating the name of a service

If you do not know the exact name of your JIRA service or your database service, you can find out what they are by following the steps below:

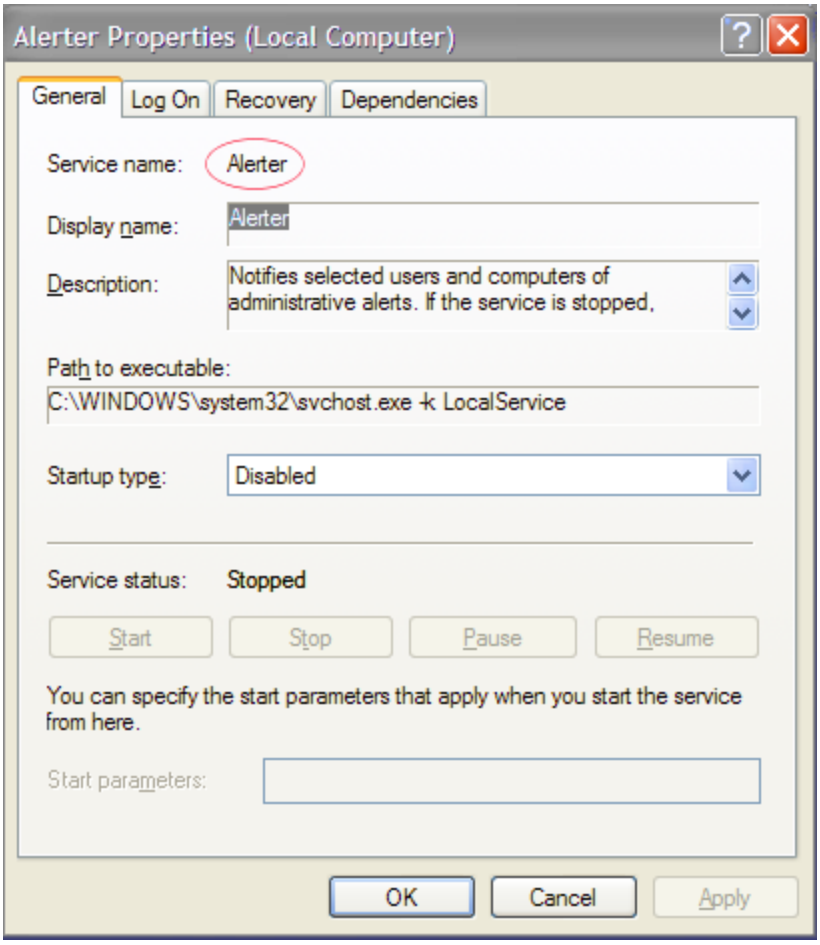
1. Navigate to **'Control Panel' -> 'Administrative Tools' -> 'Services'**.
2. The 'Services' window should appear:



3. Right-click on the service you wish to find out the name of, and select '**Properties**' from the popup menu:



4. The 'Service name' should appear in the 'General' tab:



Troubleshooting

Problems may occur when trying to setup JIRA Standalone to run as a Windows service with JDK 1.6. The problem is due to failure to locate "MSVCR71.DLL", which can be found in `%JAVA_HOME%/bin`. There are two options to resolve this problem:

- Add %JAVA_HOME/bin to PATH, then restart the JIRA server.
- Copy MSVCR71.DLL to system path, C:\WINDOWS\SYSTEM32 or C:\WINNT\SYSTEM32

Installing Confluence into JIRA Standalone

It is possible to install a version of Confluence into a version of JIRA standalone. For more information please see this online document.

You may also wish to read Integrating JIRA and Confluence for helpful information on integrating JIRA and Confluence.

Changing JIRA Standalone's port

By default, JIRA Standalone runs on port **8080** (and hence is available at (http://`<yourserver>`:8080, eg. http://localhost:8080).

If the server on which you run JIRA Standalone already has a service claiming port **8080**, there will be a conflict, and JIRA will fail to start. You may see errors like this:

```
LifecycleException:  Protocol handler initialization failed: java.net.BindException: Address already
in use:8080
```

This can be fixed by changing JIRA to use another port (eg. 8090). This is done by editing `conf\server.xml` (eg. in Wordpad). The start of the file looks like:

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" />
```

Here, change "**8005**" to "**8006**", and change "**8080**" to "**8090**" (or some other free port — see below).

Then restart JIRA (bin\shutdown.bat; bin\startup.bat) and point a browser to http://<yourserver>:**8090**

If you are running on a Unix server and choose port 80 (or any port below 1024), you will **need to start JIRA as root** in order to bind to the port

Which port number should I choose?

If you are not sure which port number to choose, try new numbers one by one, starting with default port numbers and incrementing by one. Keep going until you find an available port number.

A note about firewalls

When you choose a port number for JIRA, bear in mind that your firewall may prevent people from connecting to JIRA based on the port number. Organisations with a local network protected by a firewall typically need to consider modifying their firewall configuration whenever they install a web-based application (such as JIRA) that is running on a new port or host. Even personal laptop and desktop machines often come with firewall software installed that necessitates the same sort of change as described above.

If JIRA does not need to be accessed from outside the firewall, then no firewall configuration changes will be necessary.

Note that Confluence Standalone also runs on port 8080 by default. If you're looking to change the port of Confluence Standalone, see Changing listen port for Confluence Standalone.

Installing JIRA WAR-EAR

This Installation Guide applies if you are installing JIRA for the first time. If you are upgrading JIRA, please refer to the Upgrade Guide.

Which 'Distribution' Should You Choose?

JIRA is available in two 'distributions':

Standalone distribution	WAR/EAR distribution
Pre-packaged with Tomcat application server	Deploys into an existing application server
Pre-packaged with HSQLDB database, but can be easily connected to any supported database	Connects to any supported database
Requires virtually no setup	Requires manual configuration
Recommended for all users	Suitable only for system administrators

The **Standalone distribution is recommended,** even for organisations with an existing application server environment. Please note however, that the HSQLDB database pre-packaged with the Standalone distribution is suitable for evaluation purposes only.

Installing JIRA WAR/EAR

The distribution ZIP file contains an Ant build script to create a deployable EAR or WAR for your particular application server (and any necessary library directories).

The basic set of steps to install JIRA WAR/EAR is as follows:

1. Download and unzip JIRA (but not with XP's unzipper nor the default tar utility on Solaris). Ensure that you download the **WAR/EAR version,** not the Standalone version that is recommended on the Downloads page.

   > **Avoid the Windows XP built-in unzip tool!** The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

   > **Avoid the Solaris default tar utility!** On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

   A new directory containing JIRA will be created, hereafter referred to as `$JIRA_INSTALL`.

   > If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
   > `$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira`

2. Follow the instructions in the **readme.txt** file, located in the root directory of the unpacked JIRA distribution. This refers to the server-specific installation guides:
   - Tomcat 6.0.x
   - Tomcat 5.5.x
   - Weblogic 8.1 and 9.x
   - Websphere 6
3. Set your JIRA home directory as described in Setting your JIRA Home Directory.
4. The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

   ```
   mail.mime.decodeparameters=true
   ```

   System properties are set in different ways depending on your application server.
5. Access JIRA using your web browser and run through the brief Setup Wizard.
6. For production use, please check that your application server is allocated sufficient memory. Java applications claim up to only 64Mb by default, and this usually isn't enough for JIRA. See Increasing JIRA Memory for more information.

Next step

Please see Application Server Guides for JIRA WAR-EAR .

Installing JIRA on Tomcat 6.0

This section describes how to install JIRA on Tomcat 6.0, a popular open-source server from the Apache Jakarta project.

Tomcat can be downloaded from the Apache site.

> **Warning**
> All versions of Tomcat effectively leak memory by caching JSPs, which can result in OutOfMemoryErrors if large pages (eg. RSS or Excel) are requested. There is a flag you should set to disable this caching.

> **Warning**
> If you are using version 6.0.13 or higher of Tomcat 6, with a MySQL database, you must set up Tomcat to survive connection closures. These versions of Tomcat have been noted to exhibit problems maintaining connections to MySQL databases. Please read this document for details on the changes required.

Note
The JIRA 'Standalone' download is JIRA preconfigured with a copy of Tomcat 5.5.26. If you have JIRA Standalone, you don't need to follow the steps below.

On this page:

- 1. Unpack JIRA
- 2. Configure JIRA
  - 2.1 Database Connection
  - 2.2 JIRA Home
- 3. Build JIRA
- 4. Update Tomcat Libraries
- 5. Configure Tomcat
- 6. Modify Tomcat server.xml
- 7. Fix Tomcat memory settings
- 8. Set mail.mime.decodeparameters
- 9. Start Tomcat
- Troubleshooting
- User-contributed notes

1. Unpack JIRA

Download and unzip JIRA (but not with XP's unzipper nor the default tar utility on Solaris). Ensure that you download the **WAR/EAR version,** not the Standalone version that is recommended on the Downloads page.

Avoid the Windows XP built-in unzip tool! The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

Avoid the Solaris default tar utility! On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

A new directory containing JIRA will be created, hereafter referred to as `$JIRA_INSTALL`.

If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
```
$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

2. Configure JIRA

2.1 Database Connection

JIRA needs to be told what type of database you'll be using. The database is specified in `$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml` . Locate the **<datasource>** tag near the bottom, and change the **field-type-name** attribute value:

```
<datasource name="defaultDS"
          field-type-name="hsql"
          schema-name="PUBLIC"
          helper-class="org.ofbiz.core.entity.GenericHelperDAO"
          check-on-start="true"
          use-foreign-keys="false"
          use-foreign-key-indices="false"
          check-fks-on-start="false"
          check-fk-indices-on-start="false"
          add-missing-on-start="true">
          <jndi-jdbc jndi-server-name="default"
          jndi-name="java:comp/env/jdbc/JiraDS" />
        </datasource>
```

Possible values include **cloudscape, db2, firebird, hsql, mckoidb, mysql, mssql, oracle, postgres72, sapdb,** and **sybase**

For Postgres 7.3+ and DB2 you also need to set a **schema-name** attribute (see the PostgreSQL and DB2 pages).

Also in `entityengine.xml` , ensure the `<transaction-factory>...</transaction-factory>` tag contains:

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
        <user-transaction-jndi jndi-server-name="default" jndi-name="java:comp/env/UserTransaction"/>
      <transaction-manager-jndi jndi-server-name="default" jndi-name="java:comp/env/UserTransaction"/>
    </transaction-factory>
```

More details on JIRA's database access layer are available on the EntityEngine configuration page.

2.2 JIRA Home

To specify the location of your JIRA Home Directory (note that you need to do this *before* you build JIRA):

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

3. Build JIRA

Now build JIRA by typing **build** (Windows) or **./build.sh** (Unix) on the command line in the $JIRA_INSTALL directory. This will produce the deployable WAR file in the **$JIRA_INSTALL/dist-tomcat/tomcat-6** directory.

4. Update Tomcat Libraries

Tomcat does not come with some libraries required to run JIRA. To fix this, download jira-jars-tomcat6.zip (1.5Mb), and copy the contained jars to Tomcat's `lib/` directory.

5. Configure Tomcat

A JIRA 'context' now needs to be set up in Tomcat. To do this:

1. Make a `Catalina/localhost/` directory in Tomcat's `conf/` directory.
2. Copy **dist-tomcat/tomcat-6/jira.xml** from the built JIRA distribution to your Tomcat's **conf/Catalina/localhost/** directory.
3. Customise the copied **jira.xml** as follows:

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.13.war">
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="sa"
            password=""
            driverClassName="org.hsqldb.jdbcDriver"
            url="jdbc:hsqldb:path/to/database/jiradb/"
            minEvictableIdleTimeMillis="4000"
            timeBetweenEvictionRunsMillis="5000"/>

  <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
    factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
  <Manager pathname=""/>

</Context>
```

The paths (denoted as **path/to/)** will be correct by default, assuming you want to deploy the .war from the **dist-tomcat/tomcat-6** directory.

> **Note**
> If you are *not* using hsqldb, make sure you comment out the `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` params, or JIRA will run slower than normal.

If you are installing in Windows, make sure that the paths you specify for the location of the WAR file and database are full paths with drive letters (e.g. c:\yourdb\tomcatdb). **N.B.** the last part of the path is the name of the database and is not a directory.The above example assumes you are using hsql (an in-memory database - a good choice for evaluation purposes). Here is an example using MySQL:

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.13.war">
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
    username="jirauser"
    password="mypassword"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/jiradb?autoReconnect=true&amp;useUnicode=true
&amp;characterEncoding=UTF8"
maxActive="20"
    validationQuery="select 1"/>
  <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
    factory="org.objectweb.jotm.UserTransactionFactory"
    jotm.timeout="60"/>
  <Manager pathname=""/>
</Context>
```

Notice the lack of `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` parameters — those should **only** be used with hsql.

4. If using a different database than hsql,
   - copy the JDBC driver jar to `CATALINA_HOME/lib/` (see the database configuration guide); and
   - ensure you have updated the **field-type-name** (see above).

6. Modify Tomcat server.xml

In order for JIRA to correctly display internationalised characters in user and group names you need to modify your Tomcat distributions `conf/server.xml` file. You need to set the property `useBodyEncodingForURI="true"` within the connector definition for your http protocol. The connector block should look very much like this:

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443"/>
```

You should modify the block to contain the addition of the `useBodyEncodingForURI` property:

```
<Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" useBodyEncodingForURI="true"/>
```

Note
Because you must define this property in at the connector level this setting will effect all web-applications you have deployed under the connector. This should not adversely effect the other web-applications but please be aware of this. JIRA will run fine without this property set but you will run into issues if a user or group is created which contains international characters. It is best to set this property to true.

7. Fix Tomcat memory settings

Tomcat has a memory leak where large JSP page requests can fill up memory. To avoid this, edit Tomcat's `bin/setenv.sh` (create it if it does not exist) and set:

```
export CATALINA_OPTS="$CATALINA_OPTS -Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true"
```

or when installed as a Windows service, run:

```
tomcat5 //US//JIRA ++JvmOptions="-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true
```

For other environments, and for more info on memory settings, see the memory settings page.

8. Set mail.mime.decodeparameters

The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

```
mail.mime.decodeparameters=true
```

System properties are set in different ways depending on your application server.

9. Start Tomcat

JIRA should now be ready to run in Tomcat. To start using JIRA, first start (or restart) the Tomcat server with Tomcat's `bin/startup.(sh|bat)` scripts, and point your browser to http://localhost:8080/jira

You should now see the Setup Wizard, which will take you through the brief setup procedure.

Troubleshooting

It is easy to make a mistake in this process, and even more so if you are trying to connect to a database other than hsqldb. First, check that you have followed the process described above:

- If you are using an external database (not hsqldb), have you set the **field-type-name** attribute in **$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml** ? (step 2)
- Have you previously started JIRA with an incorrect **field-type-name** value? If so, the database schema would have been created incorrectly.
- If you have made changes to **$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml** (step 2) and re-run the build script (step 3), but your changes are not being picked up, delete the Tomcat `webapps/jira` directory, then restart JIRA. It would seem that in some circumstances Tomcat does not correctly re-expand the web application.
- Have you copied the extra Tomcat jars (step 4)? Check if you have `CATALINA_HOME/lib/objectweb-datasource-1.4.3.jar` present.
- If using an external database, did you copy the JDBC driver jar to `CATALINA_HOME/lib/` (step 5)?
- Is the path to the .war file in **conf/Catalina/localhost/jira.xml** correct?
- Have you copied the .war file to Tomcat's **webapps/** directory? This is almost guaranteed to cause problems - please move it elsewhere, and delete any JIRA subdirectories created in **webapps/** from previous Tomcat starts.
- Have you configured JIRA centrally in **conf/server.xml** instead of in **conf/Catalina/localhost/jira.xml** ? This is fine, but then be sure you *don't* also have a **conf/Catalina/localhost/jira.xml** present.
- The log files are usually vital to debugging problems. On Windows, these will appear in the console window that loads when running

`startup.bat` , or in one of the log files in the `logs/` directory. On Linux/Unix, logs will appear in a log file in `logs/` , usually `logs/*` (not just logs/catalina.out!). Check the log file for errors after startup.

- If you experience high memory usage / memory leaks (eg OutOfMemoryError), you may wish to set the system property `-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true` in **setenv.sh** / **setenv.bat.** This property is only valid for Tomcat 5.5.15 and later. For more information please see JRA-10145.
- If you receive exceptions related to logging, please ensure that **commons-logging-1.0.4.jar** and **log4j-1.2.7.jar** are present in Tomcat's `/lib/` directory. Also ensure that these files are not present in Tomcat's `webapps/jira/WEB-INF/lib` directory.
- If the connection to your MySQL database is dropping, you will need to set up Tomcat to survive connection closures.
  If you're stuck, please raise a support request, and attach your logs, configuration files, plus anything else relevant, and we'll get back to you as soon as possible. If you have a general question, please try the jira-user mailing list (which Atlassian staff monitor).

User-contributed notes

Do you have experiences to share with Tomcat 6.0.x and JIRA? We welcome your thoughts. Please see the user-contributed Tomcat 6.0.x notes.

Installing JIRA on Tomcat 5.5

This section describes how to install JIRA on Tomcat 5.5, a popular open-source server from the Apache Jakarta project.

Tomcat can be downloaded the Apache site.

Warning
Please use Tomcat 5.5.15 or higher. All versions of Tomcat effectively leak memory by caching JSPs, which can result in OutOfMemoryErrors if large pages (eg. RSS or Excel) are requested. In 5.5.15+ there is a flag you should set to disable this caching.

Warning
If you are using version 5.5.25 or higher of Tomcat 5, with a MySQL database, you must set up Tomcat to survive connection closures. These versions of Tomcat have been noted to exhibit problems maintaining connections to MySQL databases. Please read this document for details on the changes required.

Note
The JIRA 'Standalone' download is JIRA preconfigured with a copy of Tomcat 5.5.26. If you have JIRA Standalone, you don't need to follow the steps below.

On this page:

- 1. Unpack JIRA
- 2. Configure JIRA
  - 2.1 Database Connection
  - 2.2 Set JIRA Home
- 3. Build JIRA
- 4. Update Tomcat Libraries
- 5. Configure Tomcat
- 6. Modify Tomcat server.xml
- 7. Fix Tomcat memory settings
- 8. Set mail.mime.decodeparameters
- 9. Start Tomcat
- Troubleshooting
- User-contributed notes

1. Unpack JIRA

Download and unzip JIRA (but not with XP's unzipper nor the default tar utility on Solaris). Ensure that you download the **WAR/EAR version,** not the Standalone version that is recommended on the Downloads page.

Avoid the Windows XP built-in unzip tool! The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

Avoid the Solaris default tar utility! On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

A new directory containing JIRA will be created, hereafter referred to as `$JIRA_INSTALL`.

If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
`$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira`

{}

2. Configure JIRA

2.1 Database Connection

JIRA needs to be told what type of database you'll be using. The database is specified in `$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml` . Locate the **<datasource>** tag near the bottom, and change the **field-type-name** attribute value:

```
<datasource name="defaultDS"
          field-type-name="hsql"
          schema-name="PUBLIC"
          helper-class="org.ofbiz.core.entity.GenericHelperDAO"
          check-on-start="true"
          use-foreign-keys="false"
          use-foreign-key-indices="false"
          check-fks-on-start="false"
          check-fk-indices-on-start="false"
          add-missing-on-start="true">
          <jndi-jdbc jndi-server-name="default"
          jndi-name="java:comp/env/jdbc/JiraDS" />
        </datasource>
```

Possible values include **cloudscape, db2, firebird, hsql, mckoidb, mysql, mssql, oracle, postgres, postgres72, sapdb,** and **sybase**

For PostgreSQL 7.3+ and DB2 you also need to set a **schema-name** attribute (see the PostgreSQL and DB2 pages).

Also in `entityengine.xml` , ensure the `<transaction-factory>...</transaction-factory>` tag contains:

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
        <user-transaction-jndi jndi-server-name="default" jndi-name="java:comp/env/UserTransaction"/>
      <transaction-manager-jndi jndi-server-name="default" jndi-name="java:comp/env/UserTransaction"/>
    </transaction-factory>
```

More details on JIRA's database access layer are available on the EntityEngine configuration page.

2.2 Set JIRA Home

To specify the location of your JIRA Home Directory (note that you need to do this *before* you build JIRA):

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

3. Build JIRA

Now build JIRA by typing **build** (Windows) or **./build.sh** (Unix) on the command line in the $JIRA_INSTALL directory. This will produce the deployable WAR file in the **$JIRA_INSTALL/dist-tomcat** directory.

4. Update Tomcat Libraries

Tomcat does not come with some libraries required to run JIRA. To fix this, download jira-jars-tomcat5.zip (1.2Mb), and copy the contained jars to Tomcat's `common/lib/` directory.

5. Configure Tomcat

A JIRA 'context' now needs to be set up in Tomcat. To do this:

1. Copy **dist-tomcat/tomcat-5.5/jira.xml** from the built JIRA distribution to your Tomcat's **conf/Catalina/localhost/** directory.
2. Customise the copied **jira.xml** as follows:

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.13.war">
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="sa"
            password=""
            driverClassName="org.hsqldb.jdbcDriver"
            url="jdbc:hsqldb:path/to/database/jiradb/"
            minEvictableIdleTimeMillis="4000"
            timeBetweenEvictionRunsMillis="5000"/>

  <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
    factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
  <Manager pathname=""/>
</Context>
```

The paths (denoted as **path/to/)** will be correct by default, assuming you want to deploy the .war from the **dist-tomcat/** directory.

If you are *not* using hsqldb, make sure you comment out the `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` params, or JIRA will run slower than normal.

If you are installing in Windows, make sure that the paths you specify for the location of the WAR file and database are full paths with drive letters (e.g. c:\yourdb\tomcatdb). **N.B.** the last part of the path is the name of the database and is not a directory.The above example assumes you are using hsql (an in-memory database — a good choice for evaluation purposes). Here is an example using MySQL:

```
<Context path="/jira" docBase="path/to/atlassian-jira-3.13.war">
  <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
    username="jirauser"
    password="mypassword"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/jiradb?autoReconnect=true&amp;useUnicode=true
&amp;characterEncoding=UTF8"
maxActive="20"
    validationQuery="select 1"/>
  <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
    factory="org.objectweb.jotm.UserTransactionFactory"
    jotm.timeout="60"/>
  <Manager pathname=""/>
</Context>
```

Notice the lack of `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` parameters — those should **only** be used with hsql.

3. If using a different database than hsql,
   - copy the JDBC driver jar to `common/lib/` (see the database configuration guide); and
   - ensure you have updated the **field-type-name** (see above).

6. Modify Tomcat server.xml

In order for JIRA to correctly display internationalised characters in user and group names you need to modify your Tomcat distributions

`conf/server.xml` file. You need to set the property `useBodyEncodingForURI="true"` within the connector definition for your http protocol. The connector block should look very much like this:

```
<Connector port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true"/>
```

You should modify the block to contain the addition of the `useBodyEncodingForURI` property:

```
<Connector port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true" useBodyEncodingForURI="true"/>
```

Note
Because you must define this property in at the connector level this setting will effect all web-applications you have deployed under the connector. This should not adversely effect the other web-applications but please be aware of this. JIRA will run fine without this property set but you will run into issues if a user or group is created which contains international characters. It is best to set this property to true.

7. Fix Tomcat memory settings

This only works for Tomcat 5.5.15 and higher!

Tomcat has a memory leak where large JSP page requests can fill up memory. To avoid this, edit Tomcat's `bin/setenv.sh` (create it if it does not exist) and set:

```
export CATALINA_OPTS="$CATALINA_OPTS -Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true"
```

or when installed as a Windows service, run:

```
tomcat5 //US//JIRA ++JvmOptions="-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true
```

For other environments, and for more info on memory settings, see the memory settings page.

8. Set mail.mime.decodeparameters

The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

```
mail.mime.decodeparameters=true
```

System properties are set in different ways depending on your application server.

9. Start Tomcat

JIRA should now be ready to run in Tomcat. To start using JIRA, first start (or restart) the Tomcat server with Tomcat's `bin/startup.(sh|bat)` scripts, and point your browser to http://localhost:8080/jira

You should now see the Setup Wizard, which will take you through the brief setup procedure.

Troubleshooting

It is easy to make a mistake in this process, and even more so if you are trying to connect to a database other than hsqldb. First, check that you have followed the process described above:

- If you are using an external database (not hsqldb), have you set the **field-type-name** attribute in

$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml ? (step 2)
- Have you previously started JIRA with an incorrect **field-type-name** value? If so, the database schema would have been created incorrectly.
- If you have made changes to **$JIRA_INSTALL/edit-webapp/WEB-INF/classes/entityengine.xml** (step 2) and re-run the build script (step 3), but your changes are not being picked up, delete the Tomcat `webapps/jira` directory, then restart JIRA. It would seem that in some circumstances Tomcat does not correctly re-expand the web application.
- Have you copied the extra Tomcat jars (step 4)? Check if you have `common/lib/objectweb-datasource-1.4.3.jar` present.
- If using an external database, did you copy the JDBC driver jar to `common/lib/` (step 5)?
- Is the path to the .war file in **conf/Catalina/localhost/jira.xml** correct?
- Have you copied the .war file to Tomcat's **webapps/** directory? This is almost guaranteed to cause pain - please move it elsewhere, and delete any JIRA subdirectories created in **webapps/** from previous Tomcat starts.
- Have you configured JIRA centrally in **conf/server.xml** instead of in **conf/Catalina/localhost/jira.xml** ? This is fine, but then be sure you *don't* also have a **conf/Catalina/localhost/jira.xml** present.
- The log files are usually vital to debugging problems. On Windows, these will appear in the console window that loads when running `startup.bat` , or in one of the log files in the `logs/` directory. On Linux/Unix, logs will appear in a log file in `logs/`, usually `logs/catalina.out`. Check the log file for errors after startup.
- If you experience high memory usage / memory leaks (e.g. OutOfMemoryError), you may wish to set the system property `-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true` in **setenv.sh** / **setenv.bat.** This property is only valid for Tomcat 5.5.15 and later. For more information please see JRA-10145.
- If the connection to your MySQL database is dropping, you will need to set up Tomcat to survive connection closures.
If you're stuck, please raise a support request, and attach your logs, configuration files, plus anything else relevant, and we'll get back to you as soon as possible. If you have a general question, please try the jira-user mailing list (which Atlassian staff monitor).

User-contributed notes

Have experiences to share with Tomcat 5.5.x and JIRA? We welcome your thoughts. Please see the user-contributed Tomcat 5.5.x notes.

Installing JIRA on Oracle WebLogic

These instructions will help you install JIRA on WebLogic 8.1 Service Pack 2 and above.

Compatibility notes:

- JIRA will not run on WebLogic 8.1 without Service Pack 2, due to WebLogic bugs (specifically CR112484). JIRA will not run on WebLogic 6.1 as it has filter issues and is not J2EE 1.3 compliant.
- Some users have experienced license-related problems due to a bug in JDK 1.4.2_08 (1.4.2_05 is known to work).
- According to one user Weblogic 9 ships with a broken JDK (possibly the same bug mentioned above), so use 9.1 or 9.2 instead.

To avoid performance problems, be sure to read the weblogic.xml modifications section below.

On this page:

- 1. Unpack JIRA
- 2. Install Weblogic
- 3. Configure the WebLogic datasource
  - 3.1. Download the JDBC driver
  - 3.2. Create a connection pool
  - 3.3. Configure the datasource
- 4. Configure JIRA
  - 4.1 Configure the database connection
  - 4.2 Set JIRA Home
- 5. Configure WebLogic
  - Disabling JSP reload checks
  - Disabling servlet reload checks
  - Avoiding JSP recompiles on redeploy
- 6. Generate Weblogic WAR file
- 7. Deploy JIRA to WebLogic server
- 8. Set mail.mime.decodeparameters
- 9. Start JIRA
- User-contributed notes

1. Unpack JIRA

Download and unzip JIRA (but not with XP's unzipper nor the default tar utility on Solaris). Ensure that you download the **WAR/EAR version,** not the Standalone version that is recommended on the Downloads page.

Avoid the Windows XP built-in unzip tool! The built-in unzip tool in Windows XP is broken — it silently fails to extract files with long names (see JRA-2153). Other users have also reported problems using WinRAR. Please use another tool like WinZIP to unpack JIRA.

Avoid the Solaris default tar utility! On Solaris, please use **GNU tar** to unpack JIRA in order to handle long filenames. Do not use the Solaris default tar utility.

A new directory containing JIRA will be created, hereafter referred to as `$JIRA_INSTALL`.

If you are using **Linux/UNIX:** A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under and therefore can potentially be abused. Here is an example of how to create a dedicated user to run JIRA in Linux:
```
$ sudo /usr/sbin/useradd --create-home --home-dir /usr/local/jira --shell /bin/bash jira
```

2. Install Weblogic

1. Install WebLogic on your machine (unless already installed). The WebLogic installation directory will be hereafter referred to as **$WEBLOGIC_INSTALL.**
2. Create a domain named **atlassian** using the Domain Configuration Wizard.
3. Create a server for this domain named **jira.**
4. Create an admin user for this domain (name = **weblogic,** password = **weblogic).**

3. Configure the WebLogic datasource

3.1. Download the JDBC driver

WebLogic ships with several database drivers. If your database driver is already installed, go to section 3.2. If your database driver is not already installed, follow these steps:

1. Create directory **"$WEBLOGIC_INSTALL\user_projects\lib"**
2. Download a JDBC driver for your database (see the list of databases supported by JIRA). Copy the JDBC driver jar file to the $WEBLOGIC_INSTALL\user_projects\lib directory.
3. Add the path to the JDBC driver jar file to the WebLogic classpath, by editing "$WEBLOGIC_INSTALL\user_projects\atlassian\startWebLogic.cmd" (the following example is for the HSQL database):

```
set SERVER_NAME=jira
set WLS_USER=weblogic
set WLS_PW=weblogic
set CLASSPATH=%CLASSPATH%;$WEBLOGIC_INSTALL\user_projects\lib\hsqldb-1.7.1.jar
```

3.2. Create a connection pool

The following steps apply to the WebLogic 8.1 administration console:

1. Open the WebLogic Server Console
2. Navigate to JIRA > Services > JDBC > Connection Pools
3. Click the "Configure a new JDBC Connection Pool..." link
4. Select your "Database Type" from available types, or choose "Other" to configure a database type that is not listed.
5. Select your "Database Driver" from available options. (To install a new driver, refer to section 3.1 above).
6. Enter the "Name" for this connection pool, e.g. "JIRA Connection Pool".
7. Enter the rest of the connection properties, which are specific to your database.
8. Click the "Test" button to verify the database connection.
9. Click through the wizard to finish the configuration of the connection pool.

3.3. Configure the datasource

The following steps apply to the WebLogic 8.1 administration console:

1. Open the WebLogic Server Console
2. Navigate to JIRA > Services > JDBC > Data Sources
3. Click the "Configure a new JDBC Data Source..." link
4. Enter the "Name" for this data source. For example, "Jira Data Source".
5. Enter the "JNDI Name". Type "JiraDS", which is the default used for JIRA. This name will be referred to as **jndi-name** from the

entityengine.xml configuration file that you will configure in step 4.2 below.
6. Choose the "Pool Name" that you created in step 3.2.6 above.
7. Click through the wizard to finish the configuration of the data source.

4. Configure JIRA

4.1 Configure the database connection

Modify the transaction factory tag in the **`entityengine.xml`** file, located in the `edit-webapp\WEB-INF\classes` directory.

1. Locate the section:

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
    user-transaction-jndi jndi-server-name="default" jndi-name="java:comp/env/UserTransaction"
/>
    <transaction-manager-jndi jndi-server-name="default" jndi-name=
"java:comp/env/UserTransaction"/>
</transaction-factory>
```

   and change this to:

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
    <user-transaction-jndi jndi-server-name="default" jndi-name="java:comp/UserTransaction"/>
    <transaction-manager-jndi jndi-server-name="default" jndi-name="java:comp/UserTransaction"
/>
</transaction-factory>
```

2. Now configure the JNDI path where JIRA will expect to find a database connection factory. This is done by editing the <datasource> tag at the bottom of entityengine.xml. For example, using mssql, your <datasource> entry would be:

```
<datasource name="defaultDS" field-type-name="mssql"
    helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true" use-foreign-keys="false"
    use-foreign-key-indices="false" check-fks-on-start="false"
    check-fk-indices-on-start="false" add-missing-on-start="true">
    <jndi-jdbc jndi-server-name="default" jndi-name="JiraDS"/>
</datasource>
```

   The jndi-name **(JiraDS** in the example above) must be the same as defined in step 3.3.5.
   See the Entity Engine documentation for full details.

4.2 Set JIRA Home

To specify the location of your JIRA Home Directory (note that you need to do this *before* you build JIRA):

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

5. Configure WebLogic

Weblogic supports some additional webapp configuration parameters in the `weblogic.xml` file. This file should be created in the `edit-webapp/WEB-INF/` directory of the JIRA distribution. Here is a typical example for Weblogic 8.1:

```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE weblogic-web-app PUBLIC "-//BEA Systems, Inc.//DTD Web Application 8.1//EN"
"http://www.bea.com/servers/wls810/dtd/weblogic810-web-jar.dtd">
<weblogic-web-app>

    <jsp-descriptor>

      <jsp-param>
        <param-name>pageCheckSeconds</param-name>
        <param-value>-1</param-value>
      </jsp-param>

      <jsp-param>
        <param-name>precompile</param-name>
        <param-value>false</param-value>
      </jsp-param>

      <jsp-param>
        <param-name>workingDir</param-name>
        <param-value>./jsp_precompile_dir/jira</param-value>
      </jsp-param>

    </jsp-descriptor>

    <container-descriptor>
      <servlet-reload-check-secs>-1</servlet-reload-check-secs>
    </container-descriptor>

    <context-root>jira</context-root>

  </weblogic-web-app>
```

The full `weblogic.xml` syntax is described in BEA's documentation. Here we describe some important parameters you should consider setting.

Disabling JSP reload checks

JIRA's performance is considerably affected by Weblogic's tendency to scan JSPs for changes on every page load. This can be disabled by setting **pageCheckSeconds** to **-1.**

Disabling servlet reload checks

Similar to the JSP reload checks, Weblogic will scan servlets for modifications every second (by default), hurting performance. This can be prevented by setting the **servlet-reload-check-secs** element to **-1.**

Avoiding JSP recompiles on redeploy

If you Weblogic instance is often restarted, performance can be improved by explicitly setting a directory to save compile JSPs to (the **workingDir** parameter), and setting `precompile` to 'false'.

6. Generate Weblogic WAR file

Edit `build.xml` and customise the `bea.home` or `wl.home` properties. Then run `build.sh weblogic.war` (Unix) or `build.bat weblogic.war` (Windows) to generate aWARfile in `dist-weblogic/`

A Windows compatibiity issue is currently being tracked at JRA-11675.

7. Deploy JIRA to WebLogic server

The following steps apply to the WebLogic 8.1 administration console:

1. Open the WebLogic Server Console
2. Navigate to jira — Deployments — Web Application Modules
3. Click the link "Deploy a new Web Application Module..."
4. Locate and select the JIRA WAR file

5. Enter the "Name" for this web application module, for example "Atlassian JIRA"
6. Click though the wizard to finish the deployment of JIRA web application.

8. Set mail.mime.decodeparameters

The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

```
mail.mime.decodeparameters=true
```

System properties are set in different ways depending on your application server.

9. Start JIRA

When you access the path at which you have deployed JIRA, via your web browser, the Setup Wizard should appear.

User-contributed notes

Have experiences to share with Weblogic and JIRA? We welcome your thoughts. Please see the user-contributed Weblogic notes.

Installing JIRA on IBM Websphere 6.x

Warning
Websphere 6.0 and 6.1 are untested, but reported to work — use at your own risk.

This forum post was contributed by a user who installed JIRA on WAS 6.1.

Warning
Make sure that your `entityengine.xml` is well-formed XML. Websphere "swallows" the error messages you should get in your log file if entityengine.xml is not well-formed, and instead reports a spurious error message.

You can test that your `entityengine.xml` is well-formed by opening it in Internet Explorer or Firefox. These browsers will display an error if the file is not well-formed XML, and will indicate the location of the error in the file.

JIRA WAR-EAR Configuration Overview

While the individual server install guides provide specific instructions, it is useful to have an overall conceptual overview of what the configuration process involves.

Webapp layout

After downloading and unpackaging the JIRA webapp, one is presented with a directory containing:

```
appendcp.bat
build.bat
build.sh
build.xml
edit-webapp/
etc/
readme.txt
tools/
webapp/
```

The build.xml file is an Ant file, which when invoked with the `build.(sh|bat)` script, will construct a deployable webapp. build.xml does this by copying the contents of the `webapp/` directory, and *overwriting* it with the contents of `edit-webapp/`. Thus, **never edit files in the webapp/ directory!** If a file needs editing, first copy it from `webapp/path/to/file` to `edit-webapp/path/to/file`, and edit it there.

Database configuration

The biggest part of configuring JIRA is connecting it up to a database. InJ2EEapps like JIRA, the server (Tomcat, Orion etc) is responsible for

setting up a Database connection factory, and then makes this available to the webapp (JIRA) via a lookup API calledJNDI. So all the hard work, setting up JDBC URLs, usernames and passwords, is done in the app server. All JIRA needs is the JNDI address where it can find the database connection, in the form of a **java.sql.DataSource** object.

Beyond just getting a database connection, JIRA also needs to know what *type* of database it is dealing with, so it can issue appropriate SQL commands.

Transaction Manager configuration

J2EE apps also rely on a Transaction Manager to coordinate updates across multiple databases. While JIRA does not currently use this facility, JIRA's underlying library (OfBiz) still requires a transaction manager object. As with database connections, this is provided by the application server, as a **javax.transaction.UserTransaction** object looked up via JNDI.

User management configuration

JIRA stores all user profiles in a database table. Occasionally, integration with external user management systems like LDAP is required. This requires some configuration of `osuser.xml`. See The LDAP Integration guide for more information.

So to recap, JIRA needs to know at least three things:

- The JNDI address of its database connection (a **java.sql.DataSource** object).
- What type of database it is dealing with.
- The JNDI address of a Transaction Manager (a **javax.transaction.UserTransaction** object).

This is all done in the `entityengine.xml` file, as described in the database configuration guide. You'll find a copy of `entityengine.xml` is `edit-webapp/WEB-INF/classes`, ready to be edited.

There are a few app servers which, for various reasons, we have been unable to use the normal JNDI lookup procedure to obtain a datasource/transaction manager. See the individual app server guides for more details on how these are configured - it generally means that more configuration details end up in `entityengine.xml`.

Configuring the Entity Engine for JIRA

The Entity Engine from the OFBiz project is what JIRA uses to persist data to a database. You can find out more about why we chose the EE at the bottom of this page. See the configuration overview for a conceptual overview of what is being done here.

On this page:

- Configuring the Entity Engine for JIRA
  - Transaction Factory
- Altering the Entity Model
- Why we chose the Entity Engine

Configuring the Entity Engine for JIRA

The configuration of the Entity Engine is done through an XML file called `entityengine.xml`. This file is used to define parameters for persistence servers such as JDBC datasource parameters.

For JIRA, this file is located in the distribution at `edit-webapp/WEB-INF/classes/entityengine.xml`.

As outlined in the overview, the settings which generally need to be configured are:

- **Transaction Factory** — see below
- **field type** — edit the `field-type-name` attribute of the `<datasource>` tag.
- **datasource location:**
  - edit the `jndi-name` attribute of the `<jndi-jdbc>` tag relevant to your database.
  - For certain schema-aware databases **(Postgres, DB2),** add a **schema-name** attribute specifying the schema the database uses. See note in entityengine.xml
  - For databases with table/column name length limits (e.g. **DB2** on certain platforms), you may need to set a **constraint-name-clip-length** parameter. See note in entityengine.xml

Transaction Factory

By default the Entity Engine tries to obtain a JTA transaction factory from the application server using JNDI. This table shows the different values for different application servers:
Orion, Tomcat 4.x, Tomcat 5.0, Jetty and Weblogic (see also the Orion, Resin, Tomcat, Jetty and Weblogic guides)

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction"/>
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction"/>
</transaction-factory>
```

Tomcat 5.5 (see also the Tomcat 5.5 install guide)

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/env/UserTransaction"/>
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/env/UserTransaction"/>
</transaction-factory>
```

Resin 3 (see also the Resin 3 install guide

```
<transaction-factory
  class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="java:comp/UserTransaction" />
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:comp/TransactionManager" />
</transaction-factory>
```

JBoss (see also the [JBoss 3.x] and [JBoss 4.x] guides)

```
<transaction-factory class="org.ofbiz.core.entity.transaction.JNDIFactory">
  <user-transaction-jndi jndi-server-name="default"
    jndi-name="UserTransaction"/>
  <transaction-manager-jndi jndi-server-name="default"
    jndi-name="java:/TransactionManager"/>
</transaction-factory>
```

Altering the Entity Model

The Entity Model describes the table and column layout that JIRA uses in a database. It can be completely altered without changing any of the internal workings of JIRA.

The model provided should work with almost any database (care has been taken to ensure the column and table names are SQL compliant).

The entity model is configured through an XML file called `entitymodel.xml` (located in the distribution at `webapp/WEB-INF/classes/entitydefs/entitymodel.xml`). To edit this file, copy it to `edit-webapp/WEB-INF/classes/entitydefs/entitymodel.xml` and make changes there. When the WAR/EAR is built using build.(sh|bat), the 'edit-webapp' version of the file will be used.

The format of the file should be fairly self explanatory - basically JIRA always refers to the `entity-name` and `field-name` attributes within the code. The `type` attribute of a `<field>` tag should always match the `type` attribute of a `<field-type-def>` tag in your `fieldtype-*.xml` files.

To change where entities and fields are persisted in your database, simply add (or edit) the attribute `table-name` (for entities) or `col-name` (for fields).

Why we chose the Entity Engine

We chose the EE over CMPorBMPentity beans because:

- it is more portable between application servers
- table schemas are automatically created and updated

- using the field type definitions, we can add support for new databases very quickly
- it is faster than most CMP implementations and has some nice caching features

This document deals with configuring the entity engine for JIRA (but should be applicable to most applications). For more details on the entity engine itself and it's inner workings, see:

- OFBiz Entity Engine Guide describes the theory behind the entity engine, its architecture and usage patterns
- OFBiz Entity Engine configuration guide describes all of the entity engine configuration options, whereas this document just describes configuring the entity engine for JIRA
- API Docs the API docs for the `org.ofbiz.entity` package

Switching Application Servers

To move JIRA from one application server to another, for example from WebSphere to Tomcat, use one of the following methods:

Note
Regardless of which method you use, backup your data first and make sure you test JIRA on the new server before deploying it in production.

Method 1. Export and import the database

Follow the Upgrading JIRA instructions, installing the new version of JIRA using the installation guide for your new application server.

Method 2. Use your existing database

In the particular case that you are continuing to use the same edition (e.g. Enterprise) and version (e.g. 3.13.2) of JIRA, you do not have to export your data. You can use your existing database with the new application server. However, you cannot simply copy the WAR file or expanded WAR directory from an old JIRA EAR/WAR version in the old application server to the new application server. **This will not work.**

Follow these instructions:

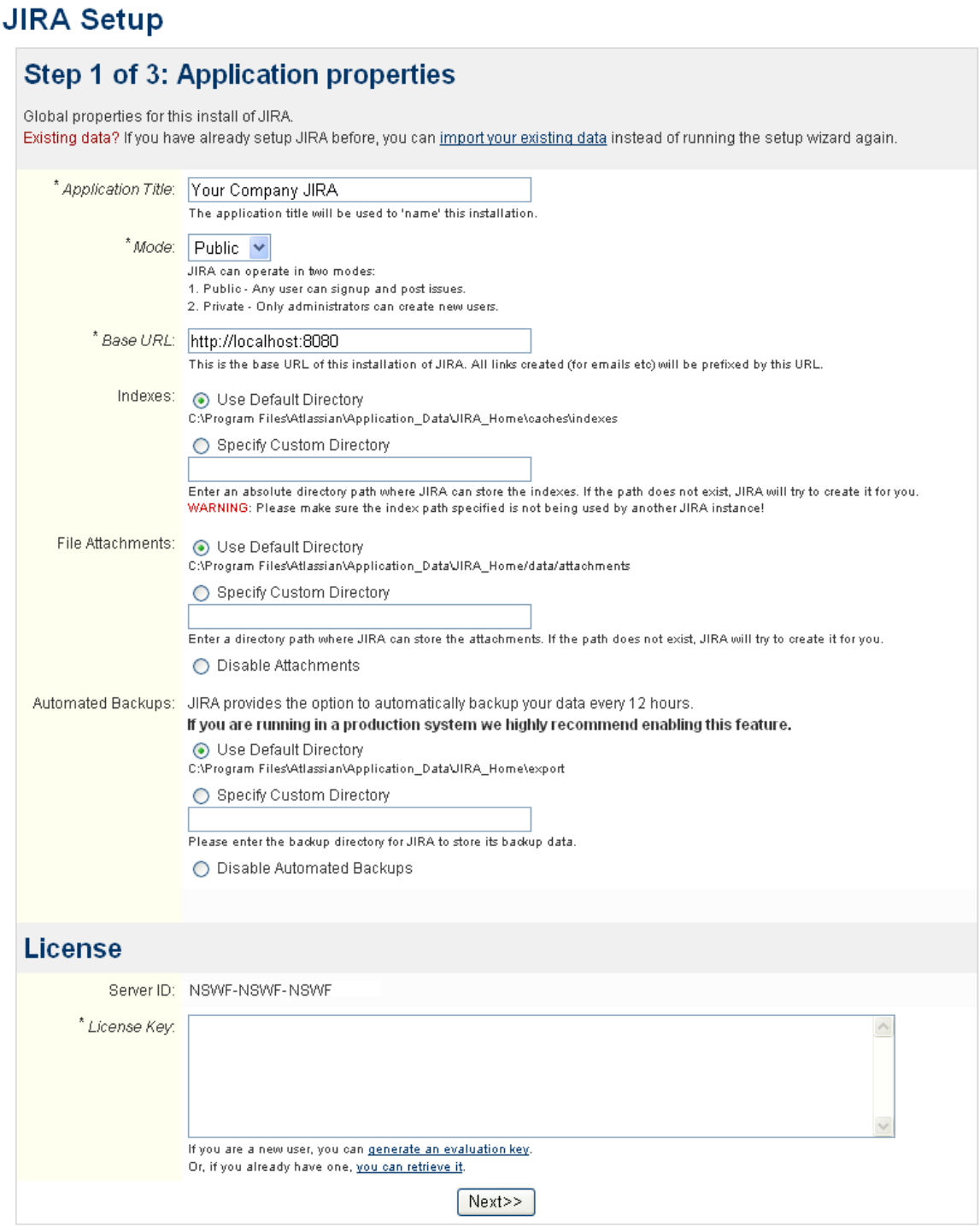
1. Install JIRA on your new application server using the relevant installation guide. As part of these instructions:
    - When you configure the JDBC datasource in your new server, use the settings for your existing database.
    - Check that the JNDI location of the UserTransaction as declared in the entityengine.xml file is correct for your server.
2. Make sure you shutdown the old server before you startup the new one.
3. If you are running the new application server on a different machine to the old one, carry out the following actions as soon as you start the new server:
    - Re-index your data.
    - Make sure that the attachment path is valid for the new server.

Running the Setup Wizard

After you have installed JIRA, and accessed its URL (eg. http://localhost:8080 or http://localhost:8080/jira) for the first time, you will be presented with a brief setup wizard to configure JIRA.

Step 1 of 3: Application Properties

The first page of the wizard looks like this:



On this page you can set some of the JIRA configuration settings, and enter your license key.
We **strongly** recommend that you specify a backup path as this will allow JIRA to periodically backup the database.

JIRA configuration settings

For more details on the settings and what they mean, see Configuring JIRA.

The first page of the wizard will also prompt you to accept or enter new locations for the following:

- **Indexes** — the default location for your indexes is in the `caches/indexes` sub-directory under your JIRA Home Directory. If you are considering changing the default location, please read Search Indexing first.
- **File Attachments** — the default location for your indexes is in the `data/attachments` sub-directory under your JIRA Home Directory. If you are considering changing the default location, please read Enabling File Attachments first.
- **Automated Backups** — the default location for your indexes is in the `exports` sub-directory under your JIRA Home Directory. If you are considering changing the default location, please read Automating JIRA Backups first.

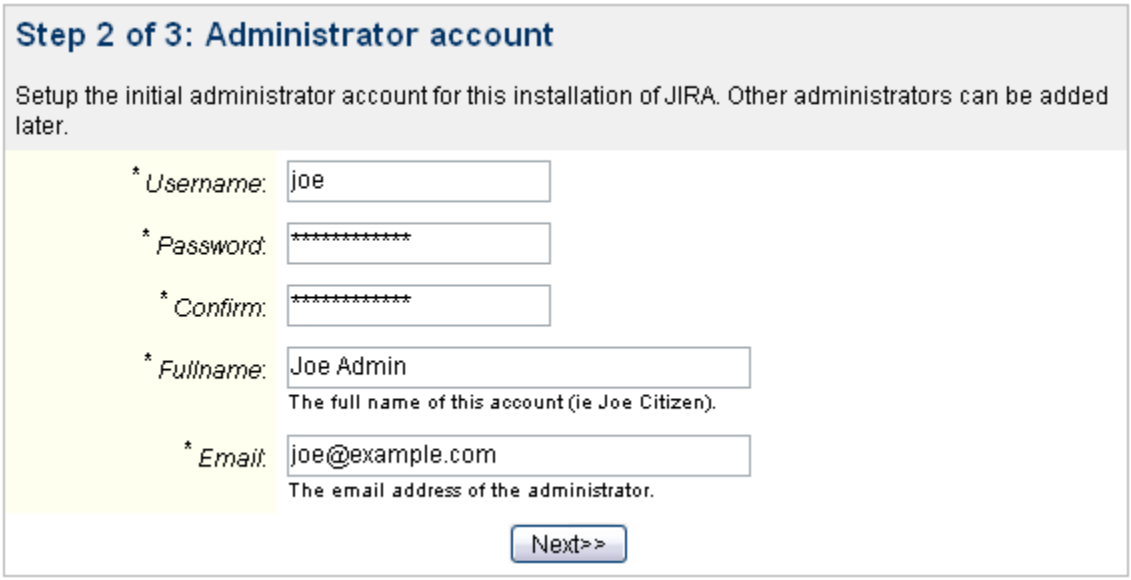
Evaluation license key

You are required to enter a valid license key before you can use JIRA. You can obtain an evaluation license key which will allow JIRA to run unrestricted for 30 days. To use your existing license key or obtain a new license key, follow the steps below:

1. If you are a new user, you will need to create an account to generate an evaluation license key. Otherwise, log in to the Atlassian website and navigate to the 'JIRA - Licenses' page.
2. Once you have created an account or logged in with an existing account, the 'JIRA - Licenses' page will display with a list of your existing JIRA licenses. If you do not have any JIRA licenses, generate one by clicking on the 'Generate evaluation license' link for the edition of JIRA you are installing (Standard/Professional/Enterprise). A new evaluation license will be generated and displayed on the page.
3. Click on the 'View' link to view the license key of the evaluation license that you wish to use. Copy the license key to your clipboard and paste it into the 'License Key' field of the Setup Wizard.

Step 2 of 3: Administrator Account

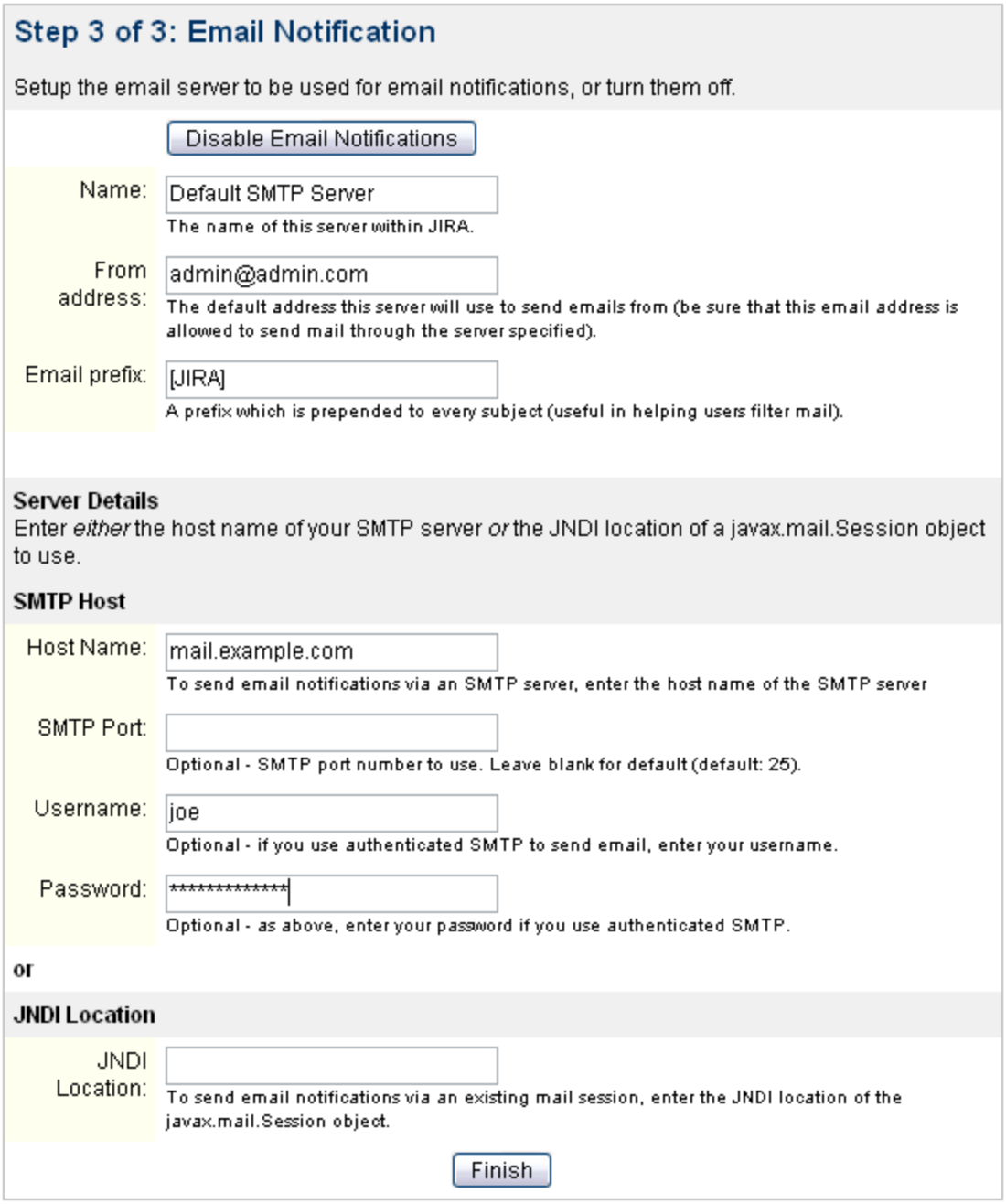
After completing the first page, the second page of the wizard sets up an administrator account:



Once this initial administrator account is created, that administrator can then create other administrators.

Step 3 of 3: Email Notification

The last page of the setup wizard allows you to configure the outgoing emails from JIRA.



To disable email notifications (you can always enable them later), just click the **Disable Email Notifications** button.

For the outgoing mail server choice, you must either specify the JNDI location of a `javax.mail.Session` object (usually provided by your application server) or specify the location of an SMTP server.

Once you complete this step, JIRA should be set up and ready for use. For details please see:

- [*JIRA User's Guide*]
- [*JIRA Administrator's Guide*]

Connecting JIRA to a Database

JIRA requires a relational database for storage of issue data.

The time at which you connect JIRA to a database depends on whether you are using the JIRA Standalone distribution or the JIRA WAR/EAR distribution:

JIRA Standalone:	As part of its installation process, JIRA Standalone automatically installs, configures and connects itself to an HSQLDB database. This is fine for evaluation purposes, however HSQLDB is prone to database corruption. For production installations, we strongly recommend that you connect JIRA to an enterprise database. This also lets you take advantage of existing database backup and recovery procedures.
JIRA WAR/EAR:	When you install JIRA WAR/EAR, you will need to manually configure your database connection.

The following instructions apply to both JIRA Standalone and JIRA WAR/EAR:

- SQL Server
- MySQL
- PostgreSQL
- Oracle
- DB2
- Firebird
- Sybase
- HSQLDB

Which database?

Your choice of database can significantly affect your subsequent experience of JIRA administration. If you have a choice of databases, please first read this review of supported databases.

If you are looking for a low-cost solution, consider using MySQL or PostgreSQL as both of these are open source (free) software.

Data migration

To transfer your issue data from one database to another, please refer to the instructions for Switching databases.

Recommended Databases

Atlassian highly recommends the following databases for use with your JIRA installation. You are welcome to use other databases with your JIRA installation, however the four databases below are the ones most commonly used by our customers and we are able to provide the best support for them.

On this page:

- PostgreSQL
- MySQL
- SQL Server
- Oracle

PostgreSQL

References
Official JIRA documentation
Supplementary documentation
PostgreSQL website
Notes
• Please see the official JIRA documentation for information on which versions of PostgreSQL are supported, which JDBC drivers to use and for instructions on connecting JIRA to PostgreSQL. • PostgreSQL is an open source database. • PostgreSQL defaults to Unicode. • Atlassian uses PostgreSQL for its web applications and websites.

MySQL

References
Official JIRA documentation
Supplementary documentation
MySQL website
Notes

- Please see the official JIRA documentation for information on which versions of MySQL are supported, which JDBC drivers to use and for instructions on connecting JIRA to MySQL.
- MySQL is an open source database.
- Unicode problems have been noted with MySQL 4.0 and earlier. This issue is fixed in MySQL 4.1 and later, however modern SQL (subselects) is only available MySQL 5.0 and later.

SQL Server

References
Official JIRA documentation
Supplementary documentation
SQL Server website
Notes

- Please see the official JIRA documentation for information on which versions of SQL Server are supported, which JDBC drivers to use and for instructions on connecting JIRA to SQL Server.
- SQL is a commercial database developed by the Microsoft corporation.
- Problems have been noted when using the Microsoft JDBC driver for SQL Server with JIRA. Please refer to the official JIRA documentation for further details.

Oracle

References
Official JIRA documentation
Supplementary documentation
Oracle website
Notes

- Please see the official JIRA documentation for information on which versions of Oracle are supported, which JDBC drivers to use and for instructions on connecting JIRA to Oracle.
- Oracle is a commercial database developed by the Oracle corporation.
- Problems have been noted when using particular versions of the Oracle server and JDBC driver with JIRA. Please refer to the official JIRA documentation for further details.

Connecting JIRA to SQL Server 2005

Note: due to numerous reported performance issues with SQL Server 2000, it is strongly recommended that you use SQL Server 2005 instead. The following instructions apply only to SQL Server 2005, not to SQL Server 2000, for which slightly different configuration is required.

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Configure SQL Server

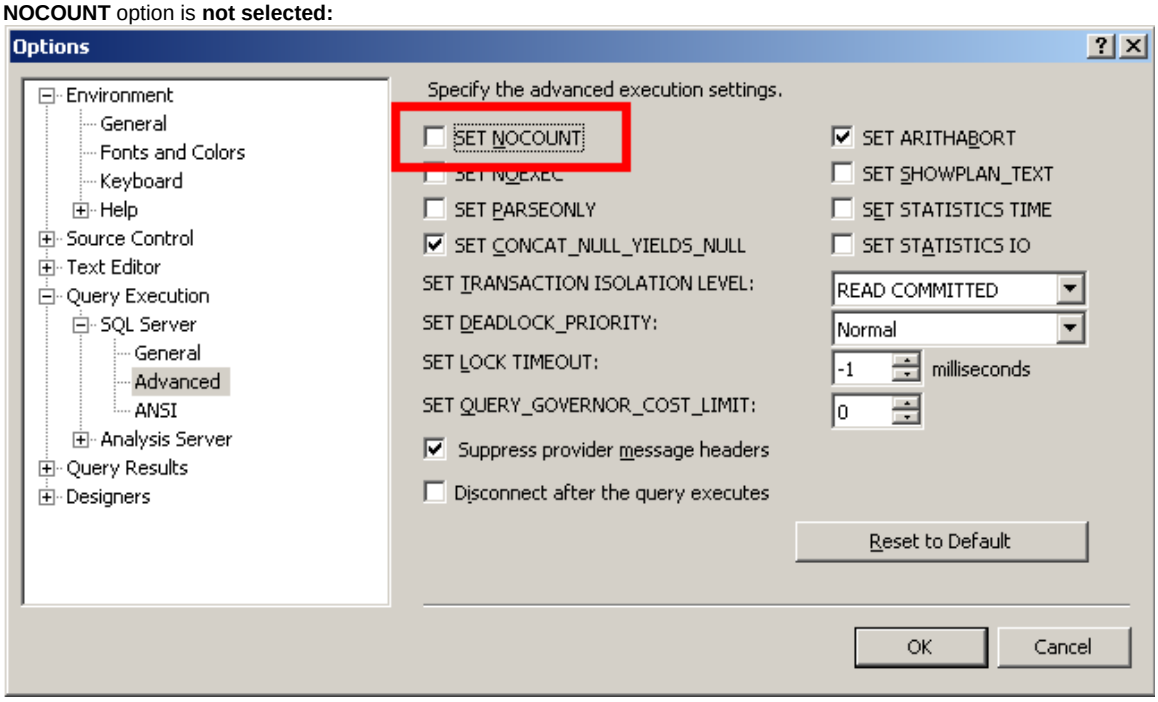
- 2. Copy the SQL Server driver to your application server
- 3. Configure your application server to connect to SQL Server
- 4. Configure the JIRA Entity Engine
- Next steps
- User-contributed notes

1. Configure SQL Server

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**). Note that `jirauser` should **not** be the database owner, but **should** be in the `db_owner` role. (See SQL Startup Errors for details.)
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Create an empty 'schema' in the database (e.g. **`jiraschema`**) for the JIRA tables. Please note that a 'schema' in SQL Server 2005 is a distinct namespace used to contain objects, and is **different** from a traditional database schema. You are not required to create any of JIRA's tables, fields or relationships (JIRA will create these objects in your empty schema when it starts for the first time). You can read more on SQL Server 2005 schemas in the relevant Microsoft documentation.
4. Ensure that the user has permission to connect to the database, and create and populate tables in the newly-created schema.

> If you are having difficulties setting up your JIRA database for SQL Server, additional information is available in the Setting up your JIRA database for MS SQL Server 2005 document.

5. Ensure that TCP/IP is enabled on SQL Server and listening on the correct port (the port is 1433 for the default instance of SQL Server). Read the Microsoft documentation for information on how to enable a network protocol (TCP/IP) and how to configure SQL server to listen on a specific port.
6. Ensure that SQL Server is operating in the appropriate authentication mode. By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, i.e. 'Microsoft SQL Server, Error: 18452' is received during JIRA startup, you will need to change the authentication mode to 'Mixed Authentication Mode'. Read the Microsoft documentation on authentication modes and changing the authentication mode to 'Mixed Authentication Mode'
7. Turn off the SET NOCOUNT option. (The JIRA on MS SQL Server document provides details on the errors that occur if SET NOCOUNT is set.) To turn off SET NOCOUNT:
   - Open **SQL Server Management Studio** and navigate to **Tools -> Options -> Query Execution -> SQL Server -> Advanced.** The following screenshot displays the configuration panel for this setting in MSSQL Server 2005. Ensure that the **SET NOCOUNT** option is **not selected:**



2. Copy the SQL Server driver to your application server

1. Download the SQL Server JDBC driver from JTDS (recommended, assumed below), or I-net software (commercial).

> **Note**
> Microsoft have their own JDBC driver but we **strongly recommend avoiding it** after receiving many reports of intermittent disconnections (JRA-5760, JRA-6872), workflow problems (JRA-8443) and Chinese character problems (JRA-5054).

2. Add the SQL Server JDBC driver jar (**`jtds-[JIRADOC:version].jar`**) to the `common/lib/` directory.

3. Configure your application server to connect to SQL Server

1. Edit the server configuration file and customise the `username` , `password` , `driverClassName` and `url` parameters for the Datasource, as shown in the code sample below.
    - If you are using JIRA Standalone, the server configuration file that you need to edit is **`conf/server.xml`** .
    - If you are using JIRA WAR/EAR, edit the appropriate file on your application server, e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`** .

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="net.sourceforge.jtds.jdbc.Driver"
            url="jdbc:jtds:sqlserver://localhost:1433/jiradb"
[ delete the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis params here ]
            />

          <Resource name="UserTransaction" auth="Container" type="javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file — search for mentions of 'jira' in the files under `conf/` .)

If you are using JIRA Standalone, you will also need to edit `conf/server.xml` , and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes. These attributes are only needed for HSQL, and will degrade performance if they are not removed.

4. Configure the JIRA Entity Engine

1. Edit the JIRA Entity Engine configuration file and change the `field-type-name` attribute to **`mssql`** .
    - If you are using JIRA Standalone, the JIRA Entity Engine configuration file that you need to edit is **`atlassian-jira/WEB-INF/classes/entityengine.xml`** .
    - If you are using JIRA WAR/EAR, the JIRA Entity Engine configuration file that you need to edit is **`edit-webapp/WEB-INF/classes/entityengine.xml`** . If you forget to do to make this change and start JIRA, it may create database tables incorrectly. See this page if this happens to you.
2. Change **`schema-name="PUBLIC"`** to the name of the schema associated with the database (i.e. the schema you created in step 1.3 above), e.g. **`schema-name="jira"`** . Note that the schema must exist in the database before you perform this step.

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="mssql"
      schema-name="jira"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true"
    use-foreign-keys="false"
    ...
```

If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. The next step is to start it up!

- If you are using JIRA Standalone, start it up and watch the logs for any errors.
- If you are using JIRA WAR/EAR, rebuild and redeploy the webapp in your application server.
  Once you have the JIRA server running, you can try accessing the JIRA application in your browser.

User-contributed notes

Have experiences to share with SQL Server and JIRA? We welcome your thoughts. Please see the user-contributed MS SQL Server notes.

Connecting JIRA to MySQL

This document applies to MySQL 3.53.x up to 5.1.x.

Note: A Linux-specific version of these instructions is available.

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

If you are using a MySQL database with any of the following:

- JIRA 3.13 Standalone or above,
- version 5.5.25 or higher of Tomcat 5,
- version 6.0.13 or higher of Tomcat 6,

you may experience problems with your connections dropping out (see http://jira.atlassian.com/browse/JRA-15731 for details). Please read Surviving Connection Closures for information on the changes required to your application server to address this.

On this page:

- 1. Configure MySQL
- 2. Copy the MySQL driver to your application server
- 3. Configure your application server to connect to MySQL
- 4. Configure the JIRA Entity Engine
- Next steps
- User-contributed notes

1. Configure MySQL

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**). The database must have a character set of UTF8. Enter the following command from within the MySQL command client:

```
create database jiradb character set utf8;
```

(if you want your database to be named **`jiradb`**).

3. Ensure that the user has permission to connect to the database, and create and populate tables.

2. Copy the MySQL driver to your application server

1. Download the MySQL Connector/J JDBC driver, e.g. version 3.1.12 (tar.gz or zip) which we have verified to work. NOTE: the older 3.1.11 driver is broken.

   > A user has reported encountering problems using the Resin JDBC driver for MySQL. However, the Connector/J driver from MySQL works correctly (except for version 3.1.11).

2. Add the MySQL JDBC driver jar (**`mysql-connector-java-3.x.x-bin.jar`**) to the `common/lib/` directory. NOTE: Do not place the Debug Driver (`mysql-connector-java-3.x.x-bin-g.jar`) on the CLASSPATH as this can cause issues (JRA-8674).

3. Configure your application server to connect to MySQL

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName` and `url` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**.)

   > The URL in the XML below assumes a UTF-8 database — i.e. created with `create database jiradb character set utf8;`. If you don't specify `character set utf8` you risk getting '`Data truncation: Data too long for column`' errors when importing data or corruption of non-supported characters. See storing non-ASCII characters in MySQL for details.

   Note: if entered into an XML file, escape the '&' with '&' as follows:

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost/jiradb?useUnicode=true&amp;characterEncoding=UTF8"[
delete the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis params here ]maxActive=
"20"
validationQuery="select 1"/>

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file - search for mentions of 'jira' in the files under `conf/`.)

MySQL closes idle connection after 8 hours, so the `autoReconnect=true` is necessary to tell the driver to reconnect

2. If you are using JIRA Standalone, edit `conf/server.xml`, and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes (which are only needed for HSQL, and degrade performance otherwise).
3. If you want to set up connection validation for your application server (i.e. your database connections are dropping out), you will need to add the `validatonQuery` parameter to the Datasource and set it to "select 1". See the XML above for an example of this.

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to **`mysql`**. (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you). Also delete the `schema-name="PUBLIC"` attribute, if it exists:

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="mysql"[ delete this, if it exists: schema-name=
"PUBLIC" ]
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
      check-on-start="true"
      use-foreign-keys="false"
      ...
```

If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

Have experiences to share with MYSQL and JIRA? We welcome your thoughts. Please see the user-contributed MYSQL notes.

Connecting JIRA to PostgreSQL

Note: *A version of these instructions specific to Linux and JIRA Standalone is available.*

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Configure PostgreSQL
- 2. Copy the PostgreSQL driver to your application server
- 3. Configure your application server to connect to PostgreSQL
- 4. Configure the JIRA Entity Engine
- Next steps
- User-contributed notes

1. Configure PostgreSQL

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Ensure that the user has permission to connect to the database, and create and populate tables.

2. Copy the PostgreSQL driver to your application server

1. Download the PostgreSQL JDBC driver from http://jdbc.postgresql.org/download.html. Get the **JDBC 3** driver specific to your PostgreSQL version, e.g. + postgresql-8.x-xxx.jdbc3.jar. Note that the 8.0-316 driver bundled with FC4 is buggy and results in "Bad+ value for type int" errors.
2. Add the Postgre JDBC driver jar to the `common/lib/` directory.

3. Configure your application server to connect to PostgreSQL

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName` and `url` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**).

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="org.postgresql.Driver"
            url="jdbc:postgresql://host:port/database" [see also
http://jdbc.postgresql.org/doc.html ]
[ delete the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis params here ]
            />

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file - search for mentions of 'jira' in the files under `conf/` .)

2. If you are using JIRA Standalone, edit `conf/server.xml`, and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to the value for your database. (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you).
    - If using **PostgreSQL 7.2 or later,** set:

      ```
      <datasource name="defaultDS" field-type-name="postgres72"
        schema-name="public"
        helper-class="org.ofbiz.core.entity.GenericHelperDAO"
        check-on-start="true"
        use-foreign-keys="false"
        use-foreign-key-indices="false"
        check-fks-on-start="false"
        check-fk-indices-on-start="false"
        add-missing-on-start="true"
        check-indices-on-start="true">
      ```

      PostgreSQL 7.2 and above require a schema to be specified. The default schema in PostgreSQL is `public`. If you are working with JIRA Standalone, be sure to change schema-name from `PUBLIC` to `public` (lowercase).

    - If using **PostgreSQL 7.1 or earlier,** set:

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="postgres"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true"
    use-foreign-keys="false"
    ...
```

For PostgreSQL 7.1 or earlier, you will also need to edit `entitymodel.xml` as described in JRA-4929, to avoid an error regarding the `POSITION` column. This is not necessary in later PostgreSQL releases.

If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

Have experiences to share with PostgreSQL and JIRA? We welcome your thoughts. Please see the user-contributed PostgreSQL notes.

Connecting JIRA to Oracle

This document applies to Oracle 9i and 10g. (Since Oracle Corporation is dropping support for Oracle 8i it has become increasingly difficult for Atlassian to support JIRA installations using this version of Oracle database. See this link for further information on Oracle's obsolescence/desupport of 8i).

Please note, a number of the Oracle server and driver versions cannot be used with JIRA or are inherently unstable. The section on JIRADOC:Oracle server and driver compatibility below lists the known issues with Oracle servers and drivers. These have been discovered at great cost to our users and we hope to prevent similar frustrations for our users.

On this page:

- Before you begin
  - Check the compatibility of your Oracle server and driver
  - Export your existing JIRA data
- 1. Configure Oracle
- 2. Copy the Oracle driver to your application server
- 3. Configure your application server to connect to Oracle
  - Tomcat (or JIRA Standalone)
    - I-Net JDBC driver
    - Oracle 10 JDBC driver
  - JIRA WAR/EAR distribution
- 4. Configure the JIRA Entity Engine
- Next steps
- Troubleshooting
  - Importing into Oracle from another database
- User-contributed notes

Before you begin

Check the compatibility of your Oracle server and driver

Oracle server:
We **strongly** recommend that you avoid using version 10.2.0.3 of the Oracle server. Oracle server version 10.2.0.3 has been noted to produce

occurrences of error ORA-01461. Oracle metalink Note:461670.1 has further details on this Oracle server issue (note, you will need an Oracle support account to view this document).
JIRA customers with this problem have reported that upgrading to Oracle server version 10.2.0.4 resolves the issue.

Oracle driver:

WARNING: *please make sure you get the* ***10.1.0.5*** *version of the driver. Every other version has problems:*

- The 9i drivers don't support CLOBs, so are unusable.
- Version 10.2.0.3.0 of the 10g Release 2 JDBC driver has been noted to produce occurrences of error ORA-01461. Oracle metalink Note:461670.1 has further details on this Oracle server issue (note, you will need an Oracle support account to view this document).
- Version 10.2.0.1.0 of the 10g Release 2 JDBC driver hangs with some databases.
- The 10g Release 1 JDBC driver (10.1.0.4) does not hang, but throws ArrayIndexOutOfBoundsExceptions. A second user reports that it silently fails to import workflows in Oracle 9i, and JIRA later dies with a NullPointerException.
- The latest 10.1.0.5 driver allegedly fixes the ArrayIndexOutOfBoundsException, and we have at least one report of it working without problems.

Anecdotally, these problems seem to affect 9i users more than 10g. Thus:

- If you are using **Oracle 9i,** we recommend you avoid Oracle's drivers altogether, and buy the I-net software JDBC driver, which is known to work without problems. Try the 10g driver at your own risk.
- If you are using **Oracle 10g,** download the 10.1.0.5 driver from Oracle's site.

Export your existing JIRA data

If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

1. Configure Oracle

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Ensure that the user has permission to connect to the database, and create and populate tables.

2. Copy the Oracle driver to your application server

1. Download the Oracle JDBC driver (from I-net software, or Oracle's site — direct link). If you are using Oracle 9i, please use Oracle 10g JDBC drivers as they are required for CLOB support.
2. Add the Oracle JDBC driver jar (**`ojdbc14.jar`**) to the `common/lib/` directory (for Tomcat), or the relevant `lib` directory in your app server.

> Both Oracle 9 and 10 drivers are called *ojdbc14.jar,* so don't rely on the name, or assume that an existing driver present in your directory is the correct one. The correct 10g R1 driver is 1,378,346 bytes, whereas the wrong 9i driver is 1.14Mb. The (broken) 10gR2 driver is 1,536,979 bytes.

3. Configure your application server to connect to Oracle

Tomcat (or JIRA Standalone)

In an editor, open **`conf/server.xml`** (JIRA Standalone) or **`conf/Catalina/localhost/jira.xml`** (regular Tomcat). Locate the section:

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
      username="sa"
      password=""
      driverClassName="org.hsqldb.jdbcDriver"
      url="jdbc:hsqldb:${catalina.home}/database/jiradb"
      minEvictableIdleTimeMillis="4000"
      timeBetweenEvictionRunsMillis="5000"
      maxActive="20" />
```

(Note: if you can't find a section like this at all, you've probably got the wrong file. Search for mentions of 'Jira' in the files under `conf/`).

Replace this section with one of the following, depending on the driver you're using.

I-Net JDBC driver

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
      username=" _[enter db username]"_
      password= _"[enter db password]"_
      driverClassName="com.inet.ora.OraDriver"
      url="jdbc:inetora:localhost:1521:jiradb"
      connectionProperties="streamstolob=true"
      maxActive="20"/>
```

Oracle 10 JDBC driver

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
      username=" _[enter db username]"_
      password= _"[enter db password]"_
      driverClassName="oracle.jdbc.driver.OracleDriver"
      url="jdbc:oracle:thin:@localhost:1521:jiradb"
      connectionProperties="SetBigStringTryClob=true"
      maxActive="20"/>
```

Customise the username, password, database server hostname (assumed to be `localhost` above, and database name (`jiradb` above).

Please ensure you have the `connectionProperties` setting, which is needed to store unlimited-text fields in Oracle. Without this, Oracle will only store strings up to 32K bytes in size.

If you were previously using hsqldb make sure you have *removed* have the `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` attributes. They will slow JIRA down if present.

JIRA WAR/EAR distribution

If you're deploying JIRA on another application server, you'll need to build a JIRA Webapp from the WAR/EAR distribution. Refer primarily to the application server setup guides for details of configuring your application server.

There is one Oracle-specific tweak required - the **`SetBigStringTryClob`** (Oracle) or **`streamstolob`** (I-Net) connection property must be set. Please see the following document for how to set this.

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and make the following changes:
    - Change the `field-type-name` attribute to **`oracle10g`**. If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you.
    - Delete the `schema-name="PUBLIC"` attribute, or replace the schema name with the schema you are using (this makes JIRA start faster too - see this thread).

    ```
    <!-- DATASOURCE - You will need to update this tag for your installation.

    -->
      <datasource name="defaultDS" field-type-name="oracle10g"
          helper-class="org.ofbiz.core.entity.GenericHelperDAO"
        check-on-start="true"
        use-foreign-keys="false"
        ...
    ```

    The `field-type-name` **must** be set to `oracle10g` **(not** `oracle`) for this configuration to work.

2. If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag). Please refer to the documentation for your application server for details.

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the

webapp in your application server.

Troubleshooting

Importing into Oracle from another database

If you have exported XML data from a JIRA instance running a different database than Oracle, and then the import into Oracle fails, you may be affected by JRA-14085.

User-contributed notes

Have experiences to share with Oracle and JIRA? We welcome your thoughts. Please see the user-contributed Oracle notes.

Connecting JIRA to DB2

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Configure DB2
- 2. Copy the DB2 driver to your application server
- 3. Configure your application server to connect to DB2
- 4. Configure the JIRA Entity Engine
- Additional notes
- Next steps
- User-contributed notes

1. Configure DB2

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Ensure that the user has permission to connect to the database, and create and populate tables.
4. Ensure that the user has a default database schema associated (e.g. Create an empty 'schema' in the database called '**`jiraschema`**').

2. Copy the DB2 driver to your application server

1. Add the DB2 JDBC driver jar **(db2jcc.jar)** to the `common/lib/` directory. (The DB2 JDBC driver is shipped with DB2.)

3. Configure your application server to connect to DB2

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName` and `url` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**.)

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="com.ibm.db2.jcc.DB2Driver"
            url="jdbc:db2://localhost:50000/jiradb"[ delete the minEvictableIdleTimeMillis and
timeBetweenEvictionRunsMillis params here ]
/>

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file - search for mentions of 'jira' in the files under `conf/`.)

2. If you are using JIRA Standalone, edit `conf/server.xml` , and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to **`db2`**. (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you.)

   If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

2. Set the value of the `schema-name` attribute to the name of the schema you are using for JIRA's database tables. Here is an example of the `<datasource>` definition in `entitymodel.xml`:

```
<datasource name="defaultDS" field-type-name="db2"schema-name="jiraschema"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
      check-on-start="true"
      use-foreign-keys="false"
      use-foreign-key-indices="false"
      check-fks-on-start="false"
      check-fk-indices-on-start="false"
      add-missing-on-start="true"
      constraint-name-clip-length="15">
        <jndi-jdbc jndi-server-name="default" jndi-name="java:comp/env/jdbc/JiraDS">
</datasource>
```

Some versions of DB2 enforce a maximum length of 18 characters for a column name. However, the OFBiz entity-engine generates column names for primary keys based on the table name - so if the table name is longer than 15 characters, DB2 will not create the table. The solution is to modify the `entitymodel.xml` and add a `constraint-name-clip-length` attribute to the `<datasource>` tag in `entitymodel.xml`. Set the value of the `constraint-name-clip-length` attribute to **15.** This does *not* apply to DB2 on iSeries, AS/400 systems.

Additional notes

- At least some versions of DB2 require the length of CLOBs and BLOBs to be explicitly set in `WEB-INF/classes/entitydefs/fieldtype-db2.xml` (`BLOB` to `BLOB(1000)` and `CLOB` to `CLOB(2000)`). See JRA-4912.
- A user reports that DB2 on z/OS will not work with JIRA, as the SQL format differs. "UDB on LUW will automatically define index spaces etc, whereas on z/OS you define them via DDL."

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

Have experiences to share with DB2 and JIRA? We welcome your thoughts. Please see the user-contributed DB2 notes.

Connecting JIRA to Firebird

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Configure Firebird
- 2. Copy the Firebird driver to your application server
- 3. Configure your application server to connect to Firebird
- 4. Configure the JIRA Entity Engine
- Additional notes
- Next steps
- User-contributed notes

1. Configure Firebird

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Ensure that the user has permission to connect to the database, and create and populate tables.

2. Copy the Firebird driver to your application server

1. Download the Firebird JDBC driver from http://firebird.sourceforge.net.
2. Add the Firebird JDBC driver jar **(`firebirdsql.jar`)** to the `common/lib/` directory.

3. Configure your application server to connect to Firebird

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName` and `url` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**.)

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="org.firebirdsql.jdbc.FBDriver"
            url=
"jdbc:firebirdsql:localhost/3050:/opt/firebird/examples/jiradb.fdb?autoReconnect=true"[ delete
the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis params here ]
            />

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file - search for mentions of 'jira' in the files under `conf/`.)

2. If you are using JIRA Standalone, edit `conf/server.xml`, and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to **`firebird`.** (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you). Also delete the `schema-name="PUBLIC"` attribute:

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="firebird"schema-name="PUBLIC"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true"
    use-foreign-keys="false"
    ...
```

If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

Additional notes

- A constraint within Firebird does not allow key sizes to be greater than 250. Due to this restriction, it should be noted that the

performance of JIRA is significantly reduced as some indexes can not be created. When starting up JIRA for the first time you will see warning messages caused by this constraint, which look like this:

```
2006-08-31 11:27:48,328 main [JIRADOC:core.entity.jdbc.DatabaseUtil] SQL Exception while executing the following:CREATE INDEX action_issue ON jiraaction (issueid, actiontype)Error was: org.firebirdsql.jdbc.FBSQLException: GDS Exception. 335544351. unsuccessful metadata updatekey size too big for index ACTION_ISSUE
```

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

Have experiences to share with Firebird and JIRA? We welcome your thoughts. Please see the user-contributed Firebird notes.

Connecting JIRA to Sybase

> **"Before you begin"**
> If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Configure Sybase
- 2. Copy the Sybase driver to your application server
- 3. Configure your application server to connect to Sybase
- 4. Configure the JIRA Entity Engine
- Next steps
- User-contributed notes

1. Configure Sybase

1. Create a database user which JIRA will connect as (e.g. **`jirauser`**).
2. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
3. Ensure that the user has permission to connect to the database, and create and populate tables.

> By default Sybase does not allow `NULL` values for a table's columns — which can be problematic for JIRA.
> A solution to this issue is to setup the Sybase database used by JIRA to allow `NULL` values by default, by setting the `ALLOW DEFAULTS ON` option. This procedure is described at: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/svrtsg/@Generic__BookTextView/15380. See also JRA-4815.

2. Copy the Sybase driver to your application server

1. Download the Sybase JDBC driver from http://jtds.sourceforge.net/
2. Add the Sybase JDBC driver jar (**`jtds-1.2.jar`**) to the `common/lib/` directory.

> **Note**
> An offical Sybase driver, called jConnect, is available. However it seems not to be able to store CLOBs larger than 16kb (see JRA-6679), and so we recommend jTDS instead

3. Configure your application server to connect to Sybase

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName` and `url` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**).

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="[enter db username]"
            password="[enter db password]"
            driverClassName="net.sourceforge.jtds.jdbc.Driver"
            url="jdbc:jtds:sybase://yourmachinename:portnumber/jiradb"
[ delete the minEvictableIdleTimeMillis and timeBetweenEvictionRunsMillis params here ]
            />

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file — search for mentions of 'jira' in the files under `conf/`.)

2. If you are using JIRA Standalone, edit `conf/server.xml`, and delete the **`minEvictableIdleTimeMillis`** and **`timeBetweenEvictionRunsMillis`** attributes (which are only needed for HSQL, and degrade performance otherwise).

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to **`sybase`**. (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you). Also delete the `schema-name="PUBLIC"` attribute:

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="sybase"schema-name="PUBLIC"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true"
    use-foreign-keys="false"
    ...
```

- If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).
- A user of JBoss and Sybase ASA 9.0.2 reported getting errors (e.g. `com.sybase.jdbc2.jdbc.SybSQLException: ASA Error -195: Column 'PASSWORD_HASH' in table 'userbase' cannot be NULL`) even after nulls were allowed (`select * from systable where "option" like '%null%'`). This was eventually resolved by modifying the JBoss datasource definition to include: `<new-connection-sql> sp_dboption jira, "allow nulls by default", true </new-connection-sql>`

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

Have experiences to share with Sybase and JIRA? We welcome your thoughts. Please see the user-contributed Sybase notes.

Connecting JIRA to HSQLDB

Before you begin: If you are already using JIRA, create an export of your data as an XML backup. You will then be able to transfer data from your old database to your new database, as described in Switching databases.

On this page:

- 1. Copy the HSQLDB driver to your application server
- 2. Configure your application server to connect to HSQLDB
- 4. Configure the JIRA Entity Engine
- Next steps
- User-contributed notes

1. Copy the HSQLDB driver to your application server

1. Download the HSQLDB JDBC driver — hsqldb-1.8.0.5.jar for JIRA 3.7+, or hsqldb-1.7.1-patched.jar for JIRA 3.6.5 and earlier. We **strongly recommend** upgrading to 3.7 if you wish to use hsqldb, as hsqldb 1.7.x is prone to data corruption.
2. Add the HSQLDB JDBC driver jar to the `common/lib/` directory.

2. Configure your application server to connect to HSQLDB

1. Edit **`conf/server.xml`** (if you are using JIRA Standalone) and customise the `username`, `password`, `driverClassName`, `url`, `minEvictableIdleTimeMillis` and `timeBetweenEvictionRunsMillis` parameters for the Datasource. (If you are using JIRA WAR/EAR, edit the appropriate file on your application server; e.g. for Tomcat, edit **`conf/Catalina/localhost/jira.xml`**.)

```
<Server port="8005" shutdown="SHUTDOWN">

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

        <Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
          <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
            username="sa"
            password=""
            driverClassName="org.hsqldb.jdbcDriver"
            url="jdbc:hsqldb:/path/to/jira/database/jiradb"
            minEvictableIdleTimeMillis="4000"
            timeBetweenEvictionRunsMillis="5000"
            maxActive="20"
            />

          <Resource name="UserTransaction" auth="Container" type=
"javax.transaction.UserTransaction"
            factory="org.objectweb.jotm.UserTransactionFactory" jotm.timeout="60"/>
          <Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="
false"/>
        </Context>

      </Host>
    </Engine>
  </Service>
</Server>
```

(Note: if you can't find this section at all, you've probably got the wrong file - search for mentions of 'jira' in the files under `conf/`.)

In Tomcat 5.x, use **'jdbc:hsqldb:${catalina.home}/database/jiradb'** and **'${catalina.home}'** will be replaced with the JIRA path at runtime.

4. Configure the JIRA Entity Engine

1. Edit **`atlassian-jira/WEB-INF/classes/entityengine.xml`** (if you are using JIRA Standalone) or **`edit-webapp/WEB-INF/classes/entityengine.xml`** (JIRA WAR/EAR), and change the `field-type-name` attribute to **`hsql`.** (If you forget to do this and start JIRA, it may create database tables incorrectly. See this page if this happens to you).

```
<!-- DATASOURCE - You will need to update this tag for your installation.

-->
  <datasource name="defaultDS" field-type-name="hsql"
      helper-class="org.ofbiz.core.entity.GenericHelperDAO"
    check-on-start="true"
    use-foreign-keys="false"
    ...
```


If you are using JIRA WAR/EAR, your application server may require other changes to `entityengine.xml` (e.g. to customise the `jndi-jdbc` tag).

Next steps

You should now have an application server configured to connect to a database, and JIRA configured to use the correct database type. If you are using JIRA Standalone, start it up and watch the logs for any errors. If you are using the JIRA WAR/EAR distribution, rebuild and redeploy the webapp in your application server.

User-contributed notes

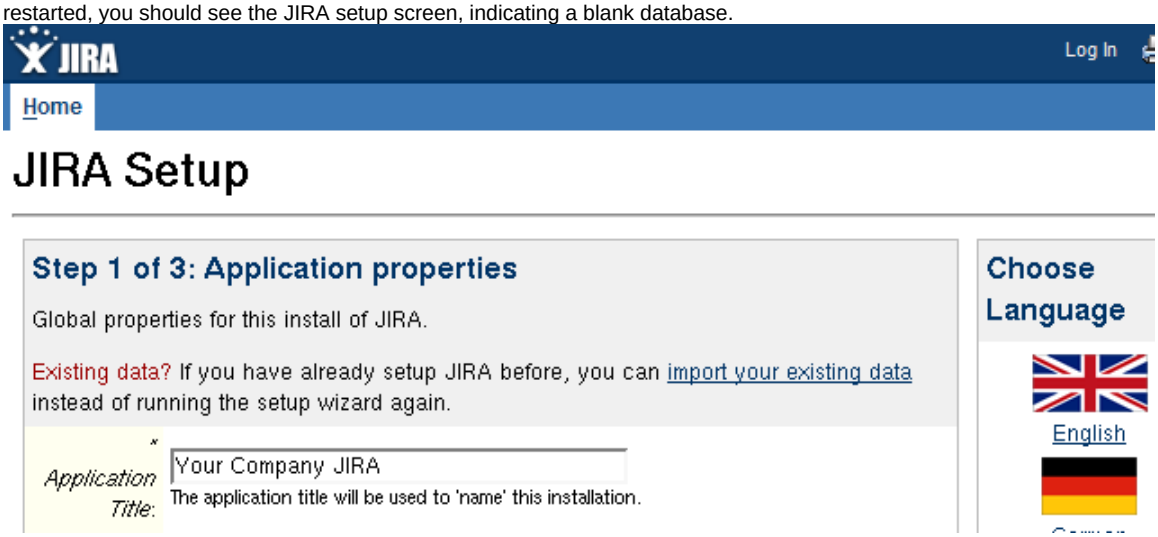
Have experiences to share with HSQL and JIRA? We welcome your thoughts. Please see the user-contributed HSQL notes.

Switching Databases

JIRA data can be migrated between databases.

To do this:

1. Create an export of your data as an XML backup.
2. Alter your JIRA instance to use the new database, following the database-specific steps described in the previous section. When restarted, you should see the JIRA setup screen, indicating a blank database.



3. On the setup page, click `Import your existing data` , and restore your data from the XML backup. Once installed, make sure the index, attachment and backup paths are correct. In particular, if you are running more than one JIRA instance on the same server, ensure that each instance uses different paths.

Upgrading JIRA

This page describes how to upgrade JIRA. Before you start, you should review the release notes and upgrade guide for the version you are upgrading to.

You should also follow these same instructions if you want to upgrade from one edition of JIRA to a higher edition. That is, from JIRA Standard to JIRA Professional or JIRA Enterprise, or from JIRA Professional to JIRA Enterprise.

On this page:

- Overview
- Set up a new instance of JIRA
  - 1. Install the new version of JIRA
  - 2. Configure your new JIRA to have the same configuration as your old JIRA
- Migrating your existing JIRA data into your new JIRA instance
  - 3. Set up a new database
    - 3.1a ("XML backup/restore" method only) Create a new empty database
    - 3.1b ("Connect JIRA to a copy of your old database" method only) Take a copy of your existing database
    - 3.2. Configure JIRA to use your new database
    - 3.3 (JIRA EAR/WAR only) Build the JIRA web application and deploy it to your application server
  - 4. ("XML backup/restore" method only) Export your data from your old JIRA
  - 5. Start the new JIRA
    - 5.1. ("XML backup/restore" method only) Import your old JIRA data into your new JIRA
    - 5.2 Post-startup checks and tasks
- Troubleshooting

If you are upgrading to JIRA 4.0 Beta:
The **only** plugin that is compatible with JIRA 4.0 Beta is the latest JIRA Toolkit. Do not install any other plugins.

Overview

In summary, the method for upgrading JIRA is to set up a new instance of JIRA, and then migrate your existing JIRA data into the new JIRA instance. There are two variations of this method, the difference being in how you choose to migrate your existing JIRA data:

- **XML backup/restore (recommended)** — back up your data as an XML file from your existing JIRA instance, back up your attachments, and then restore that data into the new JIRA instance. The new JIRA will create the database structure (tables and indexes) before importing the data; or,
- **Connect JIRA to a copy of your old database** — take a copy of your existing JIRA database and configure the new JIRA to use the copied database. When you first start up the new JIRA it will upgrade the database to the structure needed for the new JIRA version.

"Which method should I use to migrate my data into my new JIRA?"
We recommend that you use the **"XML backup/restore"** method, as the **"Connect JIRA to a copy of your old database"** method has the following restrictions:
1. JIRA may not be able to automatically update the database if you connect a copy of your old database to your new JIRA, if there have been significant changes to the database scheme between versions. This currently applies for upgrades from **JIRA 3.6.x or older to JIRA 3.7 or later.** See this page for details.
2. You may miss out on any performance improvements that we have added to the database between versions. In particular, if we have added new database indexes between versions, these indexes will not be applied if you connect your new version of JIRA to a copy of your old database.
The "Connect JIRA to a copy of your old database" method can be useful for extremely large installations, where the recommended method of "XML backup/restore" would take too long.
Both of these methods are described in more detail in the 'Migrating your existing JIRA data into your new JIRA instance' section below.

Overall, the process to upgrade JIRA takes five steps (four steps, if you are not performing an XML backup/restore to migrate your existing JIRA data):
Set up a new instance of JIRA:
1. Install the new instance of JIRA.
2. Configure your new JIRA to have the same configuration as your old JIRA.
Migrate your existing JIRA data:
3. Set up a new database.
4. *(If doing an "XML backup/restore" method)* Export your data from your old JIRA.
5. Start the new JIRA *(including the import of your old data into your new JIRA instance, if doing an XML backup/restore).*

"Test first!"
We strongly recommend that you carry out all of the following upgrade instructions in a test environment first. Do not upgrade your production JIRA until you are satisfied that the upgrade has been successful in your test environment. If you have any problems with your test upgrade that you cannot resolve, create a support case in https://support.atlassian.com so that we can help you resolve the problems with the upgrade.
If you have any problems during the upgrade of your production JIRA, do not allow your users to start using the new (upgraded) instance. Instead continue to use your old instance. This will help ensure that you do not lose production data. Create a support case in https://support.atlassian.com so that we can help you resolve the problems with the upgrade.

Set up a new instance of JIRA

If you are upgrading your version or edition of JIRA, you will need to set up a new instance of JIRA that you can migrate your existing data into. This involves installing a new empty instance of the desired version/edition of JIRA and applying configuration changes to match the configuration of your old instance of JIRA.

1. Install the new version of JIRA

Download and unpackage the JIRA distribution you require, JIRA Standalone or JIRA EAR/WAR, to a new directory. **Do not overwrite your old JIRA instance.**

2. Configure your new JIRA to have the same configuration as your old JIRA

You may have modified a number of configuration files in your existing JIRA instance. You need to make the same changes in your new JIRA

instance. However, you cannot simply copy the file from your old instance, as the format may have changed from the old version/edition to the new one. You need to manually edit all of the relevant lines in the new file, applying any changes you previously made in the file in your existing JIRA.

The following files are commonly modified in JIRA installations:

File	JIRA Standalone location	JIRA EAR WAR location	Description
entityengine.xml	`atlassian-jira/WEB-INF/classes`	`edit-webapp/WEB-INF/classes`	Used to configure the JIRA entity engine for your database type. Refer to the **Configure the JIRA Entity Engine** section of your database guide.
osuser.xml	`atlassian-jira/WEB-INF/classes`	`webapp/WEB-INF/classes`	Modified if you are integrating LDAP with JIRA, or if you are using a custom form of external user management or user authentication.
jira-application.properties	`atlassian-jira/WEB-INF/classes`	`webapp/WEB-INF/classes`	Advanced JIRA configuration properties

In addition to the above files, there are several other configuration changes that you may have to migrate:

- **Using JIRA with Atlassian's Crowd** — If you are using Crowd with JIRA, configure your new JIRA to talk to Crowd as described in Integrating Crowd with JIRA.
- **Allocating additional memory to JIRA** — If you had previously allocated additional memory to JIRA, do the same for your new JIRA instance. For more information refer to Increasing JIRA memory.
- **Plugins** — For any plugins that you had installed in your old JIRA:
    1. Download the plugin version for your new version of JIRA from the JIRA Extensions site.
    2. Install the JAR file(s) in your new JIRA, and carry out any other required installation for the plugin.
    3. If the plugin has a properties file, apply the same changes to it as you had in the old properties file (don't just copy over the old properties file).
- **Customisations** — If you had made any customisations (code, templates or configuration files), copy over compatible versions of these changes to the new JIRA. (The developers within your organisation who made the customisations to your old version will need to build and test equivalent changes for the new version, and provide you with the files to copy to your new JIRA).
- **Running JIRA on a different port** — If your new instance of JIRA is installed on the same machine as your old JIRA, make sure it is running on a different port. For JIRA Standalone, refer to Changing JIRA Standalone's Port. For JIRA EAR/WAR, consult your application server documentation.
- **Disabling email access** — If necessary, disable email access in your new JIRA instance, to prevent inadvertent SMTP/POP/IMAP access while testing.
- **Character encoding** — Please ensure that character encoding (ie. locale) is the same on the new and old locations. You may have problem with encoding of the file names, if attachments are moved between two system with incompatible encoding.
- ***(JIRA EAR/WAR only)* Configuring your application server** — If you are running JIRA EAR/WAR, read the latest instructions for your application server and perform any necessary configuration steps. If your application server needs extra libraries to run JIRA, the extra libraries currently in your application server for the old JIRA may be outdated. Make sure you get the extra libraries needed for your new version of JIRA.

Migrating your existing JIRA data into your new JIRA instance

The process for migrating your existing JIRA data into your new JIRA instance generally involves setting up a new/copied database, configuring it to work with your new instance of JIRA and populating it with data, if required. As previously described, you can choose to migrate your existing data by XML backup/restore (recommended) or by connecting JIRA to a copy of your old database.

3. Set up a new database

The first step of migrating your existing data into your new JIRA instance is to set up a new database. If you are using the **XML backup/restore** method, you need to set up a new empty database. If you are using the **Connect JIRA to a copy of your old database** method, then you must take a copy of your old database for the migration. You will then need to configure JIRA to use your new database and if you are using JIRA EAR/WAR, build and deploy the JIRA web application.

3.1a ("XML backup/restore" method only) Create a new empty database

If you are migrating your data to the new JIRA using the **XML backup/restore** method, create a new database (note, you do not need to create a new database if you are using the embedded HSQL database). For example, if you are using a database **'jiradb'** with your existing installation, and you are upgrading to JIRA 3.12, create database **'jiradb_312'** with identical access permissions. Consult with your DBA if you need assistance with this.

3.1b ("Connect JIRA to a copy of your old database" method only) Take a copy of your existing database

"Do not use your existing database for the data migration"
If you are using the **Connect JIRA to a copy of your old database** method, it is **critical** that you take a copy of your existing database for the data migration. Do not just connect the new JIRA to your existing database, as it will be upgraded to the format required by the new JIRA version. Hence, if any problems occur with your upgrade, you will not be able to revert to using your old JIRA.

If you are migrating your data to the new JIRA using the **Connect JIRA to a copy of your old database** method, follow the steps below to create a copy of your existing database:

1. Make your existing JIRA read-only or shutdown your existing JIRA, so that users cannot make changes until the upgrade is complete. If you do not do this, you will lose any updates that users make from this point until the upgrade is complete.
2. Use the vendor's tools for your database to make a copy of your existing JIRA database. Consult with your DBA if you need help to do this.
3. Check your version of JIRA. If you are upgrading from 3.6.5 (or earlier) to 3.7 or later, JIRA will not be able to update your database automatically when you start the new instance (see 'Start the new JIRA' section below). This is due to the large number of database changes between 3.6.5 and 3.7. You will need to manually upgrade your copied database to the 3.7 schema instead.

3.2. Configure JIRA to use your new database

The following instructions describe how to configure JIRA to use your new database. The steps differ depending on whether you are using JIRA Standalone or JIRA EAR/WAR:

- **JIRA Standalone***(please note, these instructions do not apply if you are using the embedded HSQL database with JIRA Standalone. We strongly recommend that you do not to use the HSQL database in production):*# **Reapply the modifications you made to the `conf/server.xml` file in your old JIRA Standalone to get your new JIRA Standalone to use your database, but substitute the new database name (*jiradb_312** in the above example) in the JDBC URL.
    1. Read the latest instructions for your particular database and perform any necessary steps. This will include copying the JDBC driver jar file from the `common/lib` directory of your old JIRA Standalone to your new JIRA Standalone.
- **JIRA EAR/WAR:**
    1. Follow the instructions for your application server and database to make any changes needed to configure the database connection, including copying over the JDBC driver and setting the JDBC URL.

3.3 (JIRA EAR/WAR only) Build the JIRA web application and deploy it to your application server

If you are using **JIRA EAR/WAR,** complete the setup of your new database by building your new JIRA web application and deploying it to your server. You can do this by running either the `build.bat` file (Windows) or the `build.sh` file (Unix). Read your application server guide to see if there are any server-specific instructions for building the web application. For example, there may be parameters you need to pass to the build script.

4. ("XML backup/restore" method only) Export your data from your old JIRA

If you are using the **XML backup/restore** method, you can now export your data from your existing JIRA instance in preparation for the import into your new database. To export your existing JIRA data into a backup file, follow the steps below:

1. Ensure that users cannot update your existing JIRA while you perform the XML backup, so that your backup will not contain inconsistent data. There are two ways that you can do this:
    - **Make JIRA inaccessible:** Shut down JIRA, start it on a different port to the one that JIRA usually runs on and do the backup there. Please note, no-one will be able to access JIRA while it is running on a different port during the backup, even to read issues; or,
    - **Make JIRA read-only:** In your database, remove **'write'** permissions from the database user (e.g. `jirauser`) that JIRA uses. If you do this, users will be able to read issues but will not be able to update them during the backup. For details, please see Making database read-only for upgrade backups. Leave your existing JIRA inaccessible or read-only until the upgrade is completed successfully. You may want to set a banner to warn users that JIRA will be inaccessible/read-only while an upgrade is performed.
2. After you have ensured that users cannot update your existing JIRA, generate the XML backup. Read these instructions on 'Backing up data' for details of how to do this. Please note, you will also need to **back up your attachments** as described in the 'Backing up data' document.
3. Remember to restore JIRA access appropriately when the upgrade is complete.

5. Start the new JIRA

The final step of upgrading your JIRA is to start your new JIRA instance and, if you are using the **XML backup/restore** method, import the data from your old JIRA instance. As previously described, starting your new JIRA instance will automatically update your new/copied database.

1. If you haven't already done so, shut down your old JIRA instance by stopping the JIRA server.
2. Start up your new instance of JIRA,
    - **JIRA Standalone:** Follow the Starting JIRA instructions.

- **JIRA EAR/WAR:** Follow the instructions for starting JIRA for your application server.

Starting up JIRA using Tomcat If you are using Tomcat with your installation of JIRA (note, JIRA Standalone ships with Tomcat by default), delete the Tomcat work directory before restarting JIRA. If you do not do this, users may encounter errors when they try to display JIRA pages.

5.1. ("XML backup/restore" method only) Import your old JIRA data into your new JIRA

If you are migrating your data to the new JIRA using the XML backup/restore method, you will need to import the data from your old instance into your new JIRA instance after you have successfully started it. You will need the backup file of data from your old JIRA that you created earlier in these instructions.

To import your old JIRA data into your new JIRA,

1. Access JIRA via your web browser. You will see the Setup Wizard.
2. Click the **'Import Existing Data'** link.
3. The 'Import Existing Data' page will display.
    - In the **'File name'** field, specify the XML backup file you created previously.
    - In the **'Index Location'** field, specify a different directory from your old JIRA.
4. Restore the attachments directory that you backed up in step 4.2 previously, into the attachments directory of your new JIRA.

5.2 Post-startup checks and tasks

We highly recommend that you perform the following checks and tasks after you have started your new instance of JIRA:

1. Check your server logs for error messages, even if JIRA appears to be running correctly. If there are any errors there that you cannot resolve, create a support case in https://support.atlassian.com, attach your log file and we will advise you on the errors.
2. If you were previously using External User Management, enable it in the new JIRA instance.
3. If you changed machines when upgrading, change the paths to the indexes, attachments and backup directories, from within the **Administration** section of JIRA.
4. Enable email, if you disabled it during testing.
5. If you migrated any customisations from your old JIRA to the new JIRA, ensure that they are tested thoroughly.

Congratulations! You have completed your upgrade of JIRA.

Troubleshooting

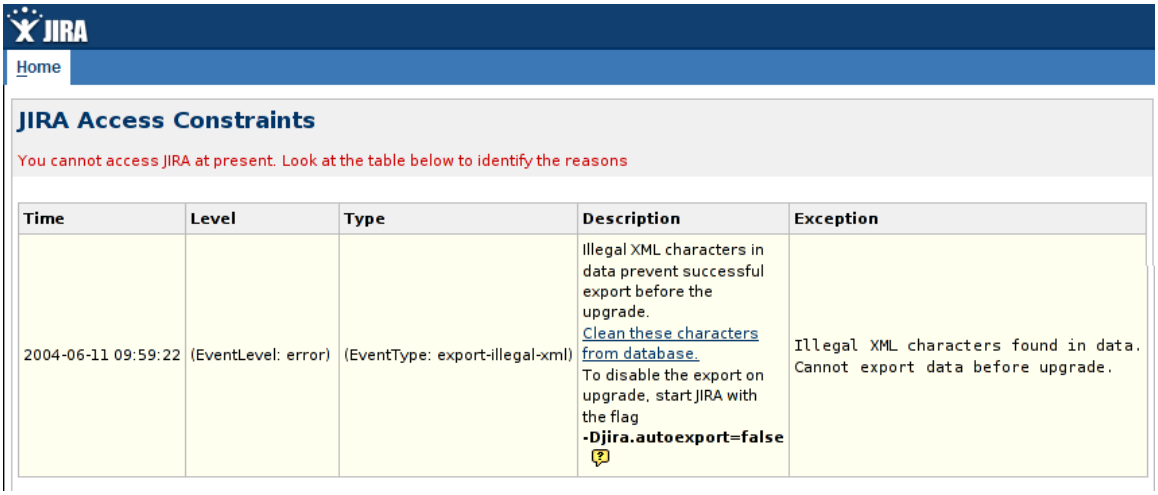
- If the upgrade fails for any reason, revert to using your old JIRA. Create a support case in https://support.atlassian.com and attach your log file, so that we can help you resolve the problems with the upgrade.
- If your XML import fails with an error message indicating an invalid XML character, use our XML cleaner utility.

Disabling Auto-Export

When upgrading JIRA, one points the new JIRA installation at the old JIRA database. JIRA will automatically make any structural database modifications required to support new JIRA features.

To be safe, JIRA first tries to create an XML backup of your data at the point just before the upgrade. This would allow you to 'roll back' to the old JIRA version, should anything go wrong.

Sometimes the automatic XML backup procedure fails, often because of characters in the database that cannot be represented in XML (typically non-displayable control characters that have been cut-and-pasted in).



In these circumstances, you can force the upgrade to proceed by editing `atlassian-jira/WEB-INF/classes/jira-application.properties`, and setting **jira.autoexport=false.**

Once you have upgraded JIRA, successfully, it is best to remove this parameter, as it is no longer needed.

If you have any upgrade problems not covered here or in the upgrade documentation, please contact us — we're happy to help.

Important Directories and Files

On this page:

- JIRA Installation Directory
  - Important Files and Directories
    - `atlassian-jira/WEB-INF/classes/jira-application.properties`
    - `atlassian-jira/WEB-INF/classes/osuser.xml`
    - `atlassian-jira/WEB-INF/lib/`
    - `atlassian-jira/WEB-INF/classes/log4j.properties`
    - `conf/server.xml`
    - Memory Settings
- JIRA Home Directory
  - Important Directories
    - `data`
    - `export`
    - `plugins`
    - `caches`
    - `log`
    - `tmp`

JIRA Installation Directory

The 'JIRA installation directory' is the directory into which the JIRA application files and libraries have been unpacked (unzipped) when JIRA was installed. JIRA does not modify or store any data in this directory.

Important Files and Directories

`atlassian-jira/WEB-INF/classes/jira-application.properties`

This file tells JIRA where to find the JIRA Home Directory, as well as storing many other configuration settings . This file is modified by the administrator when Installing JIRA Standalone (not using Windows installer) or Installing JIRA WAR-EAR.

`atlassian-jira/WEB-INF/classes/osuser.xml`

This file is modified when connecting JIRA to an external user management system such as an LDAP server or Atlassian's Crowd. For more information, refer to User and Group Management.

`atlassian-jira/WEB-INF/lib/`

This is the directory where plugins built on Atlassian's Plugin Framework 1 (i.e. 'Plugins 1' plugins) are stored. If you are installing a new 'Plugins

1' plugin, you will need to deploy it into this directory.
'Plugins 2' plugins should be stored in the JIRA Home Directory.

`atlassian-jira/WEB-INF/classes/log4j.properties`

JIRA's logging configuration file. See Logging and Profiling.

The actual log files generated by JIRA can be found in the following locations:

- **JIRA application log** — `bin/atlassian-jira.log`
- **Application server log** — generally the application server log file can be found under the `logs` directory. However, this can vary depending on the application server you are running. Please see Where are the JIRA logs? for further details.

`conf/server.xml`

This file is used for JIRA SSL configuration. See Running JIRA over SSL or HTTPS.

Memory Settings

The file used to edit JAVA_OPTS memory settings will depend on the method used to install JIRA, as well as the operating system used for your installation.

For example, if you are running JIRA on Tomcat in Windows (manual startup), you would update the following file:
`bin\setenv.bat`
whereas for JIRA on Tomcat in Unix, you would update this file:
`bin/setenv.sh`

See Increasing JIRA Memory for further details.

JIRA Home Directory

The 'JIRA home directory' contains key data that help define how JIRA works. This document outlines the purpose of the various files and directories in the JIRA home directory. (For information on specifying the location of the JIRA home directory, please see Setting your JIRA Home Directory .)

Tip: *Another term for 'home directory' would be 'data directory'.*

Important Directories

`data`

This directory contains application data for your JIRA instance, including:

- attachments (every version of each attachment stored in JIRA) — note, your JIRA administrator can define a custom attachments directory.

`export`

JIRA will place its automated backup archives in this directory. Note, your JIRA administrator can define a custom backups directory.

This does not affect manually generated backups, i.e. you must specify the target location for a manually generated backup.

`plugins`

This is the directory where plugins built on Atlassian's Plugin Framework 2 (i.e. 'Plugins 2' plugins) are stored. If you are installing a new 'Plugins 2' plugin, you will need to deploy it into this directory.

'Plugins 1' plugins should be stored in the JIRA Installation Directory.

This directory is created on JIRA startup, if it does not exist already.

`caches`

This is where JIRA stores caches including:

- Lucene indexes - see Indexing in JIRA
- OSGi framework caches

These files are vital for JIRA performance and should not be modified or removed externally while JIRA is running.

Note: *If you are running a high load JIRA instance, you may wish to consider moving your indexes from the default directory under JIRA home to another machine. The directory which holds your indexes is heavily accessed while JIRA is running. Hence, your JIRA instance may run more efficiently by hosting the indexes on the fastest disk available and reducing the number of other applications using the same disk to reduce access contention.*

See Search Indexing for further details.

tmp

Any temporary content created for various runtime functions such as exporting, importing, file upload and indexing is stored under this directory.

You can remove files from this directory while JIRA is running, but we recommend that you shut down JIRA first before altering the contents of this directory.

JIRA Home Directory

The 'JIRA home directory' contains key data that help define how JIRA works. This document outlines the purpose of the various files and directories in the JIRA home directory. (For information on specifying the location of the JIRA home directory, please see Setting your JIRA Home Directory .)

Tip: *Another term for 'home directory' would be 'data directory'.*

Important Directories

`data`

This directory contains application data for your JIRA instance, including:

- attachments (every version of each attachment stored in JIRA) — note, your JIRA administrator can define a custom attachments directory.

`export`

JIRA will place its automated backup archives in this directory. Note, your JIRA administrator can define a custom backups directory.

This does not affect manually generated backups, i.e. you must specify the target location for a manually generated backup.

`plugins`

This is the directory where plugins built on Atlassian's Plugin Framework 2 (i.e. 'Plugins 2' plugins) are stored. If you are installing a new 'Plugins 2' plugin, you will need to deploy it into this directory.

'Plugins 1' plugins should be stored in the JIRA Installation Directory.

This directory is created on JIRA startup, if it does not exist already.

`caches`

This is where JIRA stores caches including:

- Lucene indexes - see Indexing in JIRA
- OSGi framework caches

These files are vital for JIRA performance and should not be modified or removed externally while JIRA is running.

Note: *If you are running a high load JIRA instance, you may wish to consider moving your indexes from the default directory under JIRA home to another machine. The directory which holds your indexes is heavily accessed while JIRA is running. Hence, your JIRA instance may run more efficiently by hosting the indexes on the fastest disk available and reducing the number of other applications using the same disk to reduce access contention.*

See Search Indexing for further details.

`tmp`

Any temporary content created for various runtime functions such as exporting, importing, file upload and indexing is stored under this directory.

You can remove files from this directory while JIRA is running, but we recommend that you shut down JIRA first before altering the contents of

this directory.

JIRA Installation Directory

The 'JIRA installation directory' is the directory into which the JIRA application files and libraries have been unpacked (unzipped) when JIRA was installed. JIRA does not modify or store any data in this directory.

Important Files and Directories

`atlassian-jira/WEB-INF/classes/jira-application.properties`

This file tells JIRA where to find the JIRA Home Directory, as well as storing many other configuration settings . This file is modified by the administrator when Installing JIRA Standalone (not using Windows installer) or Installing JIRA WAR-EAR.

`atlassian-jira/WEB-INF/classes/osuser.xml`

This file is modified when connecting JIRA to an external user management system such as an LDAP server or Atlassian's Crowd. For more information, refer to User and Group Management.

`atlassian-jira/WEB-INF/lib/`

This is the directory where plugins built on Atlassian's Plugin Framework 1 (i.e. 'Plugins 1' plugins) are stored. If you are installing a new 'Plugins 1' plugin, you will need to deploy it into this directory.
'Plugins 2' plugins should be stored in the JIRA Home Directory.

`atlassian-jira/WEB-INF/classes/log4j.properties`

JIRA's logging configuration file. See Logging and Profiling.

The actual log files generated by JIRA can be found in the following locations:

- **JIRA application log** — `bin/atlassian-jira.log`
- **Application server log** — generally the application server log file can be found under the `logs` directory. However, this can vary depending on the application server you are running. Please see Where are the JIRA logs? for further details.

`conf/server.xml`

This file is used for JIRA SSL configuration. See Running JIRA over SSL or HTTPS.

Memory Settings

The file used to edit JAVA_OPTS memory settings will depend on the method used to install JIRA, as well as the operating system used for your installation.

For example, if you are running JIRA on Tomcat in Windows (manual startup), you would update the following file:
`bin\setenv.bat`
whereas for JIRA on Tomcat in Unix, you would update this file:
`bin/setenv.sh`

See Increasing JIRA Memory for further details.

Setting your JIRA Home Directory

The JIRA Home Directory contains key data that help define how JIRA works. You must have a JIRA home directory specified for your JIRA instance before you can start it. This document describes how to set up the JIRA home directory for your JIRA instance.

> ⚠ **One JIRA home per JIRA instance**
> You can only have one JIRA home per JIRA instance. If you have multiple JIRA instances, you will need to set up a JIRA home for each instance. A lock is placed on the root level of a JIRA home when created, to ensure that it is only used by one JIRA instance. If you have accidentally **locked your JIRA home directory**, you can unlock it by following the instructions in this FAQ.

You only need to specify the location of the root directory for your JIRA home. The sub-directories will be created automatically when JIRA is started or when you use a function in JIRA that requires a particular sub-directory.

How do I set my JIRA home?

The recommended way to specify the location of your JIRA Home Directory is to:

- Edit the `jira-application.properties` file in the JIRA Installation Directory, add a 'jira.home' property and set it to your desired location for the JIRA home directory. Please use forward-slashes ("/"), not back-slashes ("\").

> **Note:**
>
> - If you are using the Windows installer, you don't need to do this as you will be prompted to specify the location for your JIRA home during installation..
> - If you are using JIRA WAR-EAR, you need to set your JIRA home *before* you build JIRA.

Alternatively, you can choose one of the following methods of specifying the location of your JIRA Home Directory:

- Add a web context property called '`jira.home`' — this property is set in different files depending on your application server. For example, for Tomcat (and therefore for JIRA Standalone), you will need to configure the server.xml file. For other application servers you may need to configure the web.xml file, or set 'Context parameter' options on the deployment UI of the application server, etc. Note that If you have specified a JIRA home in `jira-application.properties` (ie. the recommended method), it will override your web context property.
- Add the following property to the command line when starting JIRA: `-Djira.home=`***<home-directory>***
  where <home-directory> is the desired location of the root directory for your JIRA home. If you have specified a JIRA home in `jira-application.properties` (ie. the recommended method), or have set a web context property called '`jira.home`', they will override this command-line parameter.

What location should I specify for my JIRA home?

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend that you do not specify your JIRA home directory inside your installation directory, to prevent information from being accidentally lost during major operations (e.g. backing up and restoring instances).

How do I change my JIRA home?

To change the location of your JIRA home directory,

1. Set your JIRA home to the new location, using your preferred method as described in *"How do I set my JIRA home?"* (above).
2. Restart JIRA.

JIRA Releases

Latest Production Releases

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

Please view the release notes to get up-to-date information about the improvements made in each release. If upgrading from a previous version of JIRA please pay attention to the *Upgrade Guide* of the latest version and any version of JIRA that your are 'skipping' during the upgrade.

Release Summary

The features of each JIRA release, up to and including the latest version, can be found in the JIRA Release Summary.

For full details on each of the JIRA releases, please read the release notes for the previous releases listed below.

Previous Releases

See the complete list of Releases Notes and Upgrade Guides for information about older releases.

Production Releases

This page lists release notes and upgrade guides from past versions of JIRA.

If upgrading from a previous version of JIRA please pay attention to the *Upgrade Guide* of the version you are upgrading to, and any version of

JIRA that you are 'skipping' during the upgrade.

You can also view lists of the Release Notes or Upgrade Guides separately.

- All JIRA Upgrade Guides (version 3.x and later)
  - Aggregated JIRA 3.x Upgrade Guides
- All JIRA Release Notes (version 3.x and later)
- JIRA 3.13 Release Notes — This release fulfils some of the most popular JIRA feature requests. Dashboards can now be shared, and filter sharing has been improved — so it's easy to set up multiple 'template' dashboards, each with specific portlets and filters. New JIRA users can then simply select the dashboards most suited to them.
  - JIRA 3.13.5 Release Notes — The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.5** in Standard, Professional and Enterprise editions. This point release includes over **30** bug fixes and improvements.
    - JIRA 3.13.5 Upgrade Guide
  - JIRA 3.13.4 Release Notes — The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.4** in Standard, Professional and Enterprise editions. This point release includes over **20** bug fixes and improvements.
    - JIRA 3.13.4 Upgrade Guide
  - JIRA 3.13.3 Release Notes — The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.3** in Standard, Professional and Enterprise editions. This point release includes over **85** bug fixes and improvements, including an important security fix — please see JIRA Security Advisory 2009-04-02 for details.
    - JIRA 3.13.3 Upgrade Guide
  - JIRA 3.13.2 Release Notes
    - JIRA 3.13.2 Upgrade Guide
  - JIRA 3.13.1 Release Notes
    - JIRA 3.13.1 Upgrade Guide
  - JIRA 3.13 Upgrade Guide
- JIRA 3.12 Release Notes — JIRA 3.12 provides a number of enhancements for the upcoming JIRA Studio. Because some of these enhancements may be of benefit to you, we have decided to provide them as a public release rather than making you wait until JIRA 4.0.
  - JIRA 3.12.3 Release Notes
    - JIRA 3.12.3 Upgrade Guide
  - JIRA 3.12.2 Release Notes
    - JIRA 3.12.2 Upgrade Guide
  - JIRA 3.12.1 Release Notes
    - JIRA 3.12.1 Upgrade Guide
  - JIRA 3.12 Upgrade Guide
    - JIRA 3.12 DB Schema Changes
- JIRA 3.11 Release Notes — This release focuses on time-tracking. Time-tracking data (that is, the estimated and actual time spent on an issue) now includes the issue's sub-tasks. The aggregated time-tracking data is displayed both within individual 'parent' issues and in the Issue Navigator, so it can be easily reported on, exported to Excel, etc.
  - JIRA 3.11 Upgrade Guide
- JIRA 3.10 Release Notes — Editable worklogs; new ways to browse Components and Versions.
  - JIRA 3.10.2 Release Notes
    - JIRA 3.10.2 Upgrade Guide
  - JIRA 3.10.1 Release Notes
    - JIRA 3.10.1 Upgrade Guide
  - JIRA 3.10 Upgrade Guide
- JIRA 3.9 Release Notes — Convert issues to sub-tasks, and sub-tasks to issues. Use the convenient new scheduler to subscribe to issue filters.
  - JIRA 3.9.3 Release Notes — Professional French and German translations
    - JIRA 3.9.3 Upgrade Guide
  - JIRA 3.9.2 Release Notes
    - JIRA 3.9.2 Upgrade Guide
  - JIRA 3.9.1 Release Notes
    - Important Security Patch for JIRA versions 3.7.x & 3.8.x
    - JIRA 3.9.1 Upgrade Guide
  - JIRA 3.9 Upgrade Guide
- JIRA 3.8 Release Notes — Editable Comments, Self-installer for Windows, CAPTCHA for new account signup, Integration with Crowd, Improvements to the Bugzilla importer
  - JIRA 3.8.1 Release Notes
    - JIRA 3.8.1 Upgrade Guide
  - Feedback for DHTML-loading of Issue screens
  - JIRA 3.8 Upgrade Guide
    - JIRA 3.8 Database Schema Changes
- JIRA 3.7 Release Notes — Project Roles, 'Charting' View for Issue Navigator, RSS Improvements, User Properties, SVN Project Panel plugin, SVN Commit Acceptance plugin
  - JIRA 3.7.4 Release Notes
    - JIRA 3.7.4 Upgrade Guide
  - JIRA 3.7.3 Release Notes

        - JIRA 3.7.3 Upgrade Guide
    - JIRA 3.7.2 Release Notes
        - JIRA 3.7.2 Upgrade Guide
    - JIRA 3.7.1 Release Notes
        - JIRA 3.7.1 Upgrade guide
    - Issue Operations plugin
    - JIRA 3.7 Upgrade Guide
        - ActionManager Removed
        - JIRA 3.7 Database Schema Changes
        - Possible upgrade problems + solutions
        - SQL Scripts for 3.6.x to 3.7 schema upgrade
- JIRA 3.6 Release Notes — Custom Events. Group Picker Custom Field, Wiki-Style Linking, Expandable Fields
    - JIRA 3.6.5 Release Notes
        - JIRA 3.6.5 Upgrade guide
    - JIRA 3.6.4 Release Notes
        - JIRA 3.6.4 Upgrade guide
    - JIRA 3.6.3 Release Notes
        - JIRA 3.6.3 Upgrade guide
    - JIRA 3.6.1 Release Notes
        - JIRA 3.6.1 Upgrade Guide
    - JIRA 3.6.2 Release Notes
        - JIRA 3.6.2 Upgrade guide
    - JIRA 3.6 Upgrade Guide
- JIRA 3.5 Release Notes — Bulk Workflow Transition, a new Charting Plugin, MS Word Export, JIRA Page Linker Plugin, Component Lead Notification Type
    - JIRA 3.5.3 Release Notes
        - JIRA 3.5.3 Upgrade Guide
    - JIRA 3.5.2 Release Notes
        - JIRA 3.5.2 Upgrade Guide
    - JIRA 3.5.1 Release Notes
        - JIRA 3.5.1 Upgrade Guide
    - JIRA 3.5 Upgrade Guide
- JIRA 3.4 and 3.4.1 Release Notes — Issue by project, renderers, clone portlets, issue operation plugin, improvements and bug fixes.
    - JIRA 3.4.3 Release Notes
        - JIRA 3.4.3 Upgrade Guide
    - JIRA 3.4.2 Release Notes
        - JIRA 3.4.2 Upgrade Guide
    - JIRA 3.4.1 Release Notes
    - JIRA 3.4 and 3.4.1 Upgrade Guide
        - Upgrading Custom Field Types in JIRA 3.4.1
        - Upgrading Custom Field Types in JIRA 3.4
- JIRA 3.3 Release Notes — Create filters with multiple projects, improved date search, bulk move and performance improvements.
    - JIRA 3.3.3 Release Notes
        - JIRA 3.3.3 Upgrade Guide
    - JIRA 3.3.2 Release Notes
    - JIRA 3.3.1 Release Notes
        - JIRA 3.3.1 Upgrade Guide
            - JIRA 3.3.1 Issue Tab Panel extension
    - JIRA 3.3 Upgrade Guide
        - Upgrading to JIRA 3.3 Standalone
        - Parameter changes in Issue Navigator
        - Upgrading custom CustomFieldTypes in JIRA 3.3
- JIRA 3.2 Release Notes — Fields can be configured on a per-screen basis, screens can contain tabs, more flexible custom fields, and performance improvements.
    - JIRA 3.2.3 Release Notes
    - JIRA 3.2.2 Release Notes
    - JIRA 3.2.1 Release Notes
    - 3.2 performance benchmarks
    - JIRA 3.2 Upgrade Guide
        - Notifications no longer sent to raw email addresses if anonymous browsing disabled
        - Restricting Edit based on Issue Status
        - Using Oracle 10g drivers to solve the 4000 character limitation
        - Upgrading custom CustomFieldTypes in JIRA 3.2
        - Workflows using default "Closed" status — h2. Affected users
        - Upgrading Workflow Plugins for JIRA 3.2
- JIRA 3.1 Release Notes — CSV import wizard, workflow action keyboard shortcuts, assign issues by mail and performance improvements.
    - JIRA 3.1.1 Release Notes
    - JIRA 3.1 Upgrade Notes
- JIRA 3.0 Release Notes — Configurable workflow with editor, sub-tasks, plugin system, custom fields improvements, clone issue, change reporter, multiple attachments, introduction of the standard edition.
    - JIRA 3.0 Upgrade Notes
    - JIRA 3.0.3 Release Notes
    - JIRA 3.0.2 Release Notes
    - JIRA 3.0.1 Release Notes

- JIRA 2.6 Release Notes

All JIRA Release Notes (version 3.x and later)

This page lists the release notes from JIRA versions 3.x and later.

You may be interested in the list of upgrade guides from JIRA versions 3.x and later.

JIRA 4.0 Beta 2 Release Notes

JIRA 3.13.5 Release Notes

JIRA 4.0 Beta 1 Release Notes

JIRA 3.13.4 Release Notes

JIRA 3.13.3 Release Notes

JIRA 3.13.2 Release Notes

JIRA 3.13.1 Release Notes

JIRA 3.13 Release Notes

JIRA 3.12.3 Release Notes

JIRA 3.12.2 Release Notes

JIRA 3.12.1 Release Notes

JIRA 3.12 Release Notes

JIRA 3.10.2 Release Notes

JIRA 3.11 Release Notes

JIRA 3.10.1 Release Notes

JIRA 3.9.3 Release Notes

JIRA 3.9.2 Release Notes

JIRA 3.9.1 Release Notes

JIRA 3.10 Release Notes

JIRA 3.9 Release Notes

JIRA 3.8.1 Release Notes

JIRA 3.8 Release Notes

JIRA 3.7.4 Release Notes

JIRA 3.7.3 Release Notes

JIRA 3.7.2 Release Notes

JIRA 3.7.1 Release Notes

JIRA 3.6.5 Release Notes

JIRA 3.6.4 Release Notes

JIRA 3.7 Release Notes

JIRA 3.6.3 Release Notes

- JIRA 3.6.1 Release Notes
- JIRA 3.6.2 Release Notes
- JIRA 3.6 Release Notes
- JIRA 3.5.3 Release Notes
- JIRA 3.5.2 Release Notes
- JIRA 3.5.1 Release Notes
- JIRA 3.5 Release Notes
- JIRA 3.4.1 Release Notes
- JIRA 3.4 and 3.4.1 Release Notes
- JIRA 3.3.3 Release Notes
- JIRA 3.3.2 Release Notes
- JIRA 3.3.1 Release Notes
- JIRA 3.3 Release Notes
- JIRA 3.2.3 Release Notes
- JIRA 3.2.2 Release Notes
- JIRA 3.2.1 Release Notes
- JIRA 3.2 Release Notes
- JIRA 3.1.1 Release Notes
- JIRA 3.1 Release Notes
- JIRA 3.0.3 Release Notes
- JIRA 3.0.2 Release Notes
- JIRA 3.0.1 Release Notes
- JIRA 3.0 Release Notes

All JIRA Upgrade Guides (version 3.x and later)

This page lists the upgrade guides from JIRA versions 3.x and later. You can also view the aggregated upgrade guides from JIRA versions 3.x and later (***warning:*** *the aggregated upgrade guides page is big!*).

If upgrading from a previous version of JIRA please pay attention to the *Upgrade Guide* of the version you are upgrading to, and any version of JIRA that you are 'skipping' during the upgrade.

You may be interested in the list of release notes from JIRA versions 3.x and later.

- JIRA 3.13.5 Upgrade Guide
- JIRA 3.13.4 Upgrade Guide
- JIRA 3.13.3 Upgrade Guide
- JIRA 3.13.2 Upgrade Guide
- JIRA 3.13.1 Upgrade Guide

- JIRA 4.0 Upgrade Guide
- JIRA 3.13 Upgrade Guide
- JIRA 3.12.3 Upgrade Guide
- JIRA 3.12.2 Upgrade Guide
- JIRA 3.12.1 Upgrade Guide
- JIRA 3.12 Upgrade Guide
- JIRA 3.10.2 Upgrade Guide
- JIRA 3.10.1 Upgrade Guide
- JIRA 3.11 Upgrade Guide
- JIRA 3.9.3 Upgrade Guide
- JIRA 3.9.2 Upgrade Guide
- JIRA 3.9.1 Upgrade Guide
- JIRA 3.10 Upgrade Guide
- JIRA 3.9 Upgrade Guide
- JIRA 3.8.1 Upgrade Guide
- JIRA 3.8 Upgrade Guide
- JIRA 3.7.4 Upgrade Guide
- JIRA 3.7.3 Upgrade Guide
- JIRA 3.7.2 Upgrade Guide
- JIRA 3.7.1 Upgrade guide
- JIRA 3.6.5 Upgrade guide
- JIRA 3.6.4 Upgrade guide
- JIRA 3.7 Upgrade Guide
- JIRA 3.6.3 Upgrade guide
- JIRA 3.6.1 Upgrade Guide
- JIRA 3.6.2 Upgrade guide
- JIRA 3.5.3 Upgrade Guide
- JIRA 3.5.2 Upgrade Guide
- JIRA 3.6 Upgrade Guide
- JIRA 3.5.1 Upgrade Guide
- JIRA 3.5 Upgrade Guide
- JIRA 3.4.3 Upgrade Guide
- JIRA 3.4.2 Upgrade Guide
- JIRA 3.4 and 3.4.1 Upgrade Guide

JIRA 3.3.3 Upgrade Guide

JIRA 3.3.1 Upgrade Guide

JIRA 3.3 Upgrade Guide

JIRA 3.2 Upgrade Guide

JIRA 3.1 Upgrade Notes

JIRA 3.0 Upgrade Notes

Aggregated JIRA 3.x Upgrade Guides

This page contains a live aggregate of all JIRA upgrade guides since version 3. You can also view the lists of Release Notes or Upgrade Guides for JIRA.

JIRA 2.x to 3

This page lists a few things to be aware of when upgrading from previous releases of JIRA to JIRA 3. To perform the actual upgrade, see the upgrade documentation.

Existing SMTP Mail Server 'From' address may break notifications (JRA-5089)

In JIRA 3, email notification 'From' addresses now contain the reporter name, eg. "Joe Bloggs (JIRA) <jira@company.com>", where "jira@company.com" is set by the admin as the SMTP mail server From address. If you have this address to already include a name (eg "Tech Support <jira@company.com>", then email notifications will fail with errors like:

```
2005-01-06 11:30:53,856 ERROR [atlassian.mail.queue.MailQueueImpl]
com.atlassian.mail.MailException: Sending failed;
  nested exception is: javax.mail.internet.AddressException:
Missing '<' in string ``"Joe Bloggs (JIRA)" <Tech Support <jira@company.com>>'' at position 62
```

Fix

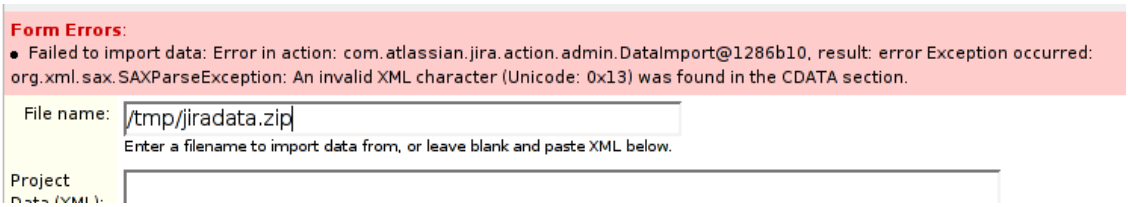
The fix is to edit `WEB-INF/classes/jira-application.properties`, and change the following property value to `false`:

```
jira.option.include.user.in.mail.from.address = true
```

- If using JIRA Standalone, the file is *atlassian-jira/WEB-INF/classes/jira-application.properties*, after which you should run bin/shutdown and bin/startup to restart.
- If using JIRA deployed as a webapp, copy *webapp/WEB-INF/classes/jira-application.properties* to *edit-webapp/WEB-IFN/classes*, make the change to the *edit-webapp* copy, run *build* to rebuild the webapp, and redeploy it on your app server.

Invalid characters break XML import

JIRA's recommended upgrade process involves deploying an XML backup of your data. Some users will find that the import fails with this error:



This is usually because the database contains control characters that cannot be represented in Unicode, and hence XML.

Fix

The fix is to follow these instructions to remove the invalid characters from the XML before import.

JIRA 3.0 to 3.1

This page lists a few things to be aware of when upgrading from JIRA 3.0.x to JIRA 3.1. To perform the actual upgrade, see the upgrade documentation. For upgrading from JIRA 2.x to JIRA 3.x see JIRA 3.0 Upgrade Notes

MySQL Users dB upgrade (JRA-5635)

The size of the descriptor field in the jiraworkflow table has been increased. MySQL users will see warnings when they start their app server. This can be fixed by running the SQL below. This will also allow for Workflows of up to 4GB as opposed to just 64k

```
alter table jiraworkflows change DESCRIPTOR DESCRIPTOR LONGTEXT;
```

JIRA 3.1 to 3.2

This page contains information you need to know when upgrading to JIRA 3.2. The general upgrade instructions can be found here.

1. If you have written any Custom Field Type plugins please refer to this document
2. If you have created any Workflow plugins (custom Validators or Post Functions) please read this document.
3. If you have any custom file based workflows (workflows *not* created through JIRA's Workflow Editor) please read this document.
4. If you wish issues that are associated with the default system workflow and are closed to be bulk editable - please read this .

Notifications now respect permissions

In 3.2, JIRA respects the permission scheme and security levels when sending notifications (see JRA-5743. People who won't be able to see an update online won't get a notification email.

This has one important effect: if you have a project where:

- the notification scheme specifies that a raw email address (eg. developers@mycompany.com) should be notified, and
- 'Browse' permission has not been granted to 'Anyone' (eg. it is granted to 'jira-users'
  then that email address ('developers@mycompany.com' in our example) won't be mailed. As JIRA cannot verify that the recipient(s) of the email address have the 'browse' permission, it makes the conservative assumption that they are not.

This can be fixed by creating a user (eg. 'developers') for the email address, making it a member of a group that has 'Browse' permission, and adding it as a recipient of notifications. The raw email address should then be removed from the notification scheme, as it serves no purpose.

JIRA 3.2 to 3.3

JIRA 3.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.3 from JIRA 3.2.x. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Known incompatibilities

3.3.x is not a good release for IBM shops:

1. JIRA 3.3.x may not work on **Websphere 5.0.x and 5.1.x** due to JRA-7699
2. When using DB2, JIRA may hang when deleting projects or performing workflow operations. See the full problem description (and possible workaround) in the documentation

Websphere or DB2 users, please stick with 3.2.x or move on to 3.4.x or higher, where these problems have been resolved.

Notes on upgrading

1. Due to web browser caches, changes to JIRA's Issue Navigator might appear corrupted or unstyled. Please refresh your browser's cache (press *Shift+Reload* on the Find Issue's page) for the changes to appear correctly.
2. JIRA's **issue cache** size will be automatically set to 0 during the upgrade, as it is no longer needed due to performance improvements in JIRA (JRA-7166)
3. If you have written any CustomFieldType or CustomFieldSearcher plugins please refer to this document
4. Users with **outgoing** trackback pings enabled (not the default) may wish to disable this until JRA-7589 is fixed, to avoid the risk of the mail queue hanging.
5. If you have bookmarks or deal with hard coded links to the issue navigator, you should read about the changed issue navigator parameters
6. If you are **using JIRA Standalone**, please do *not* simply copy your old `conf/server.xml` file to the new installation of JIRA. Please read this document.
7. If upgrading JIRA in an external Tomcat installation, be sure to delete the `work/` temporary directory before restarting JIRA, to clear cached JSPs from the old JIRA.

JIRA 3.3 to 3.3.x

JIRA 3.3.1 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.3.1 from JIRA 3.3.

> If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below:

1. If you have implemented a cutom Issue Tab Panel plugin you need to be aware of this API change.

> If you are upgrading to JIRA 3.3.1 from a previous version, due to web browser caches, changes to JIRA's Issue Navigator might appear corrupted or unstyled. Please refresh your browser's cache (press Shift+Reload on the Find Issue's page) for the changes to appear correctly.

JIRA 3.3.x to 3.4.x

JIRA 3.4 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4 from JIRA 3.3.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

> Two major new features of JIRA 3.4, wiki renderer previews, and issue types per project require that javascript be enabled to make use of their full functionality. You will still be able to use all the core features of JIRA with javascript disabled.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

1. Please do **not** copy `jira-application.properties` file from your old JIRA installation. Edit the file that is shipped with JIRA 3.4 and make needed changes. New properties have been added to this file so if you simply copy the old file across the following error would occur JRA-8645.
2. If you have written any CustomFieldType or CustomFieldSearcher plugins please refer to Upgrading Custom Field Types in JIRA 3.4
3. The default user preferences are now configured in the jira-application.properties file and are configurable through the admin section of JIRA. Any properties in the old file preferences-default.xml will no longer effect JIRA configuration.
4. Please note that to configure issue types per project you must have JavaScript turned on in your web browser.
5. If you are using MySQL please do not use Connector/J 3.1.11 JDBC Driver as it has the following bug. Connector/J 3.1.10 and earlier work fine.

JIRA 3.4.1 Upgrade Guide

This section contains specific information you need to know when upgrading to JIRA 3.4.1 from JIRA 3.4. If upgrading from JIRA 3.3.3 please read the previous section as well. If upgrading from an older version than JIRA 3.3.3, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

1. Please do **not** copy `jira-application.properties` file from your old JIRA installation. Edit the file that is shipped with JIRA 3.4 and make needed changes. New properties have been added to this file so if you simply copy the old file across the following error would occur JRA-8645.
2. If you have written a CustomFieldType that implements the `com.atlassian.jira.issue.customfields.CustomFieldType` interface directly rather than extending one of the Abstract classes that ship with JIRA please read Upgrading Custom Field Types in JIRA 3.4.1.
3. If you have written an Custom Field Searcher please have a look at Upgrading Custom Field Types in JIRA 3.4.1.
4. JIRA 3.4 and 3.4.1 do not generate an Issue Assigned event. The Issue Updated event is generated instead. In previous versions of JIRA the Issue Assigned event was generated when issues are assigned using the "Assign" operation on the View Issue page. This means that even when the "Assign" operation is used JIRA will send notifications to parties listed under the Issue Updated event. The patch to correct this behaviour is available at JRA-8533.

JIRA 3.4.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4.2 from JIRA 3.4.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.4.1 to JIRA 3.4.2.

JIRA 3.4.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4.3 from JIRA 3.4.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.4.2 to JIRA 3.4.3.

JIRA 3.4.x to 3.5.x

JIRA 3.5 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5 (release notes) from JIRA 3.4.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

JIRA 3.5 Jira Service extension

- If you have implemented a custom JIRA service you need to be aware of the following API change.

In JIRA 3.5 the `getName()` and `setName(String name)` methods was added to the `com.atlassian.jira.service.JiraService` interface. This method should return and set the name of the service respectively. The name of the service can be used to identify a service uniquely. (Fixed made due to JRA-8352 bug)

Therefore, if you have implemented this interface, you will need to implement these methods and recompile your service(s) before deploying it into JIRA 3.5. If you have extended a JIRA class instead, e.g. `com.atlassian.jira.service.AbstractService` or `com.atlassian.jira.service.JiraServiceContainer` you do not need to modify your custom services.

Introduction of global **Bulk Change** permission

JIRA 3.5 introduces the global **Bulk Change** permission. This permission governs the ability to execute the bulk change operations:

- Workflow Transition
- Edit
- Move
- Delete

An upgrade task has been added to grant the new **Bulk Change** permission to all groups with the global **JIRA Users** permission.

The JIRA documentation includes further details on this new permission.

The decision to grant the **Bulk Change** permission should be considered carefully - the permission permits a user to modify a collection of accessible issues at once. For example, in JIRA installations configured to run in 'Public' mode (anybody can sign up and create issues), a user could comment on all accessible issues with the **Bulk Change** and **Add Comments** permission. Undoing such modifications may not be possible through the JIRA UI and may require changes made directly against the database.

CustomFieldPersister changes

CustomFieldPersister is used to store custom field values to database. The methods of this class has been refactored to remove the redundant parameter, `defaultValueMarker`. For example, the create values method went from:

```
void createValues(CustomField field, Long issueId, String defaultValueMarker,
PersistenceFieldType persistenceFieldType, Collection values, String parentKey);
```

to:

```
void createValues(CustomField field, Long issueId, PersistenceFieldType persistenceFieldType,
Collection values, String parentKey);
```

You will need to update and recompile any `CustomFieldType` that you wrote to use this new interface.

VersionCFType Changes

This affects plugin writers who uses the version custom field `VersionCFType`. The change is that previously the Transport Object type was a single `Version` object, but it is now a collection that contains a single `Version` object.

This was done to handle an improved version custom field which can be a mutli-select version custom field as well

JIRA 3.5.1 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.1 from JIRA 3.5. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.5 to JIRA 3.5.1.

JIRA 3.5.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.2 from JIRA 3.5.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Issue Event Changelog Can Now Be Null

- If you have implemented a custom JIRA Issue Event Listener you need to be aware of the following API change.

In JIRA 3.5.2, the IssueEvent object thrown as a result of an edit operation, **may now return null** from a `getChangeLog()` call. The case where this happens is when a user chooses to edit an issue but only leaves a comment and makes no other changes to the issue. Prior to 3.5.2 no event was fired in this case and this was identified as a bug (JRA-9415) and has since been fixed. Check any calls to getChangeLog() for null.

JIRA 3.5.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.3 from JIRA 3.5.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.5.3 from JIRA 3.5.2.

JIRA 3.5.x to 3.6.x

JIRA 3.6 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.x from JIRA 3.5.x. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Database Intensive Upgrade Task

To introduce the Custom events to JIRA, it was necessary to upgrade a large data set within JIRA's database for 3.5.x and earlier releases. Depending on the size of your JIRA data the upgrade task (number 150) might get your DBMS to do a lot of work which might take some time. The exact amount of time also depends on the processing power of the machine running JIRA's database.

Please be patient with the upgrade task and do not restart JIRA while the upgrade is in progress. The upgrade task will report on its progress to JIRA's log file as it upgrades your data.

The following is the sample output that the upgrade task will produce. As you can see the upgrade task took roughly 5 and a half minutes to modify over 660,000 records in the database.

```
11:14:09 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Phone Support
Workflow v.6'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow
v.3'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Phone Support
Workflow v.7'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Test'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Copy of Support
Workflow'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow
v.4'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow'.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150]
************************************************************
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating 660453 records in the
'NotificationInstance' table.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] This might take a long time. Please do
NOT stop JIRA.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150]
************************************************************
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_CREATED'.
11:15:12 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_UPDATED'.
11:15:51 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_ASSIGNED'.
11:16:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_RESOLVED'.
11:16:46 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_CLOSED'.
11:16:57 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_COMMENTED'.
11:18:57 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_REOPENED'.
11:19:17 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_DELETED'.
11:19:26 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_MOVED'.
11:19:31 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKLOGGED'.
11:19:37 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKSTARTED'.
11:19:41 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKSTOPPED'.
11:19:43 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_GENERICEVENT'.
11:21:23 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Update of 'NotificationInstance' records
finished.
```

Workflow Post Functions

Applies to	users with custom workflow XMLs saved on disk - external to JIRA

JIRA stores its workflows in the database. During the upgrade, these workflows will be upgraded automatically. However, if you have stored your workflows on disk (outside the database), you will need to follow these instructions to upgrade the workflows manually.

Previously, workflow post functions referenced the event to fire through a string value of the event name. All post functions now reference the event through a numeric ID value. As mentioned, all workflows stored within JIRA will be automatically updated. However, all workflows saved to disk - external to JIRA - should be updated manually as follows. The actual workflow XML file should be updated as follows:

For each workflow post function that accepts the event ID as an argument:

1. The value of the **name** attribute of the **arg** tag has to be changed from **eventType** to **eventTypeId**
2. The body of the **arg** tag has to change according to the following table:

Event Name	Event Type Id
created	1
updated	2
assigned	3
resolved	4
closed	5
commented	6
reopened	7
deleted	8
moved	9
worklogged	10
workstarted	11
workstopped	12
genericevent	13

By default, the only post functions that accept event IDs are *FireIssueEventFunctions.* Therefore, unless you have implemented your own custom post function that also deals with events, you will only need to update the arg tags for the *FireIssueEventFunctions* everywhere in the workflows.

For example, *FireIssueEventFunction* for create issue workflow transition looked like:

```
<function type="class">
    <arg name="class.name"
>com.atlassian.jira.workflow.function.event.FireIssueEventFunction</arg>
     <arg name="eventType">created</arg>
</function>
```

and needs to be changed to:

```
<function type="class">
    <arg name="class.name"
>com.atlassian.jira.workflow.function.event.FireIssueEventFunction</arg>
     <arg name="eventTypeId">1</arg>
</function>
```

Custom Events

Applies to	users who have **modified JIRA source code or added custom code to define new notification events**. Also of interest to users wishing to define new notification templates

Releases before JIRA 3.6 did not allow users create custom events. If you have modified the JIRA source to add custom events - please follow these instructions.

If you have previously defined a custom event within JIRA - it is necessary to add appropriate entries to the following files:

- `system-event-types.xml`- used to install and upgrade all event types within the system to the new 3.6 event type object.
- `email-template-id-mappings.xml` - maps the event id to an assocaited velocity template file.

The `system-event-types.xml` file requires name and description details of the previously added custom event. For example, if the custom event type "Issue Frozen" was added to the system - the following entry should be added to the XML file:

```
<eventtype id="10000">
   <name>Issue Frozen</name>
   <description>This is the 'Issue Frozen' event type.</description>
   <notificationName>ISSUE_FROZEN</notificationName>
   <eventName>issuefrozen</eventName>
</eventtype>
```

The elements provide the following information:

- id - the new id for the event type. **All custom event types should be added from ID 10000 and above**
- notificationName - the original name for the event as found in the Notification table
- eventName - the origianl name for the event as found in workflows

The `email-template-id-mappings.xml` file requires an entry mapping the new custom event to an associated velocity email template. This mapping is used when a notification is sent for this event. Following from the above example, the following entry would be made:

```
<templatemapping id="10000">
   <name>Issue Frozen</name>
   <template>issuefrozen.vm</template>
</templatemapping>
```

The id should match that of the event as specified in the `system-event-types.xml` The `template` entity should reference the Velocity template to be used in email notifications of this event. A HTML and text version should be provided in the appropriate directory (html or text) at:

`<JIRA>/src/etc/java/templates/email/`

All custom event types added to the file *system-event-types.xml* should be added with an ID of 10000 and above

Custom Listeners

Applies to	users who have **added custom listeners** to JIRA.

For all users who have **added custom written** listeners to JIRA, it might be necessary to update the listener to follow the new JIRA 3.6 API.

There are two things to look out for:

1. signature change of the *workflowEvent* method
2. change of return type of *getIssue()* method on the *IssueEvent* object

The signature of the method *workflowEvent* in the *IssueEventListener* has changed from:

```
public void workflowEvent(int type, IssueEvent event);
```

to:

```
public void workflowEvent(IssueEvent event);
```

Note: the type parameter has been removed.

If you have implemented *IssueEventListener* directly or have extended *AbstractIssueEventListener* and have overridden the method *workflowEvent*, you will need to change and recompile your listener before installing JIRA 3.6.

In JIRA 3.6, the event type ID can be retrieved by calling the following method on the *IssueEvent* object:

```
Long eventID = event.getId();
```

However, the returned value of the *getId()* method is different to the values of the type parameter that was passed to the *workflowEvent* method. The following table represents these differences:

Event Name	Old ID	New ID
created	0	1
updated	1	2
assigned	2	3
resolved	3	4
closed	4	5
commented	5	6
reopened	6	7
deleted	7	8
moved	8	9
worklogged	9	10
workstarted	10	11
workstopped	11	12
genericevent	-1	13

Also, the *getIssue()* method of the *IssueEvent* object has changed to return an Issue object instead of a *GenericValue* object representing an issue.

Users who have created and added custom listeners must update the listener to now operate with the *Issue* object. For example:

```
Issue issueObject = event.getIssue();
```

As a quick fix, you can modify your listener to use `event.getIssue().getGenericValue()`.

The event type ID constants are now only available from the class *EventType*. Any use of the original constants must be updated to use the *EventType* constants. For listeners that reference an event ID by its numeric value - it is necessary to ensure that the IDs now match those as defined in *EventType*.

Custom permission types

Applies to	users who have modified JIRA source to add new permission types (ie. in addition to the standard 'user', 'group', 'assignee' types).

The SecurityType interface, used to implement permission types ('single user', 'group' etc) has had a `getUsers()` method added. If you have implemented your own SecurityType you will need to implement this. See the source of current implementations (eg. `GroupCF`) for tips.

Plugin upgrades required

As usual, you should check whether the plugins you use are compatible with the new release. Generally, plugins (like the Subversion plugin or JIRA toolkit) need to be upgraded when JIRA is upgraded. See the list of plugins at:

http://confluence.atlassian.com/display/JIRAEXT/Home

This page contains specific information you need to know when upgrading to JIRA 3.6.1 from JIRA 3.6. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.6.1 from JIRA 3.6.

JIRA 3.6.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.2 from JIRA 3.6.1. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Maximum Active Databased Connections

Applies to	JIRA Standalone users

In version of JIRA before 3.6.2, the maximum number of database connections was limited to 8 by default. If JIRA was used by more than 8 concurrent users or under very haeavy usages, the users could experience delays or JIRA could hang.

In JIRA 3.6.2 the default number of maximum active database connections has been increased to 20. When upgrading to JIRA 3.6.2, please ensure that your database will allow JIRA to establish 20 connections, or decrease this number to desired value. To adjust the number of connections change the value of the `maxActive` attribute of the `jdbc/JiraDS` resource in `config/server.xml` file. JIRA has to be restarted to apply the change.

JIRA 3.6.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.3 from JIRA 3.6.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.6.3 from JIRA 3.6.2.

JIRA 3.6.4 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.4 from JIRA 3.6.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading to JIRA 3.6.4 from JIRA 3.6.3.

JIRA 3.6.5 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.5 from JIRA 3.6.4. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading to JIRA 3.6.5 from JIRA 3.6.4.

JIRA 3.6.x to 3.7.x

Once you have upgraded to JIRA 3.7, downgrading to a previous version is not a straightforward task and is not recommended.

JIRA 3.7 Upgrade Notes

This page lists a few things to be aware of when upgrading from previous releases of JIRA to JIRA 3.7. To perform the actual upgrade, see the upgrade documentation.

Note: If you are upgrading from a pre-3.6.5 release, please also refer to the relevant JIRA 3.x Upgrade Guides.

Please note that JIRA 3.7 requires JDK 1.4 or above. Support for JDK 1.3 has been discontinued.

Please note that some new functionality will not be available if you are running JIRA on WebLogic or Orion. The List All Filtersportlet will not be able to fetch the issue counts for each issue. The new 'Charting' View will also be unavailable. The support for WebLogic and Orion will be added in JIRA 3.7.1.

Database Schema Changes

Due to the upgrade of HSQLDB, and to improve compatibility with Firebird and Frontbase, various database tables and columns have been renamed. For more details on the changes please see the JIRA 3.7 Database Schema Changes document.

Therefore, the easiest way to upgrade to JIRA 3.7 is to follow the Upgrading JIRA safely instructions.

If in the past, instead of performing an XML backup and restore, you have been upgrading by "pointing" new version of JIRA at an old database, this is still possible, however the procedure is more complicated. You will need to use SQL scripts to perform database schema changes. For more information please see the SQL Scripts for 3.6.x to 3.7 schema upgrade document.

If you are using HSQLDB with JIRA, you must follow Upgrading JIRA safely instructions (i.e. perform a full XML backup and restore from XML), as simply copying the `.script` file will not work. The format of the `.script` file has changed between the HSQLDB versions, and therefore, copying the `.script` file will result in the following error on startup.

Request Context Changes

In order for plugins, customfields and portlets to function better outside of a web-context (e.g.: displaying a customfield in an e-mail), all direct references to the HttpServletRequest have been replaced by a VelocityRequestContext. If you have deployed your own plugins, customfields or portlets that use the HttpServletRequest directly (i.e.: any references to `${req}`) than they should be changed to use the new `${requestContext}` object. The `${requestContext}` is an implementation of the `VelocityRequestContext` interface.

Currently the `${requestContext}` supports the following properties:

- `${requestContext.baseUrl}` - Returns the same as HttpServletRequest.getContextPath() or the base URL configured in your JIRA instance if no HttpServletRequest is available
- `${requestContext.requestParameters}` - Returns an implementation of `RequestContextParameterHolder` or null if no HttpServletRequest is available
- `${requestContext.requestParameters.servletPath}` - Returns the same as HttpServletRequest.getServletPath()
- `${requestContext.requestParameters.requestURL}` - Returns the same as HttpServletRequest.getRequestURL()
- `${requestContext.requestParameters.queryString}` - Returns the same as HttpServletRequest.getQueryString()

Integrity Checks

In JIRA 3.7 Database Integrity Checks (available from the Administration section) have been re-written to run as multiple transactions, which increased the throughput of the system while the checks are running. In large JIRA 3.6 (and earlier) installations, integrity checks could cause database lock escalation and stop users from performing operations (e.g. viewing issues).

Please note, that due to the change, each integrity check became about 10% slower.

As integrity checks are quite database intensive operations, it is still recommended to run them during off-peak hours (i.e. while the system is not under heavy load).

Change of commentLevel to groupLevel in the Comment and TransitionWorkflow jelly tags

We have changed the AddComment and TransitionWorkflow jelly tag attribute that specifies the group visibility level from 'commentLevel' to be 'groupLevel'. If you have existing jelly tags that use this attribute it will need to change. This was done so that we could introduce the 'roleLevel' attribute which allows you to specify a project role based visibility. Only one of the two attributes can be specified at a time.

Change of level to grouplevel in the XML view of a Comment

1. We have changed the XML view of a comment, as seen in the XML view of an Issue to contain either a 'grouplevel' attribute or a 'rolelevel' attribute. This attribute defines the visibility level specified on the comment. In the past the 'grouplevel' attribute was simply 'level'. If you have any existing custom code that expects the 'level' attribute in the Comment XML it must change to expect 'grouplevel'.
2. In previous versions of JIRA the XML view of the <comment> tag level attribute was always shown, even if there was no value for the level, it was rendered as an empty attribute. We have changed it so that the attributes themselves (grouplevel and rolelevel) do not display if there is no value.

Change to the RemoteComment object used via SOAP/RPC plugin

The RemoteComment object and therefore the remote SOAP/RPC api has changes to almost all properties. The 'roleLevel' attribute was added and the following attributes have changed:

1. level -> groupLevel
2. datePerformed -> created
3. username -> author

ActionManager removed

The `ActionManager` interface has been removed and its functionality has been delegated to new interfaces. For details please refer to ActionManager Removed documentation

Removal of 'Backend Actions'

1. We have removed the 'Backend Action' `com.atlassian.jira.action.action.WorklogCreate` if you were using this class in a plugin or custom code you will now need to use the `com.atlassian.jira.issue.worklog.WorklogManager` this now has method calls to return worklogs for a given user+issue and also create worklog entries.
2. We have removed the 'Backend Action' `com.atlassian.jira.action.action.ActionCreate` if you were using this class to create comments you will need to modify your code to use one of the create methods on the com.atlassian.jira.bc.issue.comment.CommentService

Issue Events

We have modified the `com.atlassian.jira.event.issue.IssueEvent` class to no longer use GenericValues. The GenericValue representing the comment is replaced by `com.atlassian.jira.issue.comments.Comment` class and the GenericValue representing the worklog is replaced by `com.atlassian.jira.issue.worklog.Worklog` class. If you have a custom listener in a previous version of JIRA this will need to be updated to use the newer IssueEvent class and `com.atlassian.jira.event.issue.IssueEventDispatcher.dispatchEvent(...)` methods.

Renaming of XML export file

By popular request, the XML filename (that is, the default filename when you choose to save the XML view in the Issue Navigator) has been changed from `issuenavigator.jspa` to `SearchRequest.xml`. Should you have any external systems or programs that utilise the exported XML file, please be aware of the changed filename.

Confluence Users Only - Pre 2.2.10 Confluence Must Be Patched To Use JIRA Issues Macro
Unable to render {include} Couldn't find a page to include called: DOC:JIRA 3.7 Link Format Change

JIRA 3.7 Downgrade Notes

Once you have upgraded to JIRA 3.7, downgrading to a previous version is not a straightforward task and is not recommended. Please be aware that in JIRA 3.7 the database schema has changed.

If upgrade to JIRA 3.7 fails, the best way to proceed is to go back to the previous version of JIRA you were using, and to the latest **pre-upgrade** data that you have. The exact steps for doing this depend on how you have upgraded JIRA.

If you have created a new database for JIRA 3.7 by following the Upgrading JIRA safely instructions, you should be able to simply shutdown JIRA 3.7 and bring up the old version of JIRA your were using. The old version should be configured to use its old (unupgraded) database.

If you have upgraded JIRA by pointing JIRA 3.7 to an older database (and ran the SQL Scripts to upgrade the database schema), then you will need to:

1. Create a new database
2. Configure the old version of JIRA you were using to point at the new (empty) database
3. Restore the latest **pre-upgrade** backup that you have
4. Start the old JIRA installation

JIRA 3.7.1 Upgrade Guide

This page contains specific information you need to know when upgrading from JIRA 3.7 to JIRA 3.7.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.7 to JIRA 3.7.1.

JIRA 3.7.2 Upgrade Guide

This page contains specific information you need to know when upgrading from JIRA 3.7.1 to JIRA 3.7.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- 3.7.2 will automatically perform a full reindex when upgrading. For more details please see JRA-11861

Upgrading from JIRA 3.7.2 to 3.7.3

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.7.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.7.3 to 3.7.4

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.7.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.7.x to 3.8.x

Upgrading from JIRA 3.7.4 to 3.8

Please follow the JIRA general upgrade instructions. Additionally, please note the following:

1. The 'Assign To' field name has been changed to 'Assignee' consistently across JIRA. This means that users need to be aware that the column heading in the Excel export has changed to 'Assignee' from 'Assign To'. Please be aware of this if for example you are exporting JIRA data to Excel and running macros on it. The field has been renamed for the following Issue Navigator Views:
    - Excel (all)
    - Word (all)
    - Full Content
2. The `issuecommentedited.vm` e-mail template for the new Issue Comment Edited event has been added to the `WEB-INF/classes/email-template-id-mappings.xml` file. The id of the e-mail template used for sending Filter Subscriptions has changed to 10000. **If** you have manually modified the `WEB-INF/classes/email-template-id-mappings.xml` file in the version of JIRA you are upgrading from, please do not simply copy the old file to JIRA 3.8. You will need to port your changes to the `WEB-INF/classes/email-template-id-mappings.xml` file that is shipped with JIRA 3.8. If you have not changed the `WEB-INF/classes/email-template-id-mappings.xml` file, you do not need to worry about this.
3. Two columns have been added to the `jiraaction` table to support editable comments.

Upgrading from JIRA 3.7.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.8 to 3.8.1

Please follow the JIRA general upgrade instructions.

Charting Plugin must be upgraded to v1.3.5

Please note that the version of JFreeChart included in JIRA 3.8.1 is **not compatible** with older versions of the Charting Plugin. If you have the Charting Plugin installed, please make sure you upgrade it to version 1.3.5 or above.

The updated JFreeChart 1.0.4 version is **not** backwards compatible with the previous 1.0.0pre2 version, so if you have any plugins that utilise JFreeChart, please make sure you test them before upgrading.

Upgrading from JIRA 3.7.4 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.8.x to 3.9.x

Upgrading from JIRA 3.8.1 to 3.9

Please follow the JIRA general upgrade instructions. Additionally, please note the following:

In this version, there has been a change to the database which may cause problems for some customers.

The Recommended Upgrade Method

If you follow the recommended export/import upgrade procedure you should not experience any problems!

Pointing JIRA 3.9 at an existing, non-empty database

Some customers have a good reason for not following the recommended upgrade method. Using this method may result in database errors in your logs. You can avoid this if you modify your table structure manually, but the procedure is different depending on whether you have already started JIRA.

To avoid this, **BEFORE** you upgrade JIRA using this method, you can just drop the qrtz_cron_triggers table. This table has not been used by JIRA before 3.9, so it should be empty.

If you have **ALREADY** started JIRA 3.9 using your existing database, you may see the following log messages when JIRA starts up:

```
2007-04-18 15:31:53,345 main WARN [core.entity.jdbc.DatabaseUtil] Column  "CRON_EXPERSSION" of
table "public.qrtz_cron_triggers" of entity  "QRTZCronTriggers" exists in the database but has
no corresponding field
2007-04-18 15:31:53,347 main WARN [core.entity.jdbc.DatabaseUtil] Entity  "QRTZCronTriggers" has
3 fields but table "public.qrtz_cron_triggers"  has 4 columns.
```

The reason for this is that we have incorrectly changed a column in the `qrtz_cron_triggers` table. The intention was to fix a misspelling, but all we did was remove an underscore ("_")! The old column name is "CRON_EXPERSSION". The new column name is "CRONEXPERSSION". Note that both columns spell the word "expression" incorrectly.

To remove the error message, you must remove the old column as it is redundant. This column will not contain any data. The following table shows all columns in the qrtz_cron_triggers table. Columns that should be present are in **green** and columns that should be deleted are in **red**.

Keep	Keep	Keep	Delete
ID	TRIGGER_ID	CRONEXPERSSION	CRON_EXPERSSION

To delete the column, you can use SQL, but this may be slightly different between databases. Here's how it might look:

```
alter table qrtz_cron_triggers drop column CRON_EXPERSSION;
```

The data in this table

If you have users who have subscribed to issue filters, note that existing SimpleTriggers (time intervals) will be automatically converted into CronTriggers during the JIRA upgrade. In some cases, there may not be an exact mapping of time intervals to Cron Expressions, and approximations will be made (e.g. 'Every 5 weeks' will be converted to 'Once a month'). If this happens, the JIRA upgrade process will send an email to the user to inform them of the new schedule.

Upgrading from JIRA 3.8 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.9 to 3.9.1

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.8.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.9/3.9.1 to 3.9.2

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.8.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Upgrading from JIRA 3.9.2 to 3.9.3

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.9.x to 3.10.x

Upgrading from JIRA 3.9.3 to 3.10

Please follow the JIRA general upgrade instructions, plus note the following:

1. Plugins

There is a new version of the JIRA Calendar Plugin that links to the new 'Project Version' pages. This new version of the plugin is not backwards compatible.

Please note that the Kaamelot plugin for JIRA has not yet been updated. If you are currently using this plugin, you may want to hold off the upgrade to JIRA 3.10 until a compatible version of this plugin has been released.

2. Developer Notes

The ordering of the ListOrderedMap returned by SchemePermissions.getSchemePermissions() has changed. This also means that the order of the RemotePermission[] array returned by the RPC Plugin's JiraSoapService.getAllPermissions() method has changed. If you have extended your instance of JIRA please confirm that any remote applications retrieving permissions via SOAP still work. You may encounter problems if you have been retrieving specific permissions by their array index.

Database changes

In JIRA 3.10, the worklog records have moved from the 'jiraactions' database table to the new 'worklog' table. This new table contains the following columns:

```
Table "public.worklog"
    Column    |           Type           | Modifiers
--------------+--------------------------+-----------
 id           | numeric(18,0)            | not null
 issueid      | numeric(18,0)            |
 author       | character varying(255)   |
 grouplevel   | character varying(255)   |
 rolelevel    | numeric(18,0)            |
 worklogbody  | text                     |
 created      | timestamp with time zone |
 updateauthor | character varying(255)   |
 updated      | timestamp with time zone |
 startdate    | timestamp with time zone |
 timeworked   | numeric(18,0)            |
```

Upgrading from JIRA 3.9.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.10 to 3.10.1

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.10.1 to 3.10.2

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.10.x to 3.11.x

- Upgrading from JIRA 3.10.x to 3.11
- Upgrading from JIRA 3.9.x and earlier

Upgrading from JIRA 3.10.x to 3.11

Please follow the JIRA general upgrade instructions, plus note the following:

Administrative notes

- To take advantage of the performance enhancements in JIRA 3.11, it is recommended that you enable GZip compression (unless you are using mod_proxy).
- The jira-application.properties file has a new option, 'progress', for the following attribute:

  ```
  jira.table.cols.subtasks
  ```

  The 'progress' option controls the display of the '**Progress**' field in issues and reports.

- JIRA 3.11 introduces a bug fix for JRA-12354. This means that the CVS and Perforce plugin will perform better at detecting commits for a particular issue key, avoiding partial matches on similar project keys. If users have taken advantage of the previous relaxed key matching, they can revert to the old behaviour by simply setting the following application property in the jira-application.properties file and restarting JIRA:

  ```
  jira.option.key.detection.backwards.compatible=true
  ```

Plugins

Updating plugins
If you are using any of the following plugins, you will need to update them to their latest versions when performing the upgrade:

- Perforce plugin
- Subversion plugin
- Toolkit Plugin
- Charting Plugin
- RPC Plugin

3rd Party and personal plugins may also be affected (esp. if using lucene to store dates). These will need to be updated as well.

If these are updated after the upgrade (instead of as part of the upgrade), you will need to do a reindex.

A failure to update these plugins will result in lots of errors that look like:

Error 1

```
2007-07-25 15:23:27,553 http-8090-Processor4 ERROR
[charting.charts.createdvsresolved.CreatedVsResolvedChart] Could not create velocity parameters
For input string: "20070725144811"
java.lang.NumberFormatException: For input string: "20070725144811"
 at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
 at java.lang.Long.parseLong(Long.java:415)
 at org.apache.lucene.document.DateField.stringToTime(DateField.java:100)
 at org.apache.lucene.document.DateField.stringToDate(DateField.java:104)
 at
com.atlassian.jira.ext.charting.data.DatePeriodStatisticsMapper.getValueFromLuceneField(DatePeriodS
at
com.atlassian.jira.ext.charting.data.OneDimensionalObjectHitCollector.adjustMapForValues(OneDimensi
at
com.atlassian.jira.ext.charting.data.OneDimensionalObjectHitCollector.collect(OneDimensionalObjectH
at org.apache.lucene.search.IndexSearcher$1.collect(IndexSearcher.java:137)
 at org.apache.lucene.search.Scorer.score(Scorer.java:49)
 at org.apache.lucene.search.IndexSearcher.search(IndexSearcher.java:146)
 at org.apache.lucene.search.Searcher.search(Searcher.java:118)
 at
com.atlassian.jira.issue.search.providers.LuceneSearchProvider.search(LuceneSearchProvider.java:111
```

Error 2

```
Caused by: java.lang.NoSuchMethodError:
org.apache.lucene.document.Document.add(Lorg/apache/lucene/document/Field;)V
 at com.atlassian.jira.plugin.labels.LabelSearcher.index(LabelSearcher.java:95)
 at
com.atlassian.jira.issue.index.indexers.impl.DefaultCustomFieldIndexer.addIndex(DefaultCustomFieldI
at com.atlassian.jira.issue.index.IssueDocument.getDocument(IssueDocument.java:34)
 at
com.atlassian.jira.issue.index.IssueDocumentBuilderImpl.get(IssueDocumentBuilderImpl.java:14)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$IssueAndCommentCreator.handleIssueIndexir
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$AbstractIssueAndCommentHandler.indexIssue
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer.indexIssuesAndComments(SingleThreadedIssu
at
com.atlassian.jira.issue.index.MultiThreadedIssueIndexer.indexIssuesAndComments(MultiThreadedIssueI
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$2.perform(SingleThreadedIssueIndexer.java
at
com.atlassian.bonnie.ConcurrentLuceneConnection.withWriter(ConcurrentLuceneConnection.java:296)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$1.perform(SingleThreadedIssueIndexer.java
at
com.atlassian.bonnie.ConcurrentLuceneConnection.withWriter(ConcurrentLuceneConnection.java:296)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer.indexIssues(SingleThreadedIssueIndexer.ja
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$6.perform(SingleThreadedIssueIndexer.java
```

If you see these errors, please ensure that you are using the latest compatible version of the plugin for 3.11. If there is no supported version for 3.11, please contact the plugin developer via the plugin's homepage.

Developer notes

Modification to SOAP clients
If you have written a SOAP client for any JIRA version prior to 3.11 and are invoking any methods to get `RemoteIssueType` you will encounter the bug JRA-13529. The reason for this is that we have added extra information to the `RemoteIssueType` object that indicates if the issue type is a subTask issue type. To avoid the problem you will need to regenerate your remote object stubs against the updated JIRA 3.11 wsdl.

If you would like your SOAP client to work against multiple versions of JIRA then you need to use the latest stubs that have been generated against JIRA 3.11. You will need to **not** use any of the new functionality and you will need to remember that the `isSubTask` variable in the `RemoteIssueType` objects will be defaulted to `false`.

ThreadLocalQueryProfiler searchers have been moved to ThreadLocalSearcherCache
There may be a number of plugins that reference the ThreadLocalQueryProfiler searcher methods directly. These need to now reference the ThreadLocalSearcherCache.

Lucene Upgrade
We upgraded our version of Lucene to 2.2. If your plugin uses to Lucene to index/read data, please ensure that it works with JIRA 3.11. If you are indexing/reading dates, more than likely it will have broken and you will need to use the new Lucene 2 methods.

Database changes

There were no database changes in this release.

Upgrading from JIRA 3.9.x and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.11 to 3.12.x

Upgrading from JIRA 3.11 to 3.12

Please follow the JIRA general upgrade instructions, plus note the following:

1. Everyone who had the '**JIRA Administrators'** global permission before the upgrade will automatically receive the new '**JIRA System Administrators**' global permission during the upgrade. This will ensure that everyone can still perform the same functions they could previously.
2. The following new Seraph property can be used to fix JRA-10508:

```
<!--  If this parameter is set to true, the cookie will never be set secure.  This is
useful if you're logging
              into JIRA via https, but want to browse JIRA over http.  This flag will
ensure that the remember me option
              works correctly. -->
        <init-param>
            <param-name>insecure.cookie</param-name>
            <param-value>true</param-value>
        </init-param>
```

3. Due to the Seraph upgrade, to fix JRA-10508 all users will be prompted to log in again. This will also affect users who have the 'Remember me' checkbox ticked.
4. If you are building JIRA from source, please note that Maven2 is now required for a build. This is because the JIRA Fisheye Plugin requires Maven2.
5. If you are using the JIRA Toolkit, it is recommended that you upgrade to the latest version in order to fix JRA-13553
6. Please note that the new Trusted Applications feature is not supported on Orion versions prior to 2.0.5. Also note that Resin2 has problems and you will need to update the Resin extra jars.
7. There is a new database table. Please see the following page for details

Upgrading from JIRA 3.10.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Using the Trusted Applications feature with Crowd

Please note that older versions of the Crowd client, (i.e. version 1.2.1 or earlier), can interfere with the correct operation of the Trusted Applications feature. If you are enabling Trusted Applications and using Crowd, please ensure that your Crowd client is **version 1.2.2 or later**.

Upgrading from JIRA 3.12 to 3.12.1

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.12.1 to 3.12.2

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.12.2 to 3.12.3

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Upgrading from JIRA 3.12.xx to 3.13

Please follow the JIRA general upgrade instructions, plus note the following:

1. Introduction of Favourite Dashboards and Filters

Favourite Dashboards

JIRA 3.13 introduces the favourite dashboards feature, which allows you to add dashboard pages that are owned by you or shared by other users as favourites (and hence, are displayed as tabs on your dashboard). On upgrade to JIRA 3.13, all your dashboard pages will be added as your favourites and displayed on your dashboard. If you do not wish any of your dashboards to be added as favourites, then you can remove them as favourites after the upgrade. See the dashboards documentation for details.

Favourite Filters

Similar to favourite dashboards, JIRA 3.13 introduces the favourite filters feature, which allows you to add issue filters that are owned by you or shared by other users as favourites. On upgrade to JIRA 3.13, all your issue filters will be added as your favourites. If you do not wish any of your filters to be added as favourites, then you can remove them as favourites after the upgrade. See the issue filters documentation for details.

please note, this change will not affect issue filter sharing, e.g. if you are using a shared issue filter in one of your dashboard portlets, it will still be shared with you after the upgrade.

please also note, that any custom developed portlets (or other JIRA objects that use filters that have been developed by 3rd parties) that have a dropdown list (not a pop-up picker) for filters, will now only show a list of the user's favourite filters, instead of all shared filters.

Favourite Filters portlet

The 'List All Filters' portlet has been replaced with the 'Favourite Filters' portlet in this release. Your dashboard will be automatically upgraded if it is currently configured to display the 'List All Filters' portlet.

2. Tomcat, MySQL database connection dropouts

Please note, if you wish to use a MySQL database with JIRA Standalone you must set up the bundled Tomcat server (version 5.5.26) to survive connection closures. You must also do this if you are running JIRA EAR/WAR in Tomcat 5.5.25 or later, or Tomcat 6.0.13 or later. Versions 5.5.25 and above of Tomcat 5, and versions 6.0.13 and above of Tomcat 6, have been noted to exhibit problems maintaining connections to MySQL databases. Please read this document for details on the changes required.

3. Changes to jira-application.properties

jira.subscription.email.max.issues property

The `jira.subscription.email.max.issues` property has been added to the `jira-application.properties` file. This property allows you to specify the maximum number of issues that can be included in an email subscription. The default value for this property is 200. You may wish to update this property after the upgrade if you wish to set a different limit on the number of issues that can be included in an email subscription. See the documentation on Advanced JIRA configuration with jira-application.properties for further details on this file.

4. Support for Portlet Plugins with JSP Views Discontinued

Portlet plugins with JSP views are no longer supported. If you have written a custom Portlet plugin and have used a JSP as the view template, you will need to convert your JSP to Velocity.

5. Updates to JIRA SOAP and XML-RPC APIs

com.atlassian.jira.rpc.soap.JiraSoapService

- replaced

```
RemoteProject[] getProjects(String token) throws RemoteException;
```

  with

```
RemoteProject[] getProjectsNoSchemes(String token) throws RemoteException
```

*You should use **getProjectsNoSchemes()** instead because it much more memory efficient and quicker.*

- added

```
RemoteProject getProjectWithSchemesById(String token, Long projectId)
throws RemoteException;
```

- deprecated

```
RemoteFilter[] getSavedFilters(String token) throws RemoteException;
```

- added

```
RemoteFilter[] getFavouriteFilters(String token) throws RemoteException;
```

com.atlassian.jira.rpc.xmlrpc.XmlRpcService

- replaced

```
Vector getProjects(String token) throws Exception;
```

  with

```
Vector getProjectsNoSchemes(String token) throws Exception;
```

- deprecated

```
Vector getSavedFilters(String token) throws Exception;
```

- added

```
Vector getFavouriteFilters(String token) throws Exception;
```

6. Crowd Cache Timeout

This is only applicable if you are using Crowd.

The default timeout for caching user details has changed from 5 minutes to 2 hours. This will improve the performance of the application but will mean that it will take longer for changes to user details to reach the application. Details on how to configure the Crowd caches can be found here.

Upgrading from JIRA 3.12 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

9 September 2008

The Atlassian JIRA team is proud to present **JIRA 3.13**.

This release fulfils some of the most popular JIRA feature requests. Dashboards can now be shared, and filter sharing has been improved — so it's easy to set up multiple 'template' dashboards, each with specific portlets and filters. New JIRA users can then simply select the dashboards most suited to them.

People using customised workflows will be pleased to learn that JIRA 3.13 provides the ability to edit active workflows — that is, workflows that are currently being used. So workflow logic, transitions, screens and post-functions can now be tweaked on the fly, and a JIRA wizard will guide decisions on how active issues should be handled.

By popular request, you can now restore individual projects from a backup, making it much easier to merge projects back into your existing JIRA instance. We are also happy to announce that personal licenses are available with this release of JIRA.

Please be aware of JIRA Security Advisory 2008-08-26, which is relevant to this release.

Upgrading to JIRA 3.13 is free for all customers with active JIRA software maintenance.

Highlights of JIRA 3.13

- Shareable dashboards
- Improved filter sharing
- Favourite filters and dashboards
- Restoring projects
- Editable active workflows
- Enhanced sub-task quick creation
- Personal licenses
- Plugins
- Progress bar for long-running operations
- Application improvements
- Plus more than **200** other fixes and improvements

Thank you for your feedback

85 new feature and improvement requests implemented!

1485 votes fulfilled!

Your votes and issues help us keep improving our products, and are much appreciated.

DOWNLOAD latest version

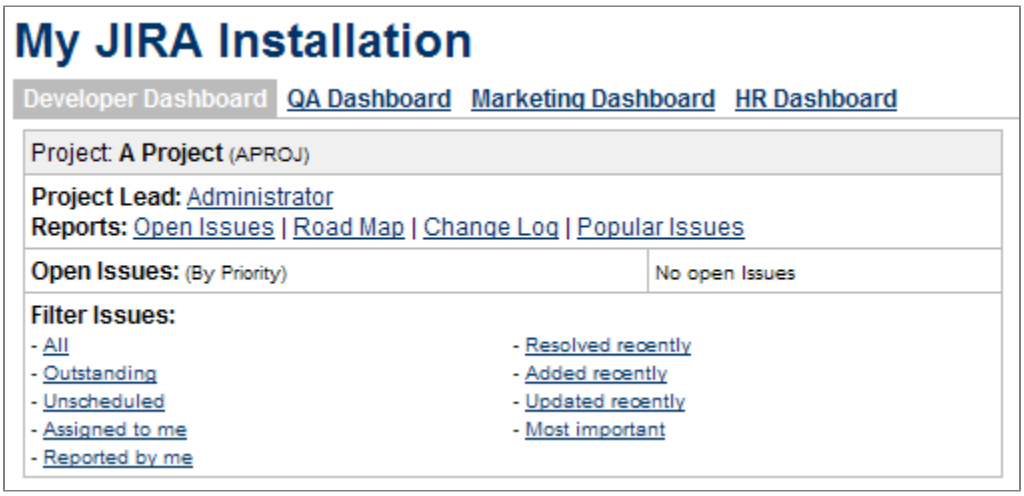
Upgrading to JIRA 3.13
You can download JIRA 3.13 from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.13 Upgrade Guide.

Highlights of JIRA 3.13

Shareable dashboards

In many organisations, different departments use different JIRA projects and so need a customised dashboard for people in each department to use. For instance, at Atlassian our HR team uses their own special 'Recruitment' project and workflow to manage the hiring process. JIRA 3.13 resolves this popular issue (434 votes!) by letting you set up dashboard pages that you can share with your user group, project or all JIRA users.

- Read more about shareable dashboards in the dashboard pages documentation.

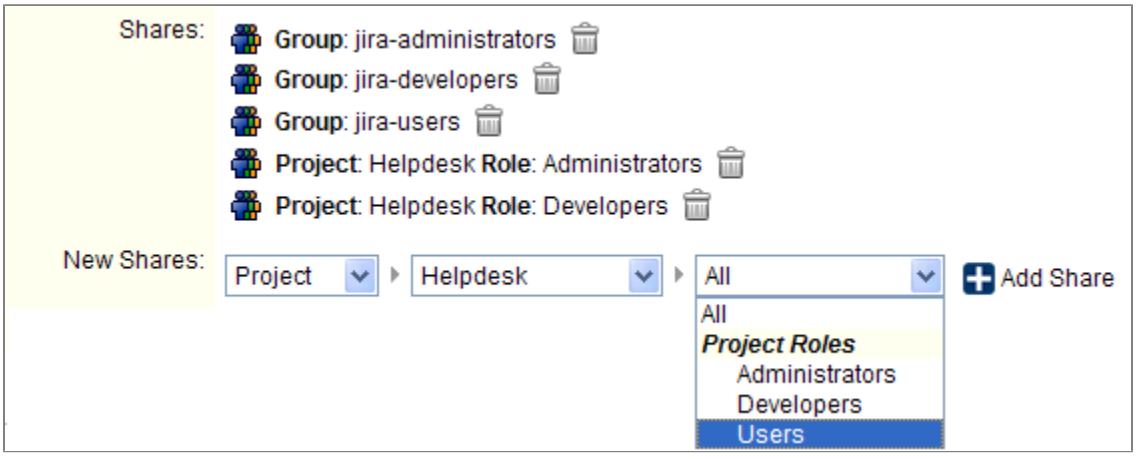


^Top

Improved filter sharing

Previous versions of JIRA allowed you to share filters, but filter sharing was restricted to sharing with a single user group or sharing with all JIRA users. Filter sharing is much more flexible in this release. You can now also share your filter with multiple user groups, projects, specific project roles or any combination of these.

- Read more about shareable filters in the issue filters documentation.

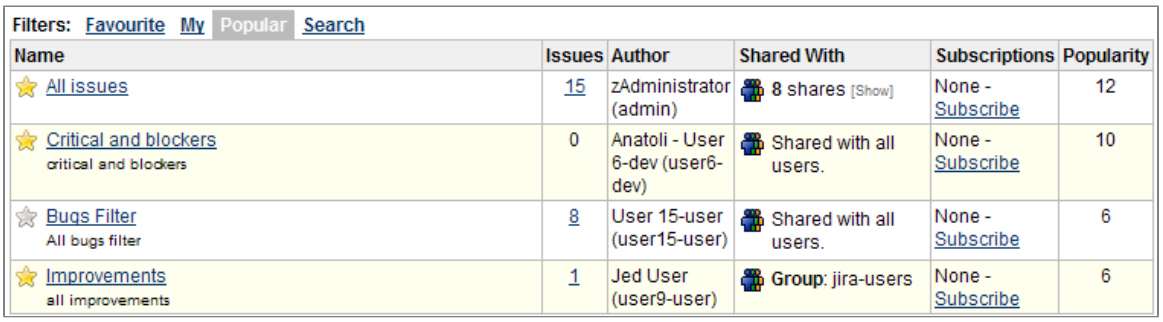


^Top

Favourite filters and dashboards

Keep your most commonly used filters and dashboards at your fingertips by adding them as favourites. You can add your own filters and dashboards as favourites in JIRA 3.13, as well as filters and dashboards shared with you by other users. We have also added search for filters and dashboards to help you find the tools you need to manage your information.

- Read more about favourite filters and dashboards in the issue filters and dashboard pages documentation.

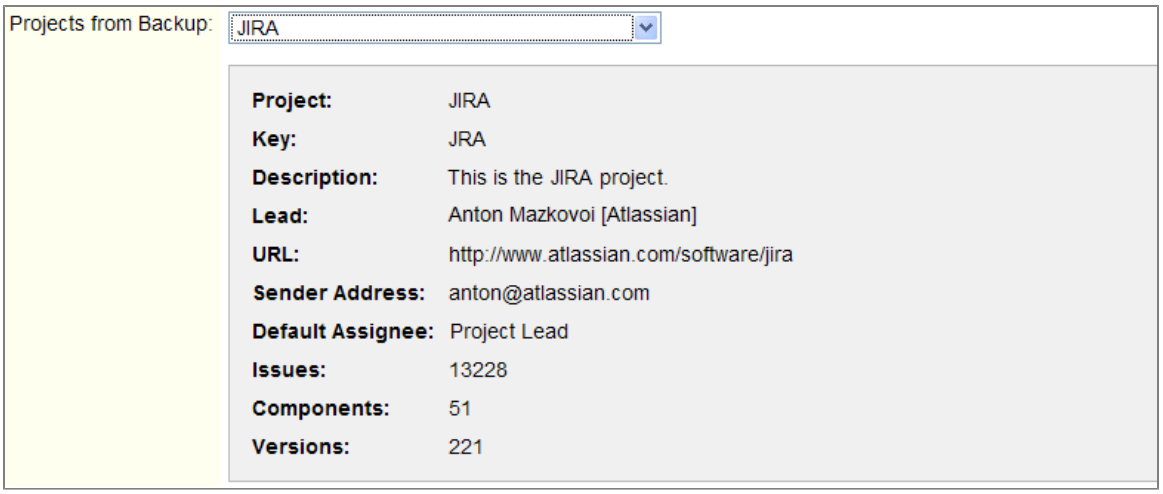


^Top

Restoring projects

One of the most voted for JIRA features has been included in this release — the ability to restore individual projects from a backup file (409 votes!). The project import feature allows you to select a project from a backup file and restore it into an existing JIRA instance, without losing the existing projects or data.

- Read more about restoring a project from backup.

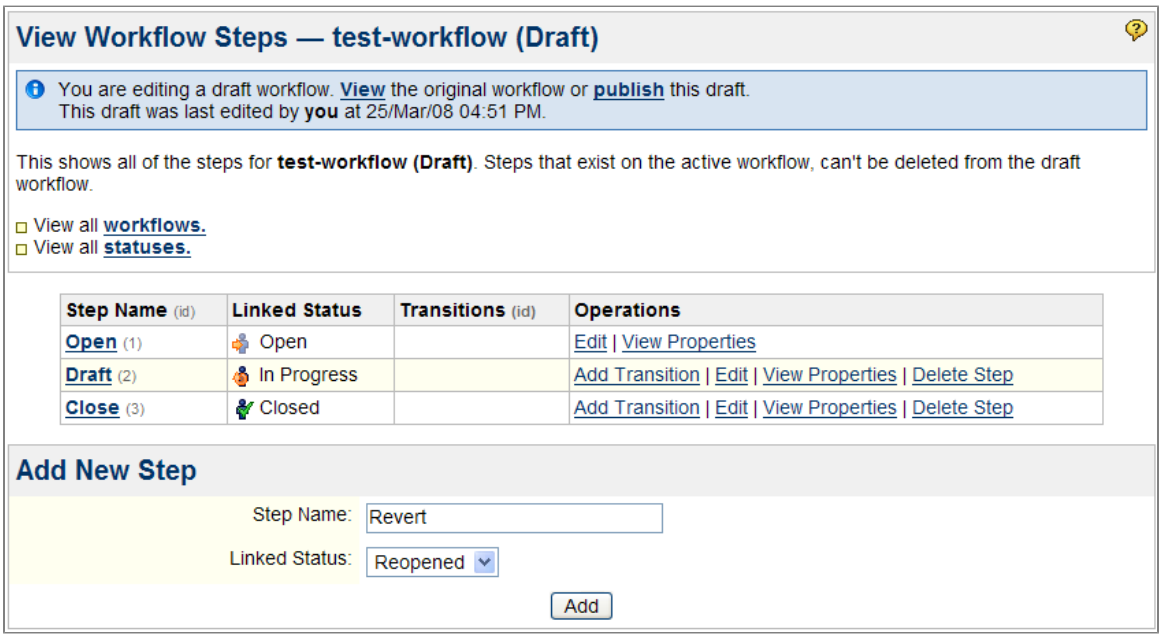


^Top

Editable active workflows

We have added the ability to edit active workflows in this release. You can now change most features of your active workflow without having to edit a separate copy of it.

- Read more about editing active workflows in the JIRA workflow documentation.



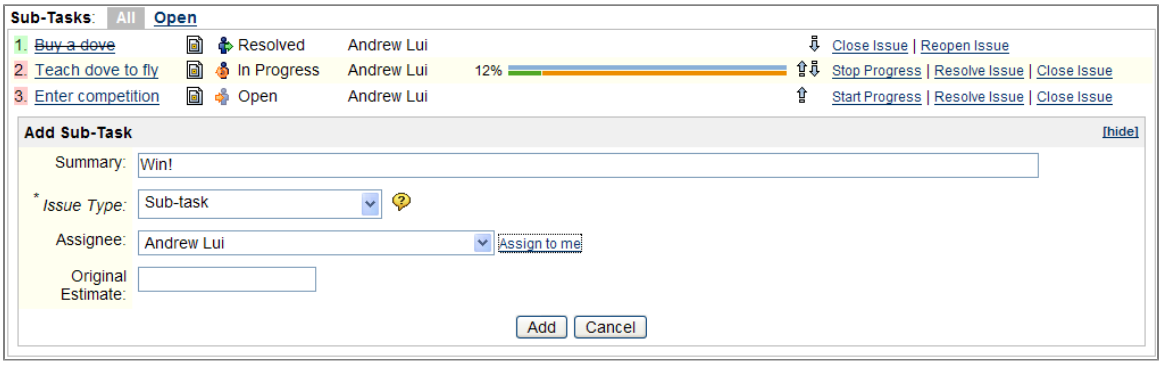
^Top

6

Enhanced sub-task quick creation

The sub-task 'quick creation' form included on the issue page in JIRA has been enhanced, making it easier for you to quickly add sub-tasks to an issue without having to navigate to a new page. The sub-task quick creation form includes new fields, such as the 'Original Estimate' field and can be easily hidden to reduce clutter on your screen.

- Read more about the sub-task 'quick creation' form in the creating a sub-task documentation.



^Top

7

Personal licenses

Personal Licenses are now available with JIRA 3.13. Run your own individual non-commercial instance of JIRA under this free license. You can run your fully functional instance of JIRA indefinitely under this license, but you will not be eligible for Atlassian support.

- Read more about personal licenses.

^Top

Plugins

We are bundling a new version of the **FishEye plugin for JIRA** in this release, helping your JIRA instance to work even more closely with your FishEye and Crucible applications. You can now use the FishEye plugin for JIRA to integrate your JIRA instance with Perforce. In addition, we've added a bunch of cool new features to the plugin, including FishEye and Crucible specific charting portlets for JIRA, the integration of reviews and code commits into JIRA workflow and trusted applications support.

- Read more about the FishEye plugin for JIRA

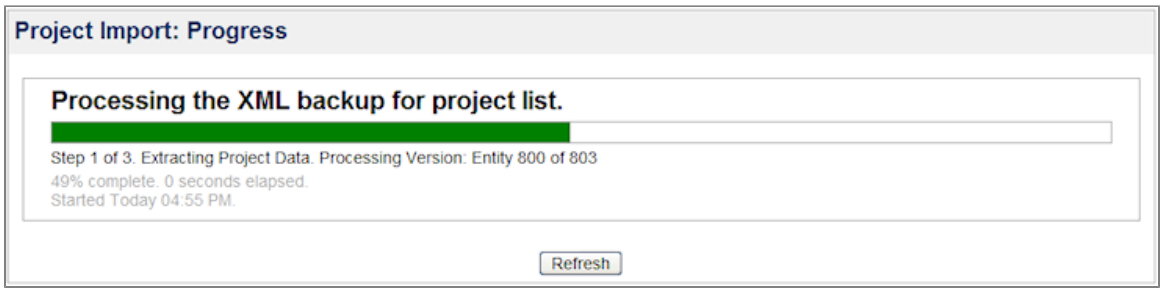
You may also be interested in integrating your JIRA instance with your Bamboo application. We have recently updated the **JIRA Bamboo plugin** with a host of new functionality to let you monitor your builds and issues. (Note that the JIRA Bamboo plugin is currently not bundled with JIRA.)

- You can get the plugin from the JIRA Bamboo Plugin page.

^Top

Progress bar for long-running operations

We have added a progress bar to a number of long-running operations in JIRA, including workflow migration, project import and re-indexing. This simple visual cue provides you with helpful information about your task, such as the time elapsed, percentage complete and the time you started it.



^Top

Application improvements

The version of Tomcat that is shipped with JIRA Standalone has also been upgraded to version 5.5.26 (previously Tomcat 5.5.20). Refer to the 5.5.x Release Notes on the Apache Tomcat website.

^Top

Plus more than 200 other fixes and improvements

The top 50 most popular issues resolved in JIRA 3.13 are listed below. See all the issues here.

JIRA Issues (50 issues)					
Key	**Type**	**Summary**	**Priority**	**Status**	**Votes**
JRA-14423		Can not browse issue on IE 7		Resolved	0
JRA-5803		Global portals, configurable project portlet, project groups		Resolved	13
JRA-14980		Suboptimal performance for Lucene queries within JIRA instances with a large number of projects and issue security levels		Resolved	2
JRA-14070		Clickable Users in Issue Navigator table to see User's Profile		Resolved	0
JRA-14074		Viewing subscriptions under standard does not render correctly.		Resolved	0
JRA-14057		Using the AJAX User Picker within the "Add New Permission" section of the admin panel causes the page to become unusable		Resolved	0
JRA-11634		Directory listing enabled on Tomcat		Resolved	0
JRA-14760		Cannot configure dashboard which contains a portlet which is now disabled / not on classpath		Resolved	0
JRA-13479		Permission denied when viewing issue moved from project with security scheme to project with none		Resolved	0
JRA-15412		NullPointerException when useragent does not exist		Resolved	0
JRA-2509		Save, reuse and share Dashboards (like dashboard picklist)		Resolved	435
JRA-1604		Import / Export (backup / restore) individual projects		Resolved	409
JRA-4817		Ability to share custom Portals		Resolved	91
JRA-4139		Share filters with *multiple* groups		Resolved	82
JRA-7887		Add saved filter types / categories - personal, favourite and all saved filters		Resolved	18
JRA-5806		Ability to subscribe to shared filters		Resolved	2
JRA-15117		Add user photos to JIRA action item headers for Studio		Resolved	0
JRA-7661		Ability to Edit Active Workflow - Editing Workflows is too inflexible		Resolved	177
JRA-11882		Filter sharing does not currently allow you to share with a role.		Resolved	41
JRA-6178		shared filters and "favorites"		Resolved	16
JRA-2394		Change attachment size limit in the Web Interface / GUI		Resolved	16
JRA-10859		Create a generic mechanism within JIRA that allows a user to fire off long running tasks but still receive feedback about the progress of the task.		Resolved	11
JRA-3769		Security levels should be sorted alphabetically or be orderable		Resolved	9

JRA-6124	Share filter per project (or project categorie)		Resolved	7
JRA-10525	Add progress bar to Workflow migration		Resolved	6
JRA-12271	Print system info to the logs on startup		Resolved	1
JRA-15098	Watches and Votes cannot handle users "disappearing" (ie being deleted from external user management).		Resolved	1
JRA-15190	Deterioration of performance due to CachingVersionStore lock		Resolved	1
JRA-15314	Put the version of JIRA in the head element of all XML/RSS results		Resolved	1
JRA-15313	Saved Filter URLs do not respect additional parameters		Resolved	1
JRA-14364	JIRA Header Filter and History Links are to be inline divs		Closed	0
JRA-15011	Fix ordering of versions and components values on the View Issue screen		Resolved	0
JRA-15163	HTML format changes caused a blank line beneath the logo/banner		Resolved	0
JRA-14668	'Resolve Issue' screen should display brief Issue Summary at top.		Resolved	0
JRA-14855	Change the message on the Add Project page for the Key input field		Resolved	0
JRA-14491	Change "Locale" to "Language" in the JIRA preferences pages		Resolved	0
JRA-14157	JIRA Web Test - Need to allocate time to merge in new func test framework		Resolved	0
JRA-14283	More advanced resampling algorithm for generation of thumbnails		Resolved	0
JRA-14818	Update text that appears on the first Setup page for the License input box		Resolved	0
JRA-15049	Add better debugging information to portlet rendering		Resolved	0
JRA-14893	Do not render a hyperlink for a change history item for linking issues when the user does not have permissions to see the linked issue		Resolved	0
JRA-13439	Smart querying : use syntax 'c:' to search for components		Resolved	0
JRA-13886	Filter Statistics Portlet: Options "Sort By" and "Sort Direction" same as in "Two Dimensional Filter Statistics" portlet		Resolved	0
JRA-15347	Skip issue type choose		Resolved	0
JRA-15127	RSS/XML issue filters must have the total number of results included		Resolved	0
JRA-13918	Add a help icon for issue security levels		Resolved	0
JRA-15226	add instruction for IIS7 to Integrating Tomcat (JIRA) with IIS		Resolved	0
JRA-14466	Place Extra JIRA system Info into exported XML		Resolved	0

JRA-14000		Unify icons of controls which do the same but look very different		Resolved	0
JRA-15394		a new dangerous Tomcat caching property - tomcat.util.buf.StringCache.byte.enabled=true		Resolved	0

^Top

JIRA 3.13 Upgrade Guide

Upgrading from JIRA 3.12.xx to 3.13

Please follow the JIRA general upgrade instructions, plus note the following:

1. Introduction of Favourite Dashboards and Filters

Favourite Dashboards

JIRA 3.13 introduces the favourite dashboards feature, which allows you to add dashboard pages that are owned by you or shared by other users as favourites (and hence, are displayed as tabs on your dashboard). On upgrade to JIRA 3.13, all your dashboard pages will be added as your favourites and displayed on your dashboard. If you do not wish any of your dashboards to be added as favourites, then you can remove them as favourites after the upgrade. See the dashboards documentation for details.

Favourite Filters

Similar to favourite dashboards, JIRA 3.13 introduces the favourite filters feature, which allows you to add issue filters that are owned by you or shared by other users as favourites. On upgrade to JIRA 3.13, all your issue filters will be added as your favourites. If you do not wish any of your filters to be added as favourites, then you can remove them as favourites after the upgrade. See the issue filters documentation for details.

please note, this change will not affect issue filter sharing, e.g. if you are using a shared issue filter in one of your dashboard portlets, it will still be shared with you after the upgrade.

please also note, that any custom developed portlets (or other JIRA objects that use filters that have been developed by 3rd parties) that have a dropdown list (not a pop-up picker) for filters, will now only show a list of the user's favourite filters, instead of all shared filters.

Favourite Filters portlet

The 'List All Filters' portlet has been replaced with the 'Favourite Filters' portlet in this release. Your dashboard will be automatically upgraded if it is currently configured to display the 'List All Filters' portlet.

2. Tomcat, MySQL database connection dropouts

Please note, if you wish to use a MySQL database with JIRA Standalone you must set up the bundled Tomcat server (version 5.5.26) to survive connection closures. You must also do this if you are running JIRA EAR/WAR in Tomcat 5.5.25 or later, or Tomcat 6.0.13 or later. Versions 5.5.25 and above of Tomcat 5, and versions 6.0.13 and above of Tomcat 6, have been noted to exhibit problems maintaining connections to MySQL databases. Please read this document for details on the changes required.

3. Changes to jira-application.properties

jira.subscription.email.max.issues property

The `jira.subscription.email.max.issues` property has been added to the `jira-application.properties` file. This property allows you to specify the maximum number of issues that can be included in an email subscription. The default value for this property is 200. You may wish to update this property after the upgrade if you wish to set a different limit on the number of issues that can be included in an email subscription. See the documentation on Advanced JIRA configuration with jira-application.properties for further details on this file.

4. Support for Portlet Plugins with JSP Views Discontinued

Portlet plugins with JSP views are no longer supported. If you have written a custom Portlet plugin and have used a JSP as the view template, you will need to convert your JSP to Velocity.

5. Updates to JIRA SOAP and XML-RPC APIs

com.atlassian.jira.rpc.soap.JiraSoapService

- replaced

```
RemoteProject[] getProjects(String token) throws RemoteException;
```

  with

```
RemoteProject[] getProjectsNoSchemes(String token) throws RemoteException
```

*You should use **getProjectsNoSchemes()** instead because it much more memory efficient and quicker.*

- added

```
RemoteProject getProjectWithSchemesById(String token, Long projectId) throws
RemoteException;
```

- deprecated

```
RemoteFilter[] getSavedFilters(String token) throws RemoteException;
```

- added

```
RemoteFilter[] getFavouriteFilters(String token) throws RemoteException;
```

com.atlassian.jira.rpc.xmlrpc.XmlRpcService

- replaced

```
Vector getProjects(String token) throws Exception;
```

  with

```
Vector getProjectsNoSchemes(String token) throws Exception;
```

- deprecated

```
Vector getSavedFilters(String token) throws Exception;
```

- added

```
Vector getFavouriteFilters(String token) throws Exception;
```

6. Crowd Cache Timeout

This is only applicable if you are using Crowd.

The default timeout for caching user details has changed from 5 minutes to 2 hours. This will improve the performance of the application but will mean that it will take longer for changes to user details to reach the application. Details on how to configure the Crowd caches can be found here.

Upgrading from JIRA 3.12 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.5 Release Notes

21 July 2009

The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.5** in Standard, Professional and Enterprise editions. This point release includes over **30** bug fixes and improvements.

JIRA 3.13.5 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.13 yet?
Take a look at all the new features in the JIRA 3.13 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.13.5 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.13.5 includes the following updates and bug fixes:

JIRA Issues (34 issues)

Type	Key	Summary	Priority	Status
	JRA-17864	Plugins with broken "project-operation" modules can prevent other modules project-operations from being loaded		Resolved
	JRA-17685	Adding group for "Project Role Browser" using group picker is not working.		Resolved
	JRA-17623	IssuePicker ajax queries can use excessive memory		Resolved
	JRA-17582	Job ClassNotFoundException's causes all services to not run		Resolved
	JRA-17546	Time Tracking Report does not include resolved sub-task with selected version		Resolved
	JRA-17513	java.lang.IllegalStateException: getAttribute: Session already invalidated - Patch available		Resolved
	JRA-17446	Unable to export to file xml workflow in IE over HTTPS		Resolved
	JRA-17421	Installer does not substitute <jira_service_id> in the {start_service, stop_service}.bat scripts when not installing JIRA as a service.		Resolved
	JRA-17373	Improve https speed by adding Cache Control Public to the caching headers for resources in JIRA		Resolved
	JRA-17367	Outlook 2007 is unable to read JIRA RSS feeds over HTTPS		Resolved
	JRA-17360	Session Timeout dumps exception when viewing manage attachments		Resolved

	JRA-17244	Project import tries to import groups for any project's comment visibility field		Resolved
	JRA-17222	Issue linking within subtasks breaks when cloning		Resolved
	JRA-17188	Bug in autocomplete-widget.js. IE have bug in removeChild		Resolved
	JRA-17169	Ugly javadoc for IssueSearcher		Resolved
	JRA-17107	Administration Group Browser has broken pager		Resolved
	JRA-17011	bulk move (and hence bulk migrate) of issue populates subtasks' assignee list with wrong users		Resolved
	JRA-17005	Merge Versions sometimes sets the wrong version		Resolved
	JRA-16990	Unable to set transition property "jira.i18n.submit"		Resolved
	JRA-16912	ViewTranslations admin page throws exception when no locales exist		Resolved
	JRA-16902	Two Dimensional Filter Statistics breaks on Select CF with same value in two contexts, one of which is not associated with any project.		Resolved
	JRA-16836	Microsoft Internet Explorer 8 will not render Issue screen tabs correctly if not in IE8 standards compliant mode		Resolved
	JRA-16825	IssueNavigator produces harmless but incorrect urls that result in logging messages		Resolved
	JRA-16802	Unpredictional user changes		Resolved
	JRA-16420	text on the Edit Issue Type Screen Scheme is incorrect		Resolved
	JRA-15966	Public sign up should not be available, if external user management is ON		Resolved
	JRA-15950	Mantis importer duplicates some log messages		Resolved
	JRA-15914	Project import failing without attachments.		Resolved
	JRA-15835	Clicking on link "Configure"" after deleting all columns in Navigator columns do nothing		Resolved
	JRA-15831	Fogbugz importer fails to create links		Resolved
	JRA-14580	Attachments are accessible for download using different urls		Resolved
	JRA-14024	Testing CVS module throws IllegalArgumentException		Resolved
	JRA-12886	Ugly reporting of failing validators on Create Issue		Resolved
	JRA-7830	Multi user picker loses checked users when Prev / Next clicked		Resolved

Upgrading from JIRA 3.13.x to 3.13.5

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.12.x and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.4 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

5 May 2009

The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.4** in Standard, Professional and Enterprise editions. This point release includes over **20** bug fixes and improvements.

JIRA 3.13.4 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.13 yet?
Take a look at all the new features in the JIRA 3.13 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.13.4 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.13.4 includes the following updates and bug fixes:

JIRA Issues (22 issues)				
Type	**Key**	**Summary**	**Priority**	**Status**
	JRA-16969	Define new views in entitymodel.xml and entitygroup.xml for FishEye plugin		Resolved
	JRA-16859	Jelly TransitionWorkflow through a workflow transition that uses a screen with the resolution field fails even if resolution is already set on the issue		Resolved
	JRA-16853	RPC Plugin does not use finally blocks around setUser calls on ActionContext.		Resolved
	JRA-16848	Mods Detector incorrectly reports removed files for JIRA Professional WAR/EAR deployments		Resolved
	JRA-16846	Allow for Pluggable Decorators in JIRA		Resolved
	JRA-16843	The hide toggle on environment fields seems to have gone in 3.13.3		Resolved
	JRA-16782	Include connection pool size details in JBoss 4 installation notes		Resolved
	JRA-16780	On startup JIRA fails to search for Tables in the specified schema name		Resolved
	JRA-16762	Changing user properties dosn't invalidates properties cache		Resolved
	JRA-16755	Edit dashboard when shared with a project the user has no browse permission over fails		Resolved
	JRA-16726	File attachment having % in their file name are not allowed		

				Resolved
	JRA-16680	Running long Jelly scripts can fail		Resolved
	JRA-16611	Time Spent does not reflect the total time spent by issue and sub-tasks in HTML filter mail subscriptions		Resolved
	JRA-16579	User is unable to attach file having $ in filename, post creation of issue		Resolved
	JRA-16433	General Configuration XSS option help references non-public issue		Resolved
	JRA-16378	add comment to jira-application properties explaining weirdness with SimpleDateFormat's handling of yy versus yyyy		Resolved
	JRA-16367	Old Filter Sharing Screenshots still exist in JIRA 3.13.x documentation.		Resolved
	JRA-16223	Bulk move attachments remain in previous project folder		Resolved
	JRA-15921	Incorrect error message when validating worklog creation for closed issues		Resolved
	JRA-15795	Adding a new context to a custom field produces "[webwork.view.taglib.IteratorTag] Value is null! Returning an empty set"		Resolved
	JRA-15475	Attachment lost with parallel bulk moves		Resolved
	JRA-14032	Subtask can be created when parent issue is closed		Resolved

JIRA 3.13.4 Upgrade Guide

Upgrading from JIRA 3.13.x to 3.13.4

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.12.x and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

2 April 2009

The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.3** in Standard, Professional and Enterprise editions. This point release includes over **85** bug fixes and improvements, including an important security fix — please see JIRA Security Advisory 2009-04-02 for details.

JIRA 3.13.3 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.13 yet?
Take a look at all the new features in the JIRA 3.13 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.13.3 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.13.3 includes the following updates and bug fixes:

JIRA Issues (92 issues)				
Type	**Key**	**Summary**	**Priority**	**Status**
	JRA-18454	Bulk move of issue type creates duplicate entries in Change History		Resolved
	JRA-16752	Broke SearchProvider backwards compatibility in JIRA 3.13.3		Resolved
	JRA-16742	SOAP search methods are unbounded - this can lead to xml-rpc generating huge xml responses causing memory problems		Resolved
	JRA-16739	Broken bars in Project Statistics portlet		Resolved
	JRA-16677	JIRA build information not included in dummy XML responses to search filter requests which users do not have access to		Resolved
	JRA-16612	AttachmentNotFoundException does not report which attachment is missing		Resolved
	JRA-16608	Indent the XML and java code samples on Plugin documentation page.		Resolved
	JRA-16566	Jelly #inactivate section slightly confusing		Resolved
	JRA-16563	Improve documentation of catchemail parameter in createorcommenthandler		Resolved
	JRA-16562	Backport Jeds webwork1 bug fixes to JDK 1.4		Resolved
	JRA-16561	Take out URL encode from WebWork URL tag so that we get rid of the jessionid= problem		Resolved
	JRA-16549	Strange sort by fields		Resolved
	JRA-16546	Shared dashboard documentation does not explain how to remove a dashboard from the users favourites		Resolved
	JRA-16498	Version/s and Component/s not validated when updating an issue		Resolved
	JRA-16478	NullPointerException when entering a bad sorter/field parameter in the URL of the Printable view link		Resolved
	JRA-16462	Transition hovertip shows dash when there is no description		Resolved
	JRA-16449	Outdated / misleading documentation of Look & Feel section in Administration panel		Resolved
	JRA-16438	Cannot create attachement with "Subtask Creation" quickform		Resolved
	JRA-16429	Remote version not validated when creating new issue over SOAP		Resolved
	JRA-16419	Resolution selector in filter creation has no scrollbar in FF3		Resolved
	JRA-16402	Visio attached files do not open when clicked on		Resolved

	JRA-16384	unintentional use of font-size:0 in versiondescription.jsp		Resolved
	JRA-16356	Date Picker fields won't "hide" on Issue Navigator page		Resolved
	JRA-16338	The Windows installer PermGen memory settings are lower than for other standalone releases.		Resolved
	JRA-16326	Create Comment Function fails when a workflow transition is invoked by a non logged in user		Resolved
	JRA-16320	There is no way to i18n some sort of plugin modules.		Resolved
	JRA-16264	Transforming wikimarkup in textarea into HTML (Email Notifications)		Resolved
	JRA-16239	A message "java.lang.IllegalArgumentException: fieldName must not be null" appears in the log when indexing language is French		Resolved
	JRA-16230	Plugin Servlets are not recreated after importing configuraton		Resolved
	JRA-16224	Environment field showing on issue summary when it's not on the View screen		Resolved
	JRA-16207	The "Surviving Connection Closures" page on Confluence still includes autoReconnect=true in the example MySQL configuration		Resolved
	JRA-16199	Plugins that index Link information can end up with stale link information in the Lucene Index.		Resolved
	JRA-16197	Time Tracking Report - displaying issue with Key ID's		Resolved
	JRA-16195	Support Request feature improvement		Resolved
	JRA-16183	NullPointerException fetching mail with CreateOrCommentHandler		Resolved
	JRA-16176	When email has an invalid address the mail handlers can throw a MessageException		Resolved
	JRA-16170	The javascript popup calendar has wrong short month names for Finnish		Resolved
	JRA-16169	Internationalisation to dutch		Resolved
	JRA-16159	how to change the german translation?		Resolved
	JRA-16156	Jelly tag AddIssueSecurityLevel ignores the provided description and creates a "tagName description" description		Resolved
	JRA-16149	No localisation for description on page fixforversion		Resolved
	JRA-16146	Inconsistent HTML under "Summary"		Resolved
	JRA-16135	Manage Versions page inconsistent across browsers		Resolved
	JRA-16126	JIRA 3.13.2 has shipped with the incorrect version of the crowd-ehcache.xml file in WEB-IN/classes for Crowd 1.5.x versions.		Resolved
	JRA-16124	Potential bug in Deleting a User Account FAQ		Resolved
	JRA-16111	Emails containing an issue key that has been moved result in new issues being created.		Resolved

	JRA-16107	German Translation Error in Manage Project Role Membership for Project page		Resolved
	JRA-16097	JIRA system info page does not show time zone in action		Resolved
	JRA-16072	DWR has a XSS security hole in it		Resolved
	JRA-16052	crashes on non-numerical input in isssue type migration		Resolved
	JRA-16034	"Action 'com.atlassian.jira.web.action.admin.workflow.ViewWorkflowSteps' could not be instantiated" when trying to draft workflow with ampersand in its name		Resolved
	JRA-16028	Quick Search excludes Environment field		Resolved
	JRA-16010	XML output from SearchRequest contains XML entities valid only in XML 1.1		Resolved
	JRA-16007	Fix version is not modified after a move depending on permissions		Resolved
	JRA-16003	Base URL in Trusted Applications page accepts illegal characters		Resolved
	JRA-16002	Auto-generated hyperlinks are broken if they contain more than one issue number		Resolved
	JRA-15998	The Search Query Syntax does not document the Escaping Special characters anymore		Closed
	JRA-15915	Missing French translations on 500 error page		Resolved
	JRA-15911	Cannot revoke global permissions from non existent groups		Resolved
	JRA-15855	Confusing error message in Edit Groups when no group is selected		Resolved
	JRA-15847	On access /si and /sr directories NPE is produced		Resolved
	JRA-15845	The colour picker only gives me one chance to select a colour		Resolved
	JRA-15749	German Translation regression in JIRA 3.13		Resolved
	JRA-15745	Some words are not translated		Resolved
	JRA-15705	wrong german translation "Fest für:"		Resolved
	JRA-15701	Put "JIRA lock up" warning message on the Indexing page		Resolved
	JRA-15641	Date Time CF picker fails validation for (at least) Japanese		Resolved
	JRA-15526	jsessionid populated in URL causing NumberFormatException		Resolved
	JRA-15504	Change button prompt from Edit to Update		Resolved
	JRA-15477	When publishing draft workflow, default name is not provided		Resolved
		When mouse is over the logo in the header the mouse icon does not change to a hand in IE 6 like it		

	JRA-15443	does in Firefox		Resolved
	JRA-15369	The 2D Statistics Portlet does not align itself vertically correctly - it adds an extra 5 pixels at the top and bottom of itself.		Resolved
	JRA-15322	Invalid email address is created when the users have an email address as their username during the CSV Import		Resolved
	JRA-15262	Windows installer doesn't allow to install JIRA service as windows service on Windows 2008		Resolved
	JRA-15063	Assignable permission given to User CF value causes errors in logs		Resolved
	JRA-14925	Make the warning more prominent that attachments are not backed up		Resolved
	JRA-14848	Zero and Eight difficult to distinguish in Internet Explorer on resolved "strike-through" issue numbers		Resolved
	JRA-14713	Use divs instead of tables for bar graphs so that they appear on printed pages properly		Resolved
	JRA-14430	Document the "Remember my login on this computer" tick box on the login page		Resolved
	JRA-14214	Office 2007 attachments (open document format files) download as ZIP files on some browsers		Resolved
	JRA-13996	"notifyusers=true" handler parameter is not interpreted correctly		Resolved
	JRA-13685	JIRA should better handle errors return from Crowd (notificaton messages)		Resolved
	JRA-13620	Custom field defaults are not applied when issue created via XML-RPC		Resolved
	JRA-13608	When downloading attachment with spaces in file name, JIRA replaces spaces to "+"		Resolved
	JRA-13157	Renaming a project role leaves the old name to database		Resolved
	JRA-12165	Unclear error message when bulk moving issues whose reporter cannot create issues		Resolved
	JRA-11693	SOAP: addAttachmentsToIssue runs out of memory when adding attachments of 1.5 MB or larger		Resolved
	JRA-9828	JIRA runs out of heap space and throws OutOfMemoryError on bulk operations		Resolved
	JRA-9339	Manage Attachments link leads to "this issue no longer exists" error intermittently.		Resolved
	JRA-8684	Clarify in documentation what "Remember my login on this computer" means; e.g. user stays logged in, but filters timeout when login (session) doesn't		Resolved
	JRA-8489	View Issue take a long time (slow) after search returning lots of issues		Resolved
	JRA-6929	Create a language pack for English (American)		Resolved

JIRA 3.13.3 Upgrade Guide

Upgrading from JIRA 3.13.2 to 3.13.3

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.12 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.2 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

9 December 2008

The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.2** in Standard, Professional and Enterprise editions. This point release includes over 45 bug fixes and improvements, including an important security fix — please see JIRA Security Advisory 2008-12-09 for details.

JIRA 3.13.2 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.13 yet?
Take a look at all the new features in the JIRA 3.13 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.13.2 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.13.2 includes the following updates and bug fixes:

JIRA Issues (52 issues)

Type	Key	Summary	Priority	Status
	JRA-17019	Attachments in E-mail not attached to issue on "Issue Creation from Email" or "Issue Update from Email"		Resolved
	JRA-16436	Bugzilla Importer documentation needs to be updated for 3.13.1		Resolved
	JRA-16201	Menu Item has a typo - Defining 'Isdownsue Type' field values		Resolved
	JRA-16200	Link on Setup Wizard page is broken		Resolved
	JRA-16081	Link to Lucene Query Parser Syntax doc is broken on Search Query Syntax page		Resolved
	JRA-16036	import / export page needs to more clearly specify where the backup file must be located		Resolved
	JRA-16032	Group Role Actor permission checks take a long time with many groups in the system		Resolved
	JRA-15997	wrong UTF-8 encoding in dashboard page title		Resolved
	JRA-15994	Issue Type select list is blank when a specific project is selected		Resolved
	JRA-15954	IMAP message handling produces issues with no description		Resolved
	JRA-15949	Issue created event for a sub-task should pass through the id of the parent issue		Resolved
	JRA-15943	Importing a backup while specifying a new index directory still deletes the old one		Resolved

	JRA-15906	HTML tags in project description are not rendered		Resolved
	JRA-15890	Single apostrophe in a group name causes javascript error in Group Picker		Resolved
	JRA-15887	During version merge an affects version or fix for version will be set even if it is not required		Resolved
	JRA-15884	Error when installing JIRA service manually in Windows 2008		Resolved
	JRA-15879	'ConcurrentModificationException' can be thrown when loading Jira services.		Resolved
	JRA-15878	Standalone service installer does not set the 'org.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER' by default.		Resolved
	JRA-15876	Issue icons with absolute URL is prepended with context path in Issue Navigator XML		Resolved
	JRA-15858	Change History Items in a subtask can be duplicated on move to another project.		Resolved
	JRA-15843	Cannot create issues using a custom workflow		Resolved
	JRA-15837	"Caching" issue in Group Browser page		Resolved
	JRA-15825	Multiple project import from bugzilla fails		Resolved
	JRA-15824	'Finish' buttons on wizards are not internationalised		Resolved
	JRA-15823	Move Issue fails to create change history items for values not in target field configuration scheme		Resolved
	JRA-15821	Search description breaks when filtered by reporter group		Resolved
	JRA-15810	Clicking cancel on the change issue type scheme issue migration scheme leads to a blank page		Resolved
	JRA-15804	Index path is not validated when importing data into a new JIRA instance		Resolved
	JRA-15793	Changing Issue Type Sceen Sheme Fails With German Language		Resolved
	JRA-15775	Data added to a backup by JIRA related to FishEye caused breakages with JIRA		Resolved
	JRA-15770	NPE when editing a filter shared with a group the user does not belong to		Resolved
	JRA-15765	XML import from JIRA 3.11 to 3.13 failing on windows		Resolved
	JRA-15753	"The string "--" is not permitted within comments" error when restoring JIRA 3.13 backup		Resolved
	JRA-15744	Missing group causes portlets to break with stacktrace		Resolved
	JRA-15717	bulk parameter for mail handlers checks Auto-Submitted headers as well		Resolved
	JRA-15709	AssignToLeadFunction shouldn't log informational message at WARN level		Resolved

	JRA-15683	No apparent order to Issue Type Schemes when in 'Choose Issue Type Scheme' view in creating a new project		Resolved
	JRA-15670	attachment handling broken for Outlook 2007		Resolved
	JRA-15658	Empty Content-Encoding in SOAP requests when using gzip Accept-Encoding in request		Resolved
	JRA-15643	NullPointerException when editing an existing Permission Condition or Permission Validator		Resolved
	JRA-15642	Footer becomes part of content on Browse Project Roles for User page		Resolved
	JRA-15632	Allow the user to input a different port other than POP3 110		Resolved
	JRA-15622	Publish Draft Workflow screen contains very confusing text.		Resolved
	JRA-15561	Recognise delivery status messages better		Resolved
	JRA-15551	Editing a comment added by a deleted user generates a bad notification email		Resolved
	JRA-15440	jndi-jdbc jndi-server-name directions in entityengine.xml is not correct for jboss and external database		Resolved
	JRA-15325	Add "Auto-submitted" header to all email notifications to prevent "Out Of Office" messages		Resolved
	JRA-15205	Duplicate Issue Keys can be created		Resolved
	JRA-14877	Deleting a link changes the "updated date" field, but it does not reindex lucene.		Resolved
	JRA-13590	Add Mail Server Name and Id to all logging from MailFetcherService		Resolved
	JRA-12786	filter subscriptions pretend to work happily even when there is no configured mail server		Resolved
	JRA-9267	Memory leak in Jelly Runner while using AddComment tags		Resolved

JIRA 3.13.2 Upgrade Guide

Upgrading from JIRA 3.13.1 to 3.13.2

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.12 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.13.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

29 October 2008

The Atlassian JIRA team is proud to announce the release of **JIRA 3.13.1** in Standard, Professional and Enterprise editions. This point release

includes over 35 bug fixes and improvements, including important security fixes — please see JIRA Security Advisory 2008-10-29 for details.

JIRA 3.13.1 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.13 yet?
Take a look at all the new features in the JIRA 3.13 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.13.1 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.13.1 includes the following updates and bug fixes:

JIRA Issues (43 issues)

Type	Key	Summary	Priority	Status
	JRA-15748	Incorrect info displayed in Project Lead lists in JIRA deployed in Resin 3.x		Resolved
	JRA-15733	XSS bug on ViewProfile page		Resolved
	JRA-15718	JiraStartupLogger fails if there are third party services and listeners		Resolved
	JRA-15707	Return URL is not HTML escaped		Resolved
	JRA-15685	Spelling mistake of russian JIRA Web interface		Resolved
	JRA-15660	"jira.option.bulk.send.notifications" property is not used in JIRA		Resolved
	JRA-15653	Add user to Issue Security Level: misprint in error message		Resolved
	JRA-15648	JIRA source release does not find certain JARS		Resolved
	JRA-15645	wrong german translation "Fest für:"		Resolved
	JRA-15624	JIRA is asking Crowd for a user called "unassigned".		Resolved
	JRA-15612	syntax errors in JIRA standalone start up scripts		Resolved
	JRA-15608	HTTPS connector in JIRA Standalone does not define the useBodyEncodingForURI="true" flag to avoid problems with UTF-8 characters in the request URL		Resolved
	JRA-15602	ForgotPassword page NullPointer exception if you dont provide a user name		Resolved
	JRA-15601	Javascript error on the bulk transition page		Resolved
	JRA-15590	UserManager cache not flushed on config import		Resolved
	JRA-15588	Improvement to the Issue Security Scheme docs		Resolved
	JRA-15586	Use yahoo-dom-event.js instead of three separate includes		Resolved

	JRA-15582	Mail Finger Print defaults to "delete mail" even if there are no fingers prints.		Resolved
	JRA-15580	MailHandler deletes message that should be ignored due to catchmail parameter		Resolved
	JRA-15578	Case of Unassigned is lowercase on the issue navigator		Resolved
	JRA-15573	The Admin Default Dashboard can not be configured on JIRA standard		Resolved
	JRA-15572	Configure Default Dashboard throws error when no projects are created		Resolved
	JRA-15570	Sort order is not saved with filter		Resolved
	JRA-15567	Attachment path does not appear in the "attachment" system field (IE8)		Resolved
	JRA-15555	The 'SavePortlet' action does not have an error view.		Resolved
	JRA-15536	Icons for issue types loaded via HTTP not displayed in the View Custom Fields screen		Resolved
	JRA-15535	JIRA Admin portlet causes errors if added to dashboard then admin permission removed		Resolved
	JRA-15523	DefaultOfBizDelegator.setValue(Object, SQLProcessor) uses java.util.Date instead of java.sql.Date due to bad import statement		Resolved
	JRA-15520	FishEye plugin 2.2 bundled in JIRA 3.13 reports itself as being version 2.1		Resolved
	JRA-15513	Update CreateOrCommentHandler doco to say that "project" parameter only refers to creating an issue		Resolved
	JRA-15510	{1} on tooltip to move sub-task		Resolved
	JRA-15506	Incorrect Dutch translation of "Close Issue"		Resolved
	JRA-15502	Extra underscore between issue type icon and description in notification e-mails		Resolved
	JRA-15501	Link on Assigned to Me portlet uses a different sort order than the portlet itself		Resolved
	JRA-15488	Shows incorrect time (zone) when subscribing to filter		Resolved
	JRA-15482	EdituserProperties has a key length bug in it on DB2		Resolved
	JRA-15462	Adding user defaults password		Resolved
	JRA-15457	Ajax portlets make remote calls on log dashboard and not shown and not logged in		Resolved
	JRA-15421	Non thread safe use of DateFormat		Resolved
	JRA-15414	Warning on the main downgrade page should be repeated on the child pages.		Resolved
	JRA-15283	incorrect info on the Ent to Pro Downgrad page		Resolved

	JRA-15017	Add summary of modified or removed files to System Info		Resolved
	JRA-11030	Date Time Picker: time element returns the javascript format code (%R).		Resolved

JIRA 3.13.1 Upgrade Guide

Upgrading from JIRA 3.13 to 3.13.1

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.12 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Known Issues

IMAP message handling produces issues with no description

If your JIRA instance is set up to create issues and/or comments from e-mails that come from an IMAP mail box, you will need to deploy a patch to address a problem with issue creation (as described in JRA-15954). Please deploy the patch file attached to JRA-15954 (instructions for deploying the patch are contained in this comment of JRA-15954).

JIRA 3.12 Release Notes

Atlassian Software Systems is proud to present JIRA 3.12.

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

JIRA 3.12 provides a number of enhancements for the upcoming JIRA Studio. Because some of these enhancements may be of benefit to you, we have decided to provide them as a public release rather than making you wait until JIRA 4.0.

The major feature of this release is the ability to 'trust' Confluence. For people who use both JIRA and Confluence, the ability to configure a 'trust' relationship between the two will allow for a seamless end-user experience, e.g. the 'JIRA Issues' macro will now display exactly the same list of issues on a Confluence page that the user would see in the JIRA Issue Navigator. No longer is there a need to hard-code JIRA user names and passwords on a Confluence page. Note that you will need Confluence 2.7 (which is due for release this month) or later.

Also included in 3.12 is a new global permission, '**JIRA System Administrators**'. This will be particularly useful for organisations where the JIRA administrators are not necessarily the same people who are responsible for maintaining the file system and network environment. Granting the 'JIRA System Administrators' permission to only a controlled number of people will give your Windows or UNIX administrators greater peace of mind, while people with the 'JIRA Administrators' permission can enjoy full control over JIRA-specific administration.

Upgrading to JIRA 3.12 is free for all customers with active JIRA software maintenance as at 30 November 2007.

Highlights of JIRA 3.12:

- 'Trusted' Confluence
- 'JIRA System Administrators' permission
- FishEye plugin now bundled with JIRA
- Improvements to the Subversion plugin
- Improvements to the 'Project Statistics' and 'Filter Statistic' portlets
- New post function for workflows: 'Assign to Current User'
- Enhanced language support for searching
- Visual SourceSafe plugin
- Plus more than 100 other fixes and improvements

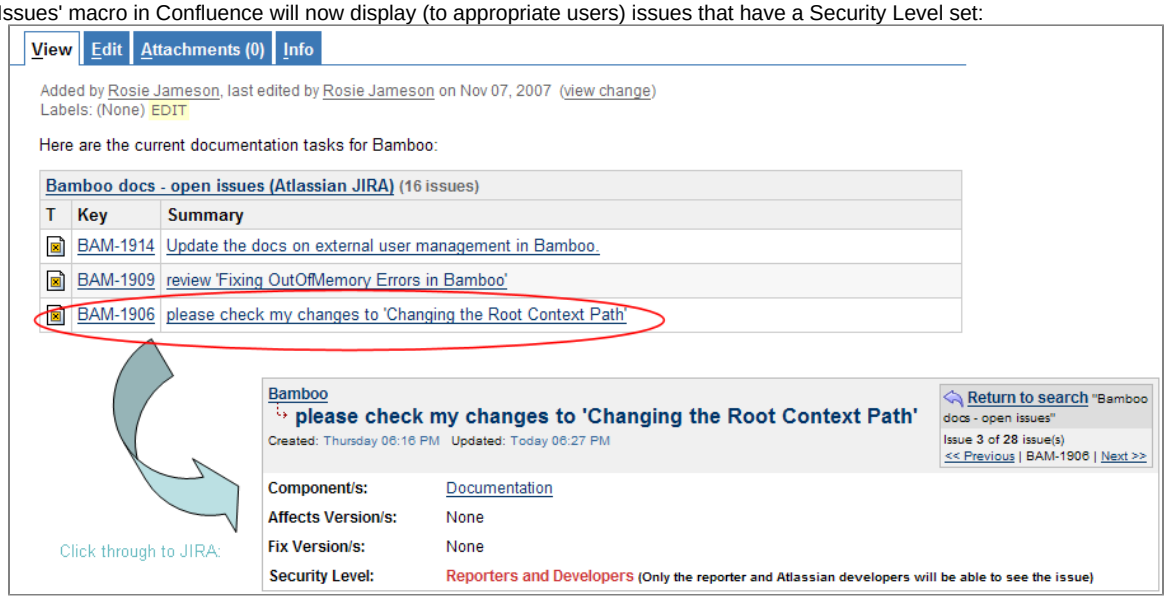
Upgrading to JIRA 3.12
JIRA 3.12 can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.12 Upgrade Guide.

Highlights of JIRA 3.12

'Trusted' Confluence

For people who use both JIRA and Confluence, the ability to configure a 'trust' relationship between the two will allow for a seamless end-user experience, e.g. the 'JIRA Issues' macro will now display exactly the same list of issues on a Confluence page that the user would see in the JIRA Issue Navigator. No longer is there a need to hard-code JIRA user names and passwords on a Confluence page. (Note that you will need Confluence 2.7 or later.)

The 'JIRA Issues' macro in Confluence will now display (to appropriate users) issues that have a Security Level set:



^Top

'JIRA System Administrators' permission

A new global permission has been added to JIRA, to allow for separation of duties.

- People who are granted the new '**JIRA System Administrators**' permission can perform *all* of the administration functions in JIRA, including functions which could affect the application environment or network (e.g. data import/export, SMTP configuration, database connection).
- People with only the '**JIRA Administrators**' permission can now perform most administration functions (e.g. creating new JIRA users; creating projects), but not functions which could affect the application environment or network.

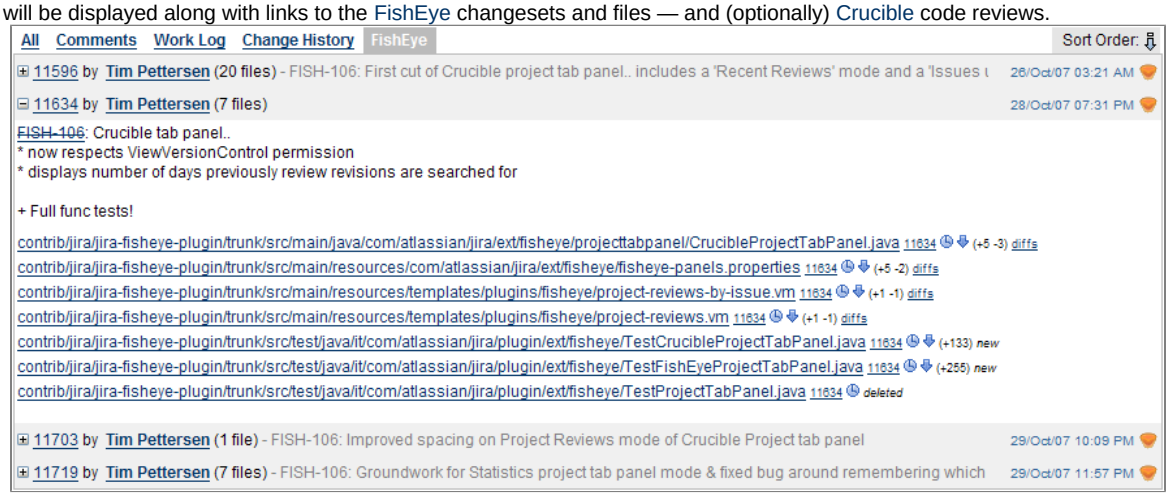
This will be useful for organisations which need to delegate JIRA-specific administration privileges to particular people, without granting them total system administration privileges.

Note that everyone who had the '**JIRA Administrators**' global permission before the upgrade will automatically receive the new '**JIRA System Administrators**' global permission during the upgrade. This will ensure that everyone can still perform the same functions they could previously.

^Top

FishEye plugin now bundled with JIRA

The FishEye plugin automatically detects JIRA issue-keys in your code commit messages. Within your JIRA issues and projects, relevant commit messages will be displayed along with links to the FishEye changesets and files — and (optionally) Crucible code reviews.

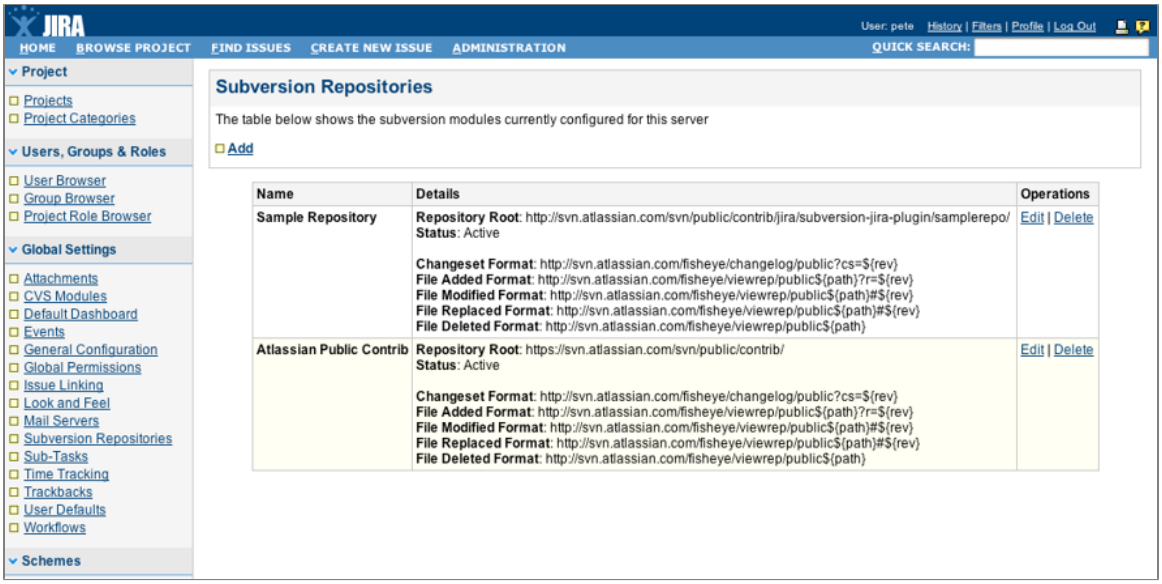


- Click here for a live example.

^Top

Improvements to the Subversion plugin

Using the Subversion plugin (available separately), it is now possible to configure Subversion repositories from within JIRA. No longer do you need to muck around with .properties files and bounce JIRA every time you make a change! If you have existing SVN repositories configured in your properties file, the new SVN plugin will read that information and create identical settings for you.

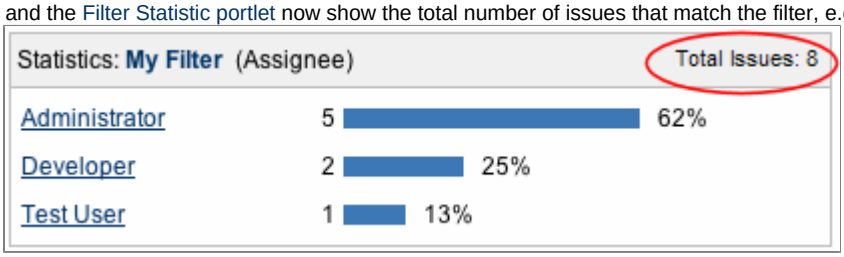


Click to enlarge

^Top

Improvements to the 'Project Statistics' and 'Filter Statistic' portlets

The Project Statistics portlet and the Filter Statistic portlet now show the total number of issues that match the filter, e.g.:



^Top

New post function for workflows: 'Assign to Current User'

With the new post function 'Assign to Current User', you can now automatically assign an issue to the logged-in user when the issue moves through a particular workflow transition.

This is useful if you need to assign an issue to the logged-in user under particular circumstances, but not give them full rights to assign issues. For example, on Atlassian's support system, when a support specialist clicks 'Start Investigating', the issue is automatically assigned to them — even if they don't have 'Assign Issues' permission.

^Top

Enhanced language support for searching

The range of available languages for JIRA search indexes has been expanded. This means that even more people around the world can now choose to have JIRA index their issue data in their native language.

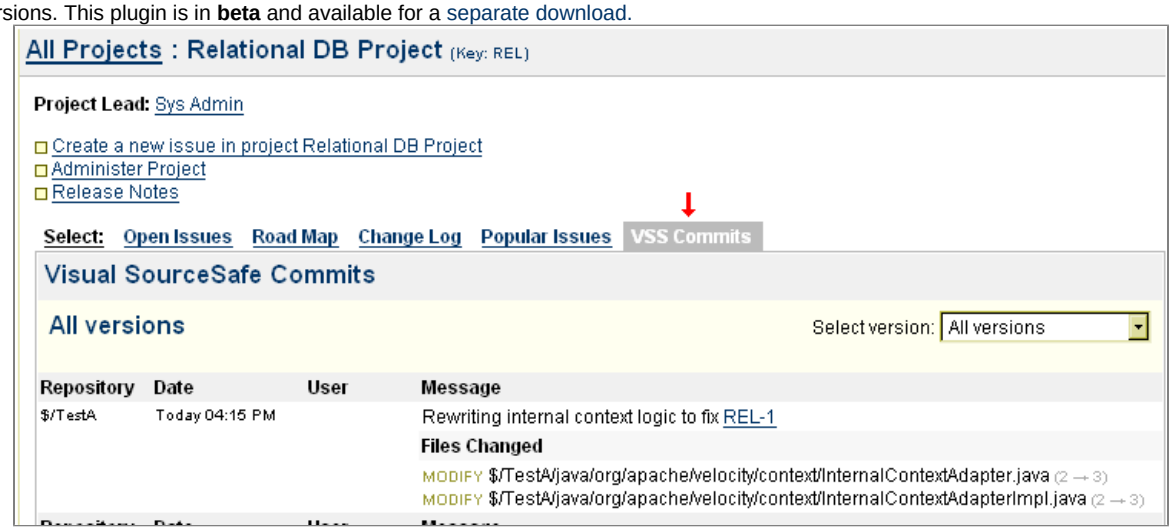
This provides more meaningful search results for end-users, because:

- 'stop' words (i.e. words that are deliberately ignored by the JIRA search engine, such as 'the') are now recognised in more non-English languages (Brazilian, Chinese, Czech, Greek, French, Dutch, Thai).
- 'stemming' (i.e. the derivation of related words, such as 'archived', from a stem such as 'archive) is now supported in French, Brazilian, German, French, Dutch, Russian and English. For example, if your index language is set to French:
  - a search for "marchera" will find "L'enfant a marché"; and
  - a search for "marché" will find "l'enfant marchera".

^Top

Visual SourceSafe plugin

The new VSS plugin displays Microsoft Visual SourceSafe commit information (along with the changed paths) related to JIRA issues, projects or project versions. This plugin is in **beta** and available for a separate download.



^Top

Plus more than 100 other fixes and improvements

JIRA Issues (120 issues)

Key	Type	Summary	Priority	Status
JRA-4085		links in jira issues should be relative ratherr than use the BASE URL		Resolved
JRA-5819		Assign to current user workflow post-function		Resolved
JRA-5900		Error when trying to edit a group		Resolved

JRA-7136	Statistics are wrong when using a filter that searches through comments		Resolved
JRA-8457	Cannot administer my own profile: "The user does not exist. Please try another"		Resolved
JRA-8872	Provide a wrap option with the noformat markup		Resolved
JRA-9171	A timed out session causes stacktraces in half-completed actions		Resolved
JRA-10489	JIRA does not process multipart/alternative mails properly		Resolved
JRA-10508	Insecure "Remember my Login" cookie on https-sites		Resolved
JRA-10515	Adding CSV field to a Mutli User Custom Field causes error		Resolved
JRA-10546	Moving a subtask doesn't inherit the security level of its new parent		Resolved
JRA-10989	WIKI renderer preview does not display correctly		Resolved
JRA-11358	Make plugins configurable		Resolved
JRA-11396	Redundant JIRA Global Permission (Manage Group Filter Subscriptions) in Standard Edition		Resolved
JRA-11446	AccessLogFilter logs everything twice		Resolved
JRA-11767	Allow code and noformat sections in Wiki text to scroll horizontally		Resolved
JRA-11788	Filter window is too small when it opens		Resolved
JRA-11892	Support for Dutch Language		Resolved
JRA-12042	Typo in an RPC API parameter name		Resolved
JRA-12091	Issue's parentId not set on subtask deletion event		Resolved
JRA-12143	Improve doc on hiding fields		Resolved
JRA-12513	Allow NotificationType objects to be registered dynamically		Resolved
JRA-13040	Log critical system operations like reindexes		Resolved
JRA-13090	Multiple file upload fails when the sum of files size exceed the upload limit, even if each file is smaller than the limit.		Resolved
JRA-13128	Save position and size of Filter/History/Help pop up windows		Resolved
JRA-13155	Change renewal hyperlink to use new website redirects		Resolved
JRA-13187	Show Total in Caption of "Filter Statistics" portlet		Resolved
JRA-13188	Trailing + characters are truncuated from URLS in the Description / Environment / Comment fields.		Resolved

JRA-13205	Full content Word export from Issue navigator duplicates wiki style table		Resolved
JRA-13229	controlfooter.jsp always closes the html tr tag even when the nolabel param is set		Resolved
JRA-13231	JIRA installer ships with client JRE instead of server JRE		Resolved
JRA-13263	IssueNotFoundException is thrown right out to the user if a workflow action is taken on a deleted issue		Resolved
JRA-13282	'JIRA System Administrator' Permission		Resolved
JRA-13284	Problem in printing project portlet: red section displayed as white.		Resolved
JRA-13315	Non default permission types can cause Stack overflow if added to wrong permissions		Resolved
JRA-13386	Hide Log Work operation and Work Log tab if Time Tracking Field is hidden in Field Config		Resolved
JRA-13400	Remember me cookie issue with Glassfish; integrate latest Seraph into JIRA		Resolved
JRA-13402	Retain state of attachment comments when switching from single attachment to multiple		Resolved
JRA-13430	When invalid search term was entered in custom field, error message highlights Text Search Query: field.		Resolved
JRA-13436	Small French translation problem		Resolved
JRA-13442	Improve UI for component admin		Resolved
JRA-13465	Right border does is missing on Add Portlet screen on Safari		Resolved
JRA-13473	Double quotes allowed in transition name while editing a transition		Resolved
JRA-13475	Update issue field post function in Create Transition must be the first post function executed in order to actually set the issue field		Resolved
JRA-13509	Special character in group name causes permalink do not function properly		Resolved
JRA-13516	FieldLayoutSchemeImpl caching is not thread-safe		Resolved
JRA-13521	Need to add some unit tests for the bulk edit issue count limit in BulkEdit1.doValidate()		Resolved
JRA-13522	Need to improve the way the calendar-<locale>.js files are served		Resolved
JRA-13523	Multi user custom field cannot be used with the assignable user permission		Resolved
JRA-13542	Format the relative Today and Yesterday as per configured Day Format		Resolved
JRA-13553	Misleading permission violation message when attempting to edit a closed issue		Resolved
JRA-13554	Make Version Workload Report styled like Time Tracking report (nicer)		Resolved
JRA-13558	MailingListCompiler trying to send email with empty "To"		Resolved

JRA-13567	Improvement on Jelly tag documentation		Resolved
JRA-13572	in the 'Add Priority' form, should 'Status Color' be 'Priority Colour' ?		Resolved
JRA-13578	Typo in Issue Navigator in Slovak language		Resolved
JRA-13581	Replace hard-coded string in date pickers		Resolved
JRA-13592	Setting transport to SMTPS in JNDI mail resource is broken		Resolved
JRA-13597	Loading Event Listeners is not synchronized		Resolved
JRA-13598	The EmoticonRendererComponent uses the incorrect IconManager		Resolved
JRA-13613	Time Tracking Report's summary field should be linked		Resolved
JRA-13626	Renderer component does not work with profiling enabled		Resolved
JRA-13654	Allow AttachFile jelly tag to specify the created date for an attachment		Resolved
JRA-13655	The "attach" button in screenshot applet is not translated properly		Resolved
JRA-13656	Time Tracking Label for "Issue"		Resolved
JRA-13666	Missing i18n keys in notification scheme		Resolved
JRA-13673	Admin portlet can show null date for license expirery		Resolved
JRA-13677	The property field for JIRA's portlets and reports are not in order sequence		Resolved
JRA-13687	French Translation Incorrect "traitement"		Resolved
JRA-13699	Deleting a group does not check if there are any worklogs with the group restriction like it does for comments		Resolved
JRA-13703	getFieldsForEdit does not included "Reporter" field, "Due Date" field and "FixVersion" field		Resolved
JRA-13712	user value of JiraAuthenticationContext not set is SOAP service getIssue()		Resolved
JRA-13718	Update AttachFile jelly tag documentation		Resolved
JRA-13724	Caledar popup doesn't work in several languages		Resolved
JRA-13727	Trusted Applications: Support Authentication Context Passing from Confluence or another Application to JIRA		Resolved
JRA-13742	minor grammatical error in bulk move		Resolved
JRA-13744	IssueLevelSecurity permission check does not work with a DocumentIssueImpl if no security level has been set.		Resolved
JRA-13748	Clean View Issue page by moving (View) links for voters and watchers to link on actual value		Resolved

JRA-13750	Help link in browse projects page references version management page in the documentation		Resolved
JRA-13752	Issue Linking docs out of date		Resolved
JRA-13766	Deleting a version can leave gaps in the version sequence		Resolved
JRA-13784	Update Bugzilla import guide to ask users to run Bugzilla 'Sanity Check' tool first		Resolved
JRA-13792	Adding Greek support in Full-Text search		Resolved
JRA-13794	broken link on the 'Trackback Settings' screen		Resolved
JRA-13805	In quicksearch, issue type has higher priority than project key		Resolved
JRA-13809	Add more indexing and search languages		Resolved
JRA-13818	Username with # character breaks on "Assign to me" operation		Resolved
JRA-13823	Move mysql-guide-linux.html page to Confluence		Resolved
JRA-13824	identify entries in site.xml that have no label, and move to CAC where appropriate		Resolved
JRA-13829	Jelly Documentation - error in comment tags		Resolved
JRA-13840	Filter parameters panel on the the left should be collapsed by default when I come to Issue navigator from dashborad plugins		Resolved
JRA-13851	List of available colours for {color} tag in Wiki Style Renderer		Resolved
JRA-13853	No space above the Road Map portlet		Resolved
JRA-13856	Upgrade atlassian-extras for new license types.		Resolved
JRA-13881	Sub-tasks are visibile in Issue Finder while the Parent task is not		Resolved
JRA-13905	Unable to remove group at the Assign Groups to Project Role page if the group name has the double quote		Resolved
JRA-13906	Duplicate i18n-keys in the same language-files		Resolved
JRA-13910	Update the comment in jira-application.properties to indicate that a hyphen should not be used in the project key.		Resolved
JRA-13911	Projects portlet sometimes displays Components and Versions links and sometimes doesn't		Resolved
JRA-13916	"Manage Portal" screen is missing the default template info		Resolved
JRA-13920	Page title is incorrect when the user logs out		Resolved
JRA-13921	Resetting custom version picker field results in incorrect search results		Resolved
JRA-13932	Document [permlink]		Closed

JRA-13935	Anonymous reporter makes rss feeds throw NullPointerException		Resolved
JRA-13941	Add New Issue Type Scheme form does not validate name nicely		Resolved
JRA-13949	Custom Field Type: "Version Picker" - Scroll Bar for displaying List Not Working		Resolved
JRA-13952	Set up redirects for JIRA doc pages moved to Confluence		Resolved
JRA-13974	Review changes to Profiling documentation - new content on making Profiling permanent		Resolved
JRA-13977	Correct Jelly Tags documentation for new permissions in AddPermission		Resolved
JRA-13991	Translation for French and German breaks in the change password screen		Resolved
JRA-13994	Document that users importing from other systems such as via CSV should backup their data first		Resolved
JRA-13998	New german translation is "buggy" concerning Bulkchange		Resolved
JRA-14012	Authenticating security providers fails due to ClassLoader bugs		Resolved
JRA-14022	JIRA Tomcat 6.0 doco changes		Resolved
JRA-14130	charting plugin ignores text query term		Resolved
JRA-14138	User picker still shows user after being removed from all groups		Resolved
JRA-14278	Update documentation - PersistenceManager has to be disabled for Tomcat		Resolved
JRA-15199	Formatting of code sections of LDAP debugging documentation has gone awry		Resolved
JRA-15230	Typo on Downgrading JIRA doc		Resolved
JRA-15587	Link to Issue Type in 'What is an Issue?' page points to the incorrect anchor		Resolved

^Top

JIRA 3.12 Upgrade Guide

Upgrading from JIRA 3.11 to 3.12

Please follow the JIRA general upgrade instructions, plus note the following:

1. Everyone who had the '**JIRA Administrators'** global permission before the upgrade will automatically receive the new '**JIRA System Administrators**' global permission during the upgrade. This will ensure that everyone can still perform the same functions they could previously.
2. The following new Seraph property can be used to fix JRA-10508:

```
<!--  If this parameter is set to true, the cookie will never be set secure.  This is useful if
you're logging
              into JIRA via https, but want to browse JIRA over http.  This flag will ensure
that the remember me option
              works correctly. -->
        <init-param>
            <param-name>insecure.cookie</param-name>
            <param-value>true</param-value>
        </init-param>
```

3. Due to the Seraph upgrade, to fix JRA-10508 all users will be prompted to log in again. This will also affect users who have the 'Remember me' checkbox ticked.
4. If you are building JIRA from source, please note that Maven2 is now required for a build. This is because the JIRA Fisheye Plugin requires Maven2.
5. If you are using the JIRA Toolkit, it is recommended that you upgrade to the latest version in order to fix JRA-13553
6. Please note that the new Trusted Applications feature is not supported on Orion versions prior to 2.0.5. Also note that Resin2 has problems and you will need to update the Resin extra jars.
7. There is a new database table. Please see the following page for details

Upgrading from JIRA 3.10.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Using the Trusted Applications feature with Crowd

Please note that older versions of the Crowd client, (i.e. version 1.2.1 or earlier), can interfere with the correct operation of the Trusted Applications feature. If you are enabling Trusted Applications and using Crowd, please ensure that your Crowd client is **version 1.2.2 or later**.

JIRA 3.12 DB Schema Changes

The table below summarises the changes to the database schema. Please note, that if you have developed any custom utilities which query or modify the JIRA database directly (i.e. without using the JIRA API), please check whether the utilities need to be updated.

New Database Table Table - TRUSTEDAPP

The database table `trustedapp` has been added to schema support Trusted Applications:

It has the following columns:

Column Name	Type
id	numeric
applicationId	long-varchar
name	long-varchar
publicKey	very-long
ipMatch	very-long
urlMatch	very-long
timeout	numeric
created	date-time
createdBy	long-varchar
updated	date-time
updatedBy	long-varchar

The ID column is the primary key.

For a mapping of the above type to your particular database, please see the appropriate `fieldtype-*.xml` file in JIRA's `WEB-INF/classes/entitydefs/` directory.

JIRA 3.12.3 Release Notes

30 April 2008

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.12.3 Release Notes

The Atlassian JIRA team is proud to announce the release of **JIRA 3.12.3** in Standard, Professional and Enterprise editions. This point release includes over 40 bug fixes and improvements which can be viewed below. Click a specific issue to see details of the fix, and to download patches where relevant.

JIRA 3.12.3 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.12 yet?
Take a look at all the new features in the JIRA 3.12 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.12.3 Upgrade Guide.

Updates and Fixes in this Release

JIRA Issues (43 issues)

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due
	JRA-15587	Link to Issue Type in 'What is an Issue?' page points to the incorrect anchor	Andrew Lui [Atlassian]	Andrew Lui [Atlassian]		Resolved	Fixed	Sep 11, 2008	Sep 11, 2008	
	JRA-15230	Typo on Downgrading JIRA doc	Andrew Lui [Atlassian]	Jeff McMillan [Atlassian]		Resolved	Fixed	Jul 14, 2008	Jul 17, 2008	
	JRA-15199	Formatting of code sections of LDAP debugging documentation has gone awry	Andrew Lui [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Jul 07, 2008	Sep 14, 2008	
	JRA-15153	Improve Upgrading JIRA Safely page	Andrew Lui [Atlassian]	Andrew Lui [Atlassian]		Resolved	Fixed	Jun 24, 2008	Jun 26, 2008	
	JRA-15138	Trusted Application "URL Paths to Allow" documentation lists incorrect URL	Andrew Lui [Atlassian]	Jed Wesley-Smith [Atlassian]		Resolved	Fixed	Jun 19, 2008	Jun 25, 2008	
	JRA-15106	<head> tag in the summary causes the HTML page to display wrongly	Unassigned	Timothy Chin [Atlassian]		Resolved	Fixed	Jun 16, 2008	Jun 18, 2008	
	JRA-14827	$comment.roleLevel is not a valid reference errors in logs	Michael Tokar [Atlassian]	Jed Wesley-Smith [Atlassian]		Resolved	Fixed	Apr 17, 2008	Apr 18, 2008	
	JRA-14825	Extra note about installing JIRA on DB2 is needed.	Andrew Lui [Atlassian]	Bogdan Dziedzic [Atlassian]		Resolved	Fixed	Apr 17, 2008	Jun 11, 2008	
	JRA-14749	Start Watching Link is missing in the Italian language pack	Dushan Hanuska [Atlassian]	Terry [Atlassian]		Resolved	Fixed	Apr 03, 2008	Apr 18, 2008	
	JRA-14725	Issue Search Summary has Extra Line Break Under IE7.	Unassigned	Brenden Bain [Atlassian]		Resolved	Fixed	Mar 30, 2008	Mar 31, 2008	
	JRA-14692	Active statuses can be deleted via the URL	Dushan Hanuska [Atlassian]	Stafford Vaughan [CustomWare]		Resolved	Fixed	Mar 24, 2008	Mar 27, 2008	
		Manual Backup Data to XML								

	JRA-14686	fails without warning when filename contains a colon character under WinXP.	Dushan Hanuska [Atlassian]	W. van Beelen		Resolved	Fixed	Mar 21, 2008	Mar 27, 2008	
	JRA-14685	version workload report shows a subtask that is resolved	Dushan Hanuska [Atlassian]	Danita Day		Resolved	Fixed	Mar 20, 2008	Jun 09, 2009	
	JRA-14684	Broken links in Resin 2 documentation	Ian Daniel [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Mar 20, 2008	Apr 21, 2008	
	JRA-14675	Minor errors/typos in documentation	Andrew Lui [Atlassian]	Margaret Votava		Resolved	Fixed	Mar 19, 2008	Jun 25, 2008	
	JRA-14671	Updated French and German Translations	Chris Mountford [Atlassian]	Chris Mountford [Atlassian]		Resolved	Fixed	Mar 18, 2008	Apr 16, 2008	
	JRA-14663	project role name not html escaped	Chris Mountford [Atlassian]	Chris Mountford [Atlassian]		Resolved	Fixed	Mar 18, 2008	Mar 18, 2008	
	JRA-14662	After successfully importing an XML the rendering breaks showing the next page	Michael Tokar [Atlassian]	Kay Nny Lee [Atlassian]		Resolved	Fixed	Mar 18, 2008	Mar 26, 2008	
	JRA-14656	Parent(default) field configuration has overridden setting in issue type field configuration	Michael Tokar [Atlassian]	Chai Ying Chan [Atlassian]		Resolved	Fixed	Mar 17, 2008	Mar 26, 2008	
	JRA-14612	Please update the URL inside JIRA pointing to www.atlassian.jp	Unassigned	Neeraj Jhanji		Resolved	Fixed	Mar 10, 2008	Mar 17, 2008	
	JRA-14594	AddPermission action needs validation on the permission field - throws NPE if not selected	Michael Tokar [Atlassian]	Peter White [Atlassian]		Resolved	Fixed	Mar 05, 2008	Mar 18, 2008	
	JRA-14591	java soap client returns a null for key after creating a remote issue	Timothy Chin [Atlassian]	Lance Selvidge		Resolved	Cannot Reproduce	Mar 05, 2008	May 21, 2009	
	JRA-14579	Using curly braces in the project's name cause to inability to add roadmap portlet to the personal dashboard	Dushan Hanuska [Atlassian]	Alexey Serba		Resolved	Fixed	Mar 03, 2008	Mar 28, 2008	
	JRA-14575	Screenshot link in the exported excel is redirecting to the security breach page	Michael Tokar [Atlassian]	Kay Nny Lee [Atlassian]		Resolved	Fixed	Mar 03, 2008	Apr 15, 2008	
	JRA-14514	Broken version and component links on issues: jsessionid added halfway through URL	Chris Mountford [Atlassian]	Jeff Turner [Atlassian]		Resolved	Fixed	Feb 24, 2008	May 07, 2008	
	JRA-14507	Random password generation for jira:CreateUser	Andrew Lui [Atlassian]	Nicolas Dumazet		Resolved	Fixed	Feb 24, 2008	Jun 11, 2008	
	JRA-14506	Failed Jelly Run does not escape HTML from the Jelly INPUT	Chris Mountford [Atlassian]	Nicolas Dumazet		Resolved	Fixed	Feb 24, 2008	Mar 06, 2008	
	JRA-14495	Cannot remove user from group that starts with number.	Andreas Knecht [Atlassian]	Brenden Bain [Atlassian]		Resolved	Fixed	Feb 21, 2008	Jul 01, 2008	
	JRA-14474	When viewing issues, JIRA hangs for 30s, then renders page without stylesheets when using GZip compression, mod_jk / mod_proxy_ajp and SSL	Unassigned	Jeff Turner [Atlassian]		Resolved	Fixed	Feb 18, 2008	May 28, 2008	
	JRA-14405	"Edit Comment" notification displays user id / name instead of fullName	Michael Tokar [Atlassian]	Alexey Serba		Resolved	Fixed	Feb 06, 2008	Feb 18, 2008	

	JRA-14404	Backup's Cancel button generates Error HTTP Status 404	Michael Tokar [Atlassian]	Jaroslaw L.		Resolved	Fixed	Feb 06, 2008	Feb 12, 2008	
	JRA-14392	Notification Fails Because of Invalid Custom Field	Unassigned	Timothy Chin [Atlassian]		Resolved	Fixed	Feb 03, 2008	Feb 10, 2008	
	JRA-14382	Misleading "Invalid certificate" error message when trusted apps IP address not allowed	Jed Wesley-Smith [Atlassian]	Jeff Turner [Atlassian]		Resolved	Fixed	Jan 31, 2008	May 19, 2008	
	JRA-14338	Bulk Delete does not respect email notification sending for sub-tasks	Andreas Knecht [Atlassian]	Nick Menere [Atlassian]		Resolved	Fixed	Jan 23, 2008	Feb 11, 2008	
	JRA-14009	NullPointerException thrown when deleting an Issue Type which was configured for a non-existent Custom Field	Jed Wesley-Smith [Atlassian]	Michael Tokar [Atlassian]		Resolved	Fixed	Nov 21, 2007	Jun 04, 2008	
	JRA-13939	Printable link for pages may double escape urls	Andreas Knecht [Atlassian]	Nick Menere [Atlassian]		Resolved	Fixed	Nov 11, 2007	Feb 20, 2008	
	JRA-13936	IssueNavigator produces harmless but incorrect urls	Unassigned	Nick Menere [Atlassian]		Resolved	Fixed	Nov 11, 2007	Feb 10, 2008	
	JRA-13635	Calendar date picker can't be used with Hungarian locale	Dushan Hanuska [Atlassian]	Zoltán Mezei		Resolved	Fixed	Sep 28, 2007	Feb 20, 2008	
	JRA-13591	Email should be forwarded if we can't parse it	Unassigned	Nick Menere [Atlassian]		Resolved	Fixed	Sep 23, 2007	Feb 13, 2008	
	JRA-13506	error sending e-mail to persons without an emailaddress	Dushan Hanuska [Atlassian]	Ronald Spierenburg		Resolved	Fixed	Sep 10, 2007	Mar 31, 2008	
	JRA-13093	When service deleted the parameters of this service not deleted (in database)	Dushan Hanuska [Atlassian]	Manuel AUGUSTO		Resolved	Fixed	Jul 13, 2007	Mar 27, 2008	
	JRA-10949	Email prompt shown when email is hidden	Michael Tokar [Atlassian]	Neal Applebaum		Resolved	Fixed	Aug 24, 2006	Feb 18, 2008	
	JRA-10337	more than one "Permission Scheme" being set when use soap	Andreas Knecht [Atlassian]	scud		Resolved	Fixed	Jun 02, 2006	Feb 25, 2008	

JIRA 3.12.3 Upgrade Guide

Upgrading from JIRA 3.12.2 to 3.12.3

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.12.2 Release Notes

21 February 2008

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.12.2 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.12.2 in Standard, Professional and Enterprise editions. This point release includes over 30 bug fixes and improvements, including important security fixes: please see JIRA Security Advisory 2008-02-21 for details. Additionally, the FishEye plugin now supports **trusted applications** for increased security over the existing username and password

authentication. Read more about using the FishEye plugin.

JIRA 3.12.2 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.12 yet?
Take a look at all the new features in the JIRA 3.12 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.12.2 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.12.2 includes the following updates and bug fixes:

JIRA Issues (32 issues)

Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due
	JRA-15587	Link to Issue Type in 'What is an Issue?' page points to the incorrect anchor	Andrew Lui [Atlassian]	Andrew Lui [Atlassian]		Resolved	Fixed	Sep 11, 2008	Sep 11, 2008	
	JRA-15230	Typo on Downgrading JIRA doc	Andrew Lui [Atlassian]	Jeff McMillan [Atlassian]		Resolved	Fixed	Jul 14, 2008	Jul 17, 2008	
	JRA-15199	Formatting of code sections of LDAP debugging documentation has gone awry	Andrew Lui [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Jul 07, 2008	Sep 14, 2008	
	JRA-14802	Installer Documentation for Mac OSX	Andrew Lui [Atlassian]	Matt Hodges [Atlassian]		Closed	Fixed	Apr 15, 2008	Apr 22, 2008	
	JRA-14764	Extra note needed about configuring trust between JIRA and a clustered Confluence	Andrew Lui [Atlassian]	Bogdan Dziedzic [Atlassian]		Resolved	Fixed	Apr 07, 2008	Apr 24, 2008	
	JRA-14584	No option to DEBUG, when configuring LDAP from GUI in JIRA EE	Andrew Lui [Atlassian]	Bogdan Dziedzic [Atlassian]		Resolved	Fixed	Mar 04, 2008	Apr 06, 2008	
	JRA-14512	recommend that the default user which invokes JIRA not be a system account such as root	Andrew Lui [Atlassian]	Peter White [Atlassian]		Resolved	Fixed	Feb 24, 2008	Feb 27, 2008	
	JRA-14414	Trusted authentication doesn't work for Confluence users with uppercase usernames	Jed Wesley-Smith [Atlassian]	Matt Ryall [Atlassian]		Resolved	Fixed	Feb 07, 2008	Feb 13, 2008	
	JRA-14413	TrustedApplicationFilter does not refresh TrustedApplicationsManager dependency when data imported	Jed Wesley-Smith [Atlassian]	Dusan Hornik [Atlassian]		Resolved	Fixed	Feb 06, 2008	Feb 10, 2008	
	JRA-14384	Project name that contains double-quote is not properly escaped on Issue Navigator page	Dushan Hanuska [Atlassian]	Igor Sereda		Resolved	Fixed	Feb 01, 2008	Feb 04, 2008	
	JRA-14351	Tweak JIRA startup so that starting/started is clearly marked	Brad Baker [Atlassian - JIRA BugMaster]	Brad Baker [Atlassian - JIRA BugMaster]		Resolved	Fixed	Jan 25, 2008	Jan 29, 2008	
	JRA-14318	Default issue type is not being pickup	Dushan Hanuska [Atlassian]	Eddie Kua [Atlassian]		Resolved	Fixed	Jan 21, 2008	Jul 10, 2008	
	JRA-14308	JavaScript error on IE when selecting reporter via sub-task's quick create form	Kay Nny Lee [Atlassian]	Riikka Lemminkäinen		Resolved	Fixed	Jan 18, 2008	Feb 05, 2008	

	JRA-14295	NullPointerException when clicking on email filter subscription edit link when not logged into JIRA	Dushan Hanuska [Atlassian]	Dushan Hanuska [Atlassian]		Resolved	Not a bug	Jan 16, 2008	Jan 23, 2008	
	JRA-14290	Make JIRA SOAP client work against jira.atlassian.com	Brad Baker [Atlassian - JIRA BugMaster]	Brad Baker [Atlassian - JIRA BugMaster]		Resolved	Fixed	Jan 15, 2008	Jan 22, 2008	
	JRA-14269	NullPointerException when creating an issue with poorly-configured workflow using AssignToLeadDeveloper function	Unassigned	Masha Khokhlova		Resolved	Fixed	Jan 11, 2008	Jun 18, 2008	
	JRA-14244	UI bug: text wrongly aligned in workflow editor	Andreas Knecht [Atlassian]	Jeff Turner [Atlassian]		Resolved	Fixed	Jan 09, 2008	Jan 10, 2008	
	JRA-14237	NullPointerException when moving portlet on the dashboard page	Dushan Hanuska [Atlassian]	Dushan Hanuska [Atlassian]		Resolved	Fixed	Jan 08, 2008	Feb 17, 2008	
	JRA-14233	Filters link refreshes till out of page	Dushan Hanuska [Atlassian]	Timothy Chin [Atlassian]		Resolved	Fixed	Jan 08, 2008	Sep 11, 2008	
	JRA-14226	The top navigation bars html is outside of the html body tag	Dylan Etkin [Atlassian]	Dylan Etkin [Atlassian]		Resolved	Fixed	Jan 06, 2008	Jan 06, 2008	
	JRA-14215	Exception in user-picker on EACJ	Andreas Knecht [Atlassian]	Steve Smith [Atlassian]		Resolved	Fixed	Jan 03, 2008	Jan 07, 2008	
	JRA-14187	Tomcat documentation should indicate PersistenceManager has to be disabled	Anton Mazkovoi [Atlassian]	Gili		Resolved	Fixed	Dec 27, 2007	Mar 27, 2008	
	JRA-14183	Terrible error reporting when establishing trusted connections	Jed Wesley-Smith [Atlassian]	Jeff Turner [Atlassian]		Resolved	Fixed	Dec 25, 2007	Jan 09, 2008	
	JRA-14178	Missing comment in email after bulk edit	Michael Tokar [Atlassian]	Dieter Wachters		Resolved	Fixed	Dec 21, 2007	Jan 14, 2008	
	JRA-14174	Trackback Link Does Not Have a Base URL	Brad Baker [Atlassian - JIRA BugMaster]	Timothy Chin [Atlassian]		Resolved	Fixed	Dec 20, 2007	Jan 02, 2008	
	JRA-14171	Add whether Crowd is present to the System Info page	Michael Tokar [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Dec 20, 2007	May 25, 2008	
	JRA-14164	Jelly TransitionWorkflow tag in forEach loop sets wrong assignees	Michael Tokar [Atlassian]	Jeff Turner [Atlassian]		Resolved	Fixed	Dec 18, 2007	Jan 02, 2008	
	JRA-14147	Delete notifications cause NPE	Brad Baker [Atlassian - JIRA BugMaster]	David Yu [Atlassian]		Resolved	Fixed	Dec 13, 2007	Jan 02, 2008	
	JRA-14106	Unrecognised icon for file extension (*.docx) and (*.xlsx)	Michael Tokar [Atlassian]	Chai Ying Chan [Atlassian]		Resolved	Fixed	Dec 06, 2007	Feb 25, 2008	
	JRA-13940	url rewrite doesn not rewrite the sort part for datecreated, lastupdated and duedate	Dushan Hanuska [Atlassian]	Nick Menere [Atlassian]		Resolved	Fixed	Nov 11, 2007	Feb 19, 2008	
	JRA-13728	Add a warning on the Time tracking page if the Log Work Issue operation is disabled	Michael Tokar [Atlassian]	Anton Mazkovoi [Atlassian]		Resolved	Fixed	Oct 10, 2007	Jan 07, 2008	
		Can't remove Anyone from	Dushan							

	JRA-13577	jira-users global permission in JIRA 3.10.1 upgraded from earlier version	Hanuska [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Sep 20, 2007	Jan 25, 2008	

JIRA 3.12.2 Upgrade Guide

Upgrading from JIRA 3.12.1 to 3.12.2

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.12.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

24 December 2007

JIRA 3.12.1 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.12.1 in Standard, Professional and Enterprise editions. This point release includes over 10 bug fixes and improvements, including important security fixes: please see JIRA Security Advisory 2007-12-24 for details.

JIRA 3.12.1 is of course free to all customers with active JIRA software maintenance.

Don't have JIRA 3.12 yet?
Take a look at all the new features in the JIRA 3.12 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of JIRA

If you are upgrading, please read the JIRA 3.12.1 Upgrade Guide.

Updates and Fixes in this Release

JIRA 3.12.1 includes the following updates and bug fixes:

JIRA Issues (16 issues)										
Type	**Key**	**Summary**	**Assignee**	**Reporter**	**Priority**	**Status**	**Resolution**	**Created**	**Updated**	**Due**
	JRA-15587	Link to Issue Type in 'What is an Issue?' page points to the incorrect anchor	Andrew Lui [Atlassian]	Andrew Lui [Atlassian]		Resolved	Fixed	Sep 11, 2008	Sep 11, 2008	
	JRA-15230	Typo on Downgrading JIRA doc	Andrew Lui [Atlassian]	Jeff McMillan [Atlassian]		Resolved	Fixed	Jul 14, 2008	Jul 17, 2008	
	JRA-15199	Formatting of code sections of LDAP debugging documentation has gone awry	Andrew Lui [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Jul 07, 2008	Sep 14, 2008	
	JRA-14467	Improvement on Tomcat 6.x, 5.5 documents	Andrew Lui [Atlassian]	Bogdan Dziedzic [Atlassian]		Resolved	Fixed	Feb 17, 2008	Feb 20, 2008	
	JRA-14394	MySQL docs page should highlight a need of setting up the JIRA db with support of unicode	Andrew Lui [Atlassian]	Bogdan Dziedzic [Atlassian]		Resolved	Fixed	Feb 03, 2008	Feb 17, 2008	
	JRA-14347	MSSQL 2005 Documentation is confusing customers	Andrew Lui [Atlassian]	Brenden Bain [Atlassian]		Resolved	Fixed	Jan 24, 2008	Jan 29, 2008	

	JRA-14346	Emphasize that JIRA Standalone can be re-configured to use a "real" database	Andrew Lui [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Jan 24, 2008	Jan 28, 2008	
	JRA-14148	NumberFormatException occurs while trying to set up a trusted application if the	Jed Wesley-Smith [Atlassian]	Jed Wesley-Smith [Atlassian]		Resolved	Fixed	Dec 13, 2007	Jan 02, 2008	
	JRA-14107	[~username] points always to own profile	Michael Tokar [Atlassian]	Patricia Montag		Resolved	Fixed	Dec 07, 2007	Feb 15, 2008	
	JRA-14105	XSS on CreateIssue post	Michael Tokar [Atlassian]	Anton Mazkovoi [Atlassian]		Resolved	Fixed	Dec 06, 2007	Feb 28, 2008	
	JRA-14086	Setup page is accessable after JIRA instance has been setup already	Michael Tokar [Atlassian]	Michael Tokar [Atlassian]		Resolved	Fixed	Dec 04, 2007	Dec 23, 2007	
	JRA-13999	Anyone can delete a filter which is shared with them.	Michael Tokar [Atlassian]	Emily Stumpf [Atlassian]		Resolved	Fixed	Nov 20, 2007	Dec 23, 2007	
	JRA-13842	Add "stay in touch" links to footer	Michael Tokar [Atlassian]	Jonathan Nolen [Atlassian]		Resolved	Fixed	Oct 29, 2007	Dec 11, 2007	
	JRA-13821	RSS feeds from JIRA are displayed in Internet Explorer 7 as raw XML	Michael Tokar [Atlassian]	Ian Daniel [Atlassian]		Resolved	Fixed	Oct 25, 2007	Feb 14, 2008	
	JRA-13788	User name AutoPicker renders a "FALSE" and pushbuttons disappear.	Michael Tokar [Atlassian]	Shari Terry		Resolved	Fixed	Oct 19, 2007	Apr 03, 2009	
	JRA-13032	Improvement on JBoss 4 document	Andrew Lui [Atlassian]	Eddie Kua [Atlassian]		Resolved	Fixed	Jul 06, 2007	Jan 28, 2008	

JIRA 3.12.1 Upgrade Guide

Upgrading from JIRA 3.12 to 3.12.1

Please follow the JIRA general upgrade instructions

Upgrading from JIRA 3.11 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.11 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is delighted to present JIRA 3.11.

Upgrading to JIRA 3.11 is free for all customers with active JIRA software maintenance as at 24th September 2007.This release focuses on time-tracking. Time-tracking data (that is, the estimated and actual time spent on an issue) now includes the issue's sub-tasks. The aggregated time-tracking data is displayed both within individual 'parent' issues and in the Issue Navigator, so it can be easily reported on, exported to Excel, etc.
Being able to track your project's Road Map (scheduled issues) has long been a useful feature of JIRA. But how do you manage programs of multiple, related projects? In JIRA 3.11, the new Road Map portlet shows upcoming milestones across multiple projects of your choice.

Upgrading to JIRA 3.11
JIRA 3.11 can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.11 Upgrade Guide.

Thank you for your feedback:

31 new feature and improvement requests implemented
223 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Highlights of JIRA 3.11:

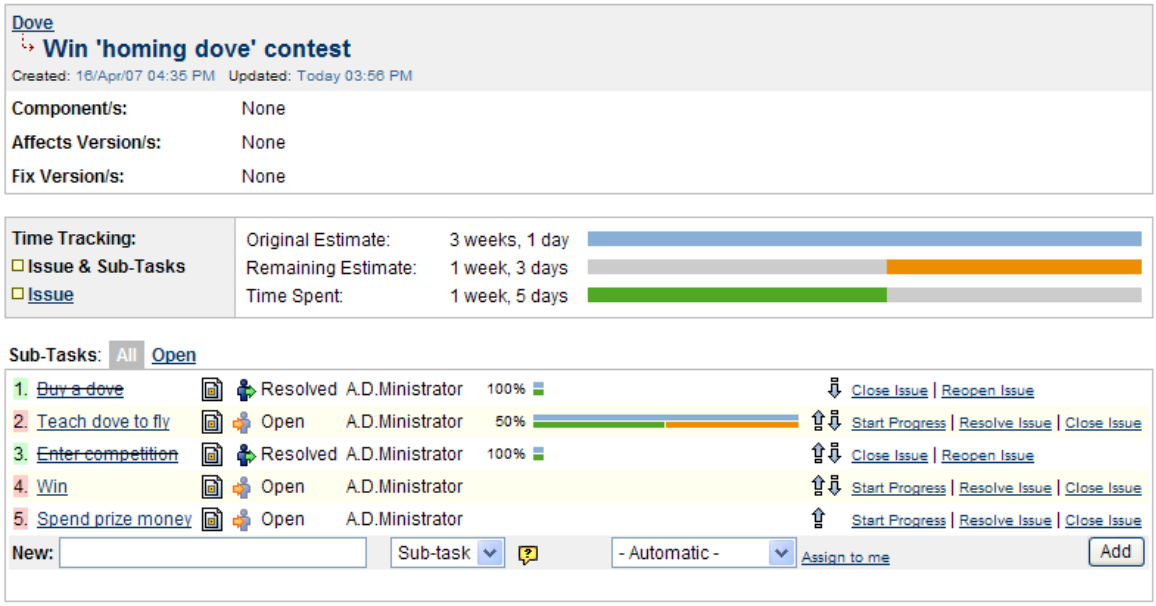
- Sub-task progress shown within issues
- Issue Navigator offers sub-task aggregates
- Time Tracking reports now include sub-tasks
- Multi-project 'Road Map' portlet
- Performance improvements
- Indexing improvements
- JIRA Labels Plugin
- Plus over 70 other fixes and improvements

Highlights of JIRA 3.11:

1

Sub-task progress shown within issues

When viewing an issue, you can now choose whether to view time tracking data for the issue only, or for the issue plus its sub-tasks:



- Click '**Issue**' (in the Time Tracking box) to show time tracking data for the 'parent' issue only, or '**Issue & Sub-Tasks**' to include the issue's sub-tasks.
- Time tracking data comprises:
  - **Original Estimate** (blue) — the amount of time the issue was expected to take to resolve, when it was first created.
  - **Remaining Estimate** (orange) — the remaining amount of time the issue is currently expected to take to resolve.
  - **Time Spent** (green) — the amount of time logged working on the issue so far.

Please note that sub-tasks are only available in the Enterprise and Professional editions of JIRA.

Issue Navigator offers sub-task aggregates

To take advantage of the new sub-task aggregates, the following time tracking fields are now available in the Issue Navigator:

- **Progress** — an issue's Time Spent, as a percentage of the issue's Original Estimate.
- **Progress** — the aggregate time spent on an issue's sub-tasks, as a percentage of the sub-tasks' aggregate Original Estimate.
- **Original Estimate** — the aggregate Original Estimate for an issue's sub-tasks.
- **Remaining Estimate** — the aggregate Remaining Estimate for an issue's sub-tasks.
- **Time Spent** — the aggregate Time Spent for an issue's sub-tasks.

Time Tracking reports now include sub-tasks

- A 'parent' issue now has two separate sets of time tracking data, if applicable: (1) its own; and (2) an *aggregate* that includes the issue's own time-tracking plus all sub-tasks that the user has permission to see.
- The Time Tracking report now includes the aggregate data as shown in the new '' columns:

Report: Time Tracking Report

Description:
This report shows the time tracking details for a specific project.

Excel View

Time Tracking Report for Dove

Only including sub-tasks with the selected version

Progress: **54%** — **2w 2d** completed from current total estimate of **4w 2d**

Accuracy: **0%** — Issues in this version are on schedule with original estimate of 4w 2d.

Key	Summary	Original Estimate	Σ	Est. Time Remaining	Σ	Time Spent	Σ	Accuracy	Σ
DOVE-1	Win 'homing dove' contest	-	4w 2d	-	2w	-	2w 2d	-	on track
DOVE-6	Spend prize money	-	-	-	-	-	-	-	-
DOVE-5	Win	-	-	-	-	-	-	-	-
DOVE-3	Teach dove to fly	4w	-	2w	-	2w	-	on track	-
DOVE-4	Enter competition	1d	-	0m	-	1d	-	on track	-
DOVE-2	Buy a dove	1d	-	0m	-	1d	-	on track	-
	Total		**4w 2d**		**2w**		**2w 2d**		**on track**

- Additionally, both the Time Tracking report and the Version Workload report now include options for choosing which sub-tasks you would like to include in your reports.

Multi-project 'Road Map' portlet

The new Road Map portlet is a handy addition to your JIRA dashboard. It shows upcoming project milestones (i.e. versions which are due for release within a specified period of time), and a summary of progress made towards completing the issues in those versions.

Road Map: **Next 30 Days** (Until 14/Oct/07)

Dove : Version 1 — 01/Oct/07

0 of 3 issues resolved

Test : Version 6 — 08/Oct/07

3 of 6 issues resolved

You can:

- Click the name of a project (e.g. 'Dove') to browse the project.
- Click the name of a version (e.g. 'Version 1') to browse the version.
- Click the progress bar (shown in red and/or green) to view the version's issues in the Issue Navigator.

Performance improvements

JIRA 3.11 includes some significant performance tuning which should improve the experience of every JIRA user.

- Page size has been reduced.
- The effect of GZip compression has been improved due to optimised handling of Javascript and CSS. This will be of benefit to people using Firefox or Internet Explorer 7 browsers.
- Caching has been optimised (see the Developer Blog).

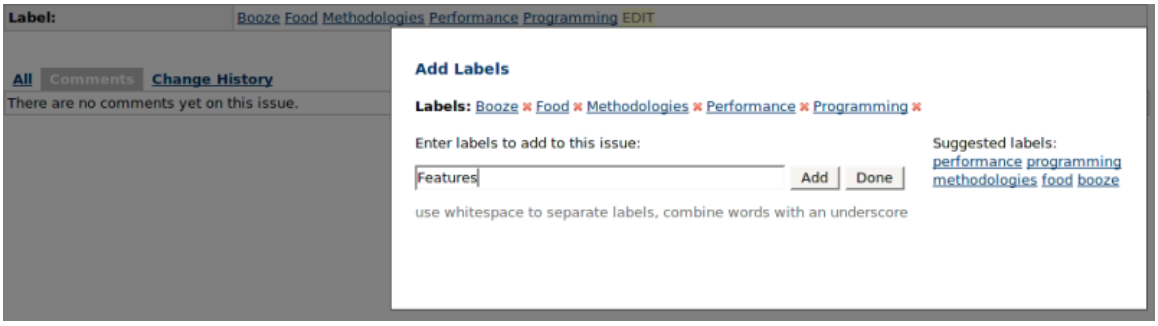
Indexing improvements

- 'Bulk operations' now re-index issues one at a time instead of all at once. This allows JIRA to better handle concurrent operations and higher user loads while maintaining index integrity.
- JIRA now uses Lucene 2.2.0. This has two main benefits:
  - JIRA can now handle the input of and search on dates before January 1st, 1970.
  - JIRA now performs atomic updates to issue and comment indexes, providing greater consistency when searching.

JIRA Labels Plugin

Alongside JIRA 3.11, we're announcing a major overhaul of the JIRA labels plugin. The plugin implements a labels (or tags, sometimes known as folksonomy) custom field for JIRA. Labels or tags make it easier to organise a large set of data by arbitrary, user-defined criteria.

You can read more details on the developer blog.



Plus over 70 other fixes and improvements

JIRA Issues (90 issues)				
Key	**Type**	**Summary**	**Priority**	**Status**
JRA-15056		Editing Version info adds incorrect schedule data .		

			Resolved
JRA-14086	Setup page is accessable after JIRA instance has been setup already		Resolved
JRA-13864	Error is thrown when session timeout expired on editing a comment		Resolved
JRA-13560	Getting permgen OOME on eac/jira during startup		Resolved
JRA-13557	Create Labels build for 3.11		Resolved
JRA-13556	setHeader(null) causing NullPointer on Resin		Resolved
JRA-13510	An extra warning info for the JIRA standalone installation page.		Resolved
JRA-13502	Sub-task creation fails when browse project permission is give to CurrentAssignee, GroupCF or UserCF		Resolved
JRA-13500	Upgrade urlrewritefilter		Resolved
JRA-13496	Error message generated in Add Comment Panel when adding comment to an issue		Resolved
JRA-13491	Only include calendar javascript and css on pages that require the calendar		Resolved
JRA-13490	Ship minified versions of javascript and css files		Resolved
JRA-13489	Don't use scriptaculous loader		Resolved
JRA-13488	Remove dashboard dependency on scriptaculous		Resolved
JRA-13466	TransitionWorkFlow Jelly Tag doco has a bad example attribute - commentLevel		Resolved
JRA-13443	Subtask quick create submit can submit multiple times with multiple clicks		Resolved
JRA-13435	Base URL ending in slash breaks filter subscription HTML email links		Resolved
JRA-13425	Create Road Map Portlet		Resolved
JRA-13422	German translation for "Log work done" ist bad		Resolved
JRA-13417	'None' hard-coded in /templates/plugins/fields/edit/edit-multiselect.vm : should be internationalized		Resolved
JRA-13412	TransitionWorkflow Jelly tag will not work if there is no transition screen		Resolved
JRA-13411	JIRA XML import does not correctly ignore the unicode non-characters \uFFFF and \uFFFE		Resolved
JRA-13408	Pop up History/Filters windows when clicking History/Filters if they are open but under the main window		Resolved
JRA-13387	Upgrade atlassian-extras for the VSS plugin		Resolved
JRA-13382	combined.css and js files are loaded from the cache even after an upgrade of JIRA		Resolved

JRA-13367	Absence of an identifier on the comment element in per-issue generated XML		Resolved
JRA-13364	Wrong onClick location. Differs from href links.		Closed
JRA-13354	Subtask quick creation breaks when spaces available in between the property values.		Resolved
JRA-13345	Jira RSS 2.0 does not work with standard java parser - pubDate elements are incorrect		Resolved
JRA-13343	Problems with 'raw' rssMode when producing XML view of issue filter (in single-xml.vm)		Resolved
JRA-13335	Remove 'back to previous view' link on the printable view from printable media		Resolved
JRA-13332	Concurrent modification exception in com.atlassian.jira.web.tags.UserTag		Resolved
JRA-13319	Remove a link to Excel view from printable view of Time Tracking report		Resolved
JRA-13318	Hardcoded English terms on reports		Resolved
JRA-13295	JIRA standalone does't run as a Windows service - Failed creating java ... jvm.dll		Resolved
JRA-13291	Modify Time Tracking report to include aggregate time information		Resolved
JRA-13260	Make SOAP addWorklog method return the id of the created worklog		Resolved
JRA-13249	JIRA Turkish Language Property Files		Resolved
JRA-13248	Make email address in Support Request success/error page configurable		Resolved
JRA-13243	"Your Watches" translated as "Your Spies" in French....		Resolved
JRA-13227	Hide the priority of linked issues if the priority field is hidden		Resolved
JRA-13219	IntegrityChecker crashes with missing Portlet data.		Resolved
JRA-13210	Description in "Delete Issues" permission is inaccurate		Resolved
JRA-13179	ExceptionInInitializerError executing PopService		Resolved
JRA-13172	Separate out the searchers from the ThreadLocalQueryProfilingFilter		Resolved
JRA-13154	Upgrade to Lucene 2.x		Resolved
JRA-13138	Source release build needs to specify the 'Source' release info by default		Resolved
JRA-13113	BulkEditUserGroups: Provide helpful technique to prune out erroneous entries		Resolved
JRA-13110	SOAP/RPC getIssueTypes() should accept Project ID		Resolved
JRA-13109	Add a license file for the mindprod CSV parser to every distribution of JIRA		Resolved

JRA-13102	Calculate description text field lenght instead of hardcoding to 40 chars		Resolved
JRA-13085	Worklog Service should not allow lightweight issues to be passed in and persistened. It wrecks the index view of the world		Resolved
JRA-13083	Enabling "External user management" should not disable "View Project Roles" on user		Resolved
JRA-13069	Project Administrators do not get to edit "System Default Field Configuration "		Resolved
JRA-13057	UnsupportedOperationException with hasPermissionToCreate when called with DocumentIssueImpl		Resolved
JRA-13054	Display all installed languages and highlight default on the 500, system info and support request page		Resolved
JRA-13049	JIRA crashes when subscribing to a filter, and not logged in.		Resolved
JRA-13044	Upgrade EasyMock and DynaMock libraries		Resolved
JRA-13033	Make font size smaller for version / component descriptions on Versions / Components browse project tabs		Resolved
JRA-13017	Ensure SearchParameters can handle non-GV values in constructor.		Resolved
JRA-12985	Jelly doco for Create Issue tag says that default value of reporter tag is the current logged in user. But it ain't that simple.		Resolved
JRA-12948	Incorrectly reporting Installation type as EAR/WAR instead of Standalone when running as a Windows service		Resolved
JRA-12925	HTML issue event notification emails render poorly in Outlook 2007		Resolved
JRA-12917	Improve on-line documentation on move permission		Resolved
JRA-12912	CommentService validation methods do not check user's security level		Resolved
JRA-12868	Customfield User Picker "corrupted" after a user is deleted		Resolved
JRA-12864	Trivial UI bug		Resolved
JRA-12863	Malformed Ressource Bundle properties files		Resolved
JRA-12839	When session expires and a login is forced the add commet operation loses the original comment text		Resolved
JRA-12837	Bugzilla importer breaks when summary contains over 255 characters		Resolved
JRA-12807	Create clear docs explaining options for import/export project		Resolved
JRA-12790	Allow versions to be rescheduled to an arbitrary position on the Edit Version Details page		Resolved
JRA-12775	investigate getting JIRA working on Tomcat 6		Resolved
JRA-12687	Need to add Tomcat 6.0 documentation		Resolved
JRA-12584	Submitting a Support Request via the admin interface should verify that the mail server has been configured correctly		Resolved

JRA-12469	Improving UrlRewrite to better handle HTTP request parameters		Resolved
JRA-12354	Version Control shows logs of wrong issue		Resolved
JRA-12336	Remove MailSender (no longer used)		Resolved
JRA-11877	Automated JIRA backup failes without proper warning		Resolved
JRA-11588	Replace usage of deprecated class DateField with DateTools		Resolved
JRA-10461	Deleting an issue type breaks custom fields that had used it		Resolved
JRA-10353	Enable GZip compression on SOAP interface		Resolved
JRA-10326	JIRAs HTTP transfers are not efficient - improve web performance		Resolved
JRA-10254	Warning message when an issue is created with 0 bytes or corrupted file		Resolved
JRA-9461	Increase the scope of the gzip compression to include JS and CSS files		Resolved
JRA-9103	Investigate support for pre-1970 dates within JIRA		Resolved
JRA-6332	Comments override updated date in CSV importer		Resolved
JRA-6249	Add more custom icons for issue constants		Resolved
JRA-6007	Can not delete double-byte group in Group Browser		Resolved
JRA-3009	Calculate issue estimates using subtask estimates		Resolved

JIRA 3.11 Upgrade Guide

- Upgrading from JIRA 3.10.x to 3.11
- Upgrading from JIRA 3.9.x and earlier

Upgrading from JIRA 3.10.x to 3.11

Please follow the JIRA general upgrade instructions, plus note the following:

Administrative notes

- To take advantage of the performance enhancements in JIRA 3.11, it is recommended that you enable GZip compression (unless you are using mod_proxy).
- The jira-application.properties file has a new option, 'progress', for the following attribute:

```
jira.table.cols.subtasks
```

  The 'progress' option controls the display of the '**Progress**' field in issues and reports.

- JIRA 3.11 introduces a bug fix for JRA-12354. This means that the CVS and Perforce plugin will perform better at detecting commits for a particular issue key, avoiding partial matches on similar project keys. If users have taken advantage of the previous relaxed key matching, they can revert to the old behaviour by simply setting the following application property in the jira-application.properties file and restarting JIRA:

```
jira.option.key.detection.backwards.compatible=true
```

Plugins

Updating plugins
If you are using any of the following plugins, you will need to update them to their latest versions when performing the upgrade:

- Perforce plugin
- Subversion plugin
- Toolkit Plugin
- Charting Plugin
- RPC Plugin

3rd Party and personal plugins may also be affected (esp. if using lucene to store dates). These will need to be updated as well.

If these are updated after the upgrade (instead of as part of the upgrade), you will need to do a reindex.

A failure to update these plugins will result in lots of errors that look like:

Error 1

```
2007-07-25 15:23:27,553 http-8090-Processor4 ERROR
[charting.charts.createdvsresolved.CreatedVsResolvedChart] Could not create velocity parameters For
input string: "20070725144811"
java.lang.NumberFormatException: For input string: "20070725144811"
 at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
 at java.lang.Long.parseLong(Long.java:415)
 at org.apache.lucene.document.DateField.stringToTime(DateField.java:100)
 at org.apache.lucene.document.DateField.stringToDate(DateField.java:104)
 at
com.atlassian.jira.ext.charting.data.DatePeriodStatisticsMapper.getValueFromLuceneField(DatePeriodStatist
at
com.atlassian.jira.ext.charting.data.OneDimensionalObjectHitCollector.adjustMapForValues(OneDimensionalOb
at
com.atlassian.jira.ext.charting.data.OneDimensionalObjectHitCollector.collect(OneDimensionalObjectHitColl
at org.apache.lucene.search.IndexSearcher$1.collect(IndexSearcher.java:137)
 at org.apache.lucene.search.Scorer.score(Scorer.java:49)
 at org.apache.lucene.search.IndexSearcher.search(IndexSearcher.java:146)
 at org.apache.lucene.search.Searcher.search(Searcher.java:118)
 at
com.atlassian.jira.issue.search.providers.LuceneSearchProvider.search(LuceneSearchProvider.java:111)
...
```

Error 2

```
Caused by: java.lang.NoSuchMethodError:
org.apache.lucene.document.Document.add(Lorg/apache/lucene/document/Field;)V
 at com.atlassian.jira.plugin.labels.LabelSearcher.index(LabelSearcher.java:95)
 at
com.atlassian.jira.issue.index.indexers.impl.DefaultCustomFieldIndexer.addIndex(DefaultCustomFieldIndexer
at com.atlassian.jira.issue.index.IssueDocument.getDocument(IssueDocument.java:34)
 at com.atlassian.jira.issue.index.IssueDocumentBuilderImpl.get(IssueDocumentBuilderImpl.java:14)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$IssueAndCommentCreator.handleIssueIndexing(Sing
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$AbstractIssueAndCommentHandler.indexIssuesAndCo
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer.indexIssuesAndComments(SingleThreadedIssueIndex
at
com.atlassian.jira.issue.index.MultiThreadedIssueIndexer.indexIssuesAndComments(MultiThreadedIssueIndexer
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$2.perform(SingleThreadedIssueIndexer.java:113)
at com.atlassian.bonnie.ConcurrentLuceneConnection.withWriter(ConcurrentLuceneConnection.java:296)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$1.perform(SingleThreadedIssueIndexer.java:107)
at com.atlassian.bonnie.ConcurrentLuceneConnection.withWriter(ConcurrentLuceneConnection.java:296)
 at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer.indexIssues(SingleThreadedIssueIndexer.java:102
at
com.atlassian.jira.issue.index.SingleThreadedIssueIndexer$6.perform(SingleThreadedIssueIndexer.java:219)
```

If you see these errors, please ensure that you are using the latest compatible version of the plugin for 3.11. If there is no supported version for 3.11, please contact the plugin developer via the plugin's homepage.

Developer notes

Modification to SOAP clients
If you have written a SOAP client for any JIRA version prior to 3.11 and are invoking any methods to get `RemoteIssueType` you will encounter the bug JRA-13529. The reason for this is that we have added extra information to the `RemoteIssueType` object that indicates if the issue type is a subTask issue type. To avoid the problem you will need to regenerate your remote object stubs against the updated JIRA 3.11 wsdl.

If you would like your SOAP client to work against multiple versions of JIRA then you need to use the latest stubs that have been generated against JIRA 3.11. You will need to **not** use any of the new functionality and you will need to remember that the `isSubTask` variable in the `RemoteIssueType` objects will be defaulted to `false`.

ThreadLocalQueryProfiler searchers have been moved to ThreadLocalSearcherCache
There may be a number of plugins that reference the ThreadLocalQueryProfiler searcher methods directly. These need to now reference the ThreadLocalSearcherCache.

Lucene Upgrade
We upgraded our version of Lucene to 2.2. If your plugin uses to Lucene to index/read data, please ensure that it works with JIRA 3.11. If you are indexing/reading dates, more than likely it will have broken and you will need to use the new Lucene 2 methods.

Database changes

There were no database changes in this release.

Upgrading from JIRA 3.9.x and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.10 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian is proud to announce **JIRA 3.10**, the latest release of our award winning issue tracking, workflow and project management software.

Major new features include:

- Editable worklogs
- 'Start Date' for worklogs
- New ways to browse Components and Versions
- AJAX-based 'User-picker' and 'Issue-picker'

This release also includes several bug fixes.

To see a list of *all* new features and improvements in this release — ask JIRA!

JIRA 3.10 is a free upgrade for any customer who purchased/renewed JIRA after the 9th of July, 2006. This release can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.10 Upgrade Guide.

Thank you for your feedback

Thank you to all the people who help us improve our products by 'voting' and providing ongoing feedback about what is important to you. JIRA 3.10 resolves nearly 360 of your votes.

In particular, our thanks to all those who voted for JRA-2411 (Ability to edit and remove work logs) and JRA-1959 (Allow date selection for work log). It gives us great satisfaction to deliver these features to you, and we hope you will find them useful (we know we will!)

Editable worklogs

Yes, worklogs can now be edited and deleted — with the issue's 'Time Spent' and 'Remaining Estimate' being adjusted appropriately in both cases.

To ensure that only appropriate people can edit/delete worklogs, four new permissions have been added:

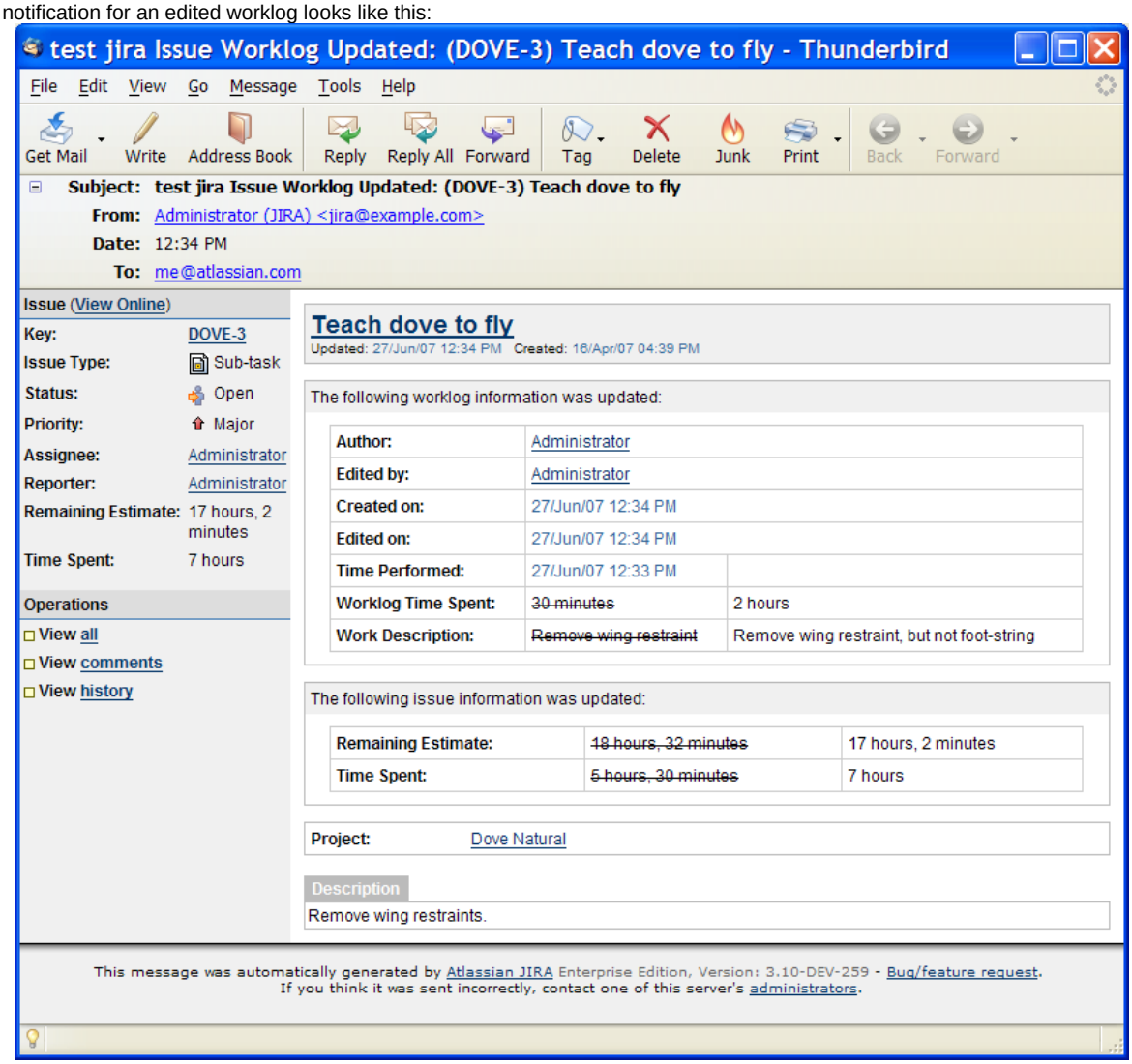
- 'Edit Own Worklogs'
- 'Edit All Worklogs'
- 'Delete Own Worklogs'

- 'Delete All Worklogs'

Similarly, to ensure that email notifications only get sent to the appropriate recipients, there are two new events:

- 'Issue Worklog Updated'
- 'Issue Worklog Deleted'

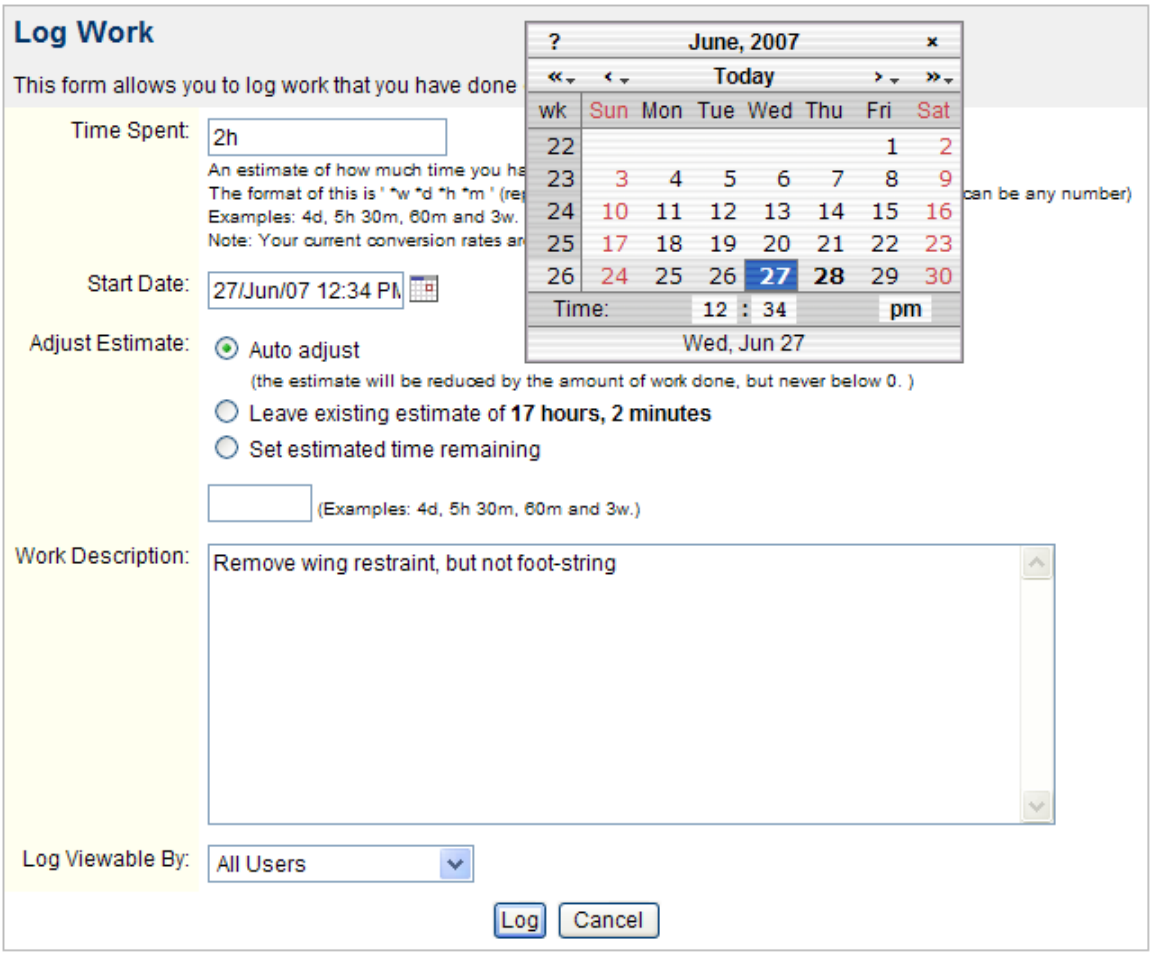
The email notification for an edited worklog looks like this:



All Worklog functions (create, retrieve, update and delete) are also available via the SOAP interface.

'Start Date' for worklogs

When logging work on an issue, you can now specify a 'Start Date'. Simply click the calendar icon to select the date/time when you started work. The calendar popup will be displayed:



... where you can:

- scroll back ('<') or forward ('>') to choose a different date.
- click the hour to increase it (or <Shift> click to decrease it).
- click the minute to increase it (or <Shift> click to decrease it).
- click 'am' / 'pm' to toggle between them.

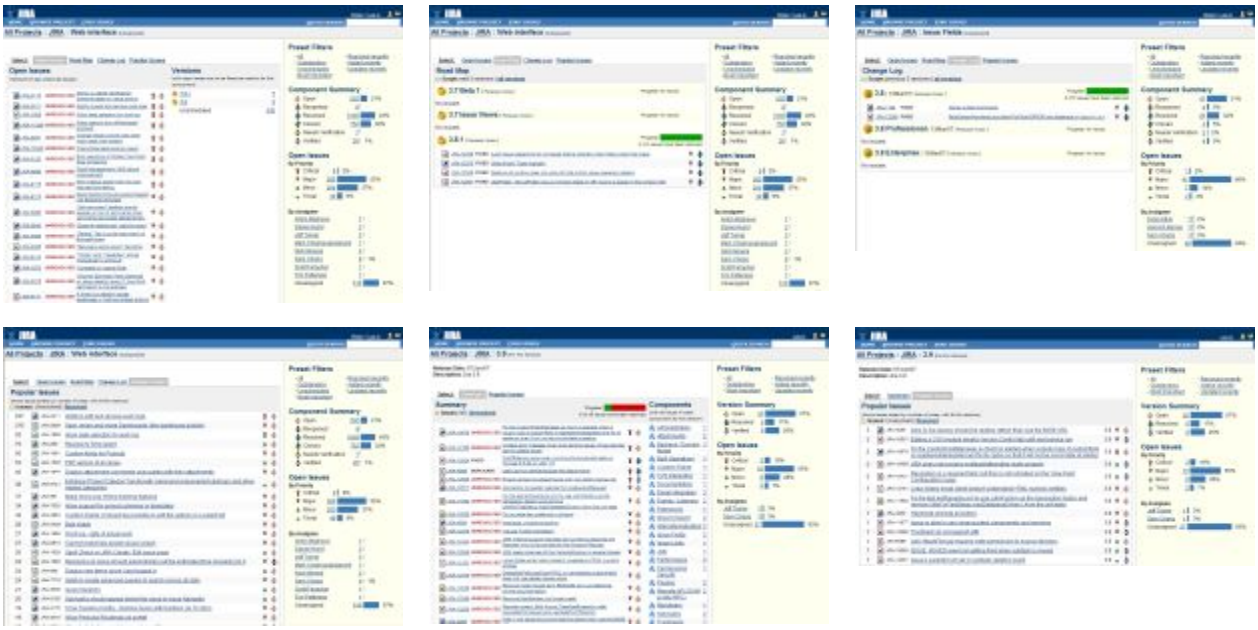
New ways to browse Components and Versions

We're all familiar with browsing a project to see a list of 'Open Issues', 'Popular Issues', and various other screens showing you important statistics about your project.

Now you can drill-down to an individual component or version of a project, by browsing a:

- Component's 'Open Issues'
- Component's 'Road Map'
- Component's 'Change Log' (i.e. resolved issues)
- Component's 'Popular Issues'
- Version's 'Summary' (i.e. all issues for that version, regardless of issue status)
- Version's 'Popular Issues'

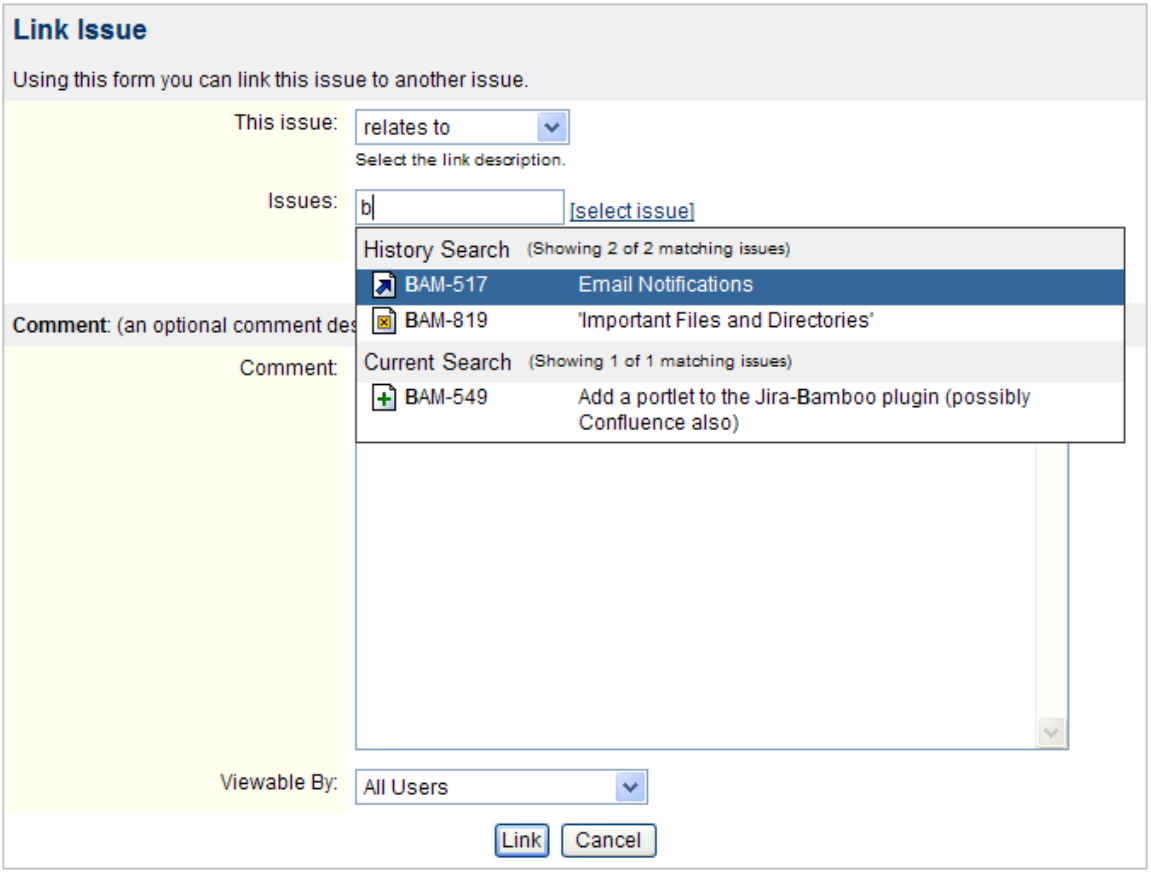
You can give this a try right now on JIRA. Just click on a component or version you're interested in!



The information on the new Component and Version summary pages is displayed using the Component Tab Panel and Version Tab Panel plugins. See the plugin types in the JIRA Plugin Guide for more information.

Auto-complete 'User-picker' and 'Issue-picker'

The 'Issue-picker' and 'User-picker' now have an AJAX-based auto-completion feature:



You're now able to simply start typing a user's name, or an issue's key or summary, and JIRA will provide a drop-down list of possible matches for you to select from. This should make selecting users and issues a lot quicker as you no longer need to click on the 'User-picker' icon or the '[select issue]' link and wait for the relevant pop-ups. The 'Issue-picker' will find matches within your latest search, as well as any matching issues

you've been browsing recently.

This feature is enabled by default (though not for the 'User-picker' if you have more than 5,000 users).

If you wish to disable this feature (e.g. if you have very large numbers of users, or if your users' browsers are incompatible with AJAX), you can easily do so at the 'General Configuration' screen.

^Top

JIRA 3.10 Upgrade Guide

Upgrading from JIRA 3.9.3 to 3.10

Please follow the JIRA general upgrade instructions, plus note the following:

1. Plugins

There is a new version of the JIRA Calendar Plugin that links to the new 'Project Version' pages. This new version of the plugin is not backwards compatible.

Please note that the Kaamelot plugin for JIRA has not yet been updated. If you are currently using this plugin, you may want to hold off the upgrade to JIRA 3.10 until a compatible version of this plugin has been released.

2. Developer Notes

The ordering of the ListOrderedMap returned by SchemePermissions.getSchemePermissions() has changed. This also means that the order of the RemotePermission[] array returned by the RPC Plugin's JiraSoapService.getAllPermissions() method has changed. If you have extended your instance of JIRA please confirm that any remote applications retrieving permissions via SOAP still work. You may encounter problems if you have been retrieving specific permissions by their array index.

Database changes

In JIRA 3.10, the worklog records have moved from the 'jiraactions' database table to the new 'worklog' table. This new table contains the following columns:

```
Table "public.worklog"
    Column    |           Type           | Modifiers
--------------+--------------------------+-----------
 id           | numeric(18,0)            | not null
 issueid      | numeric(18,0)            |
 author       | character varying(255)   |
 grouplevel   | character varying(255)   |
 rolelevel    | numeric(18,0)            |
 worklogbody  | text                     |
 created      | timestamp with time zone |
 updateauthor | character varying(255)   |
 updated      | timestamp with time zone |
 startdate    | timestamp with time zone |
 timeworked   | numeric(18,0)            |
```

Upgrading from JIRA 3.9.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.10.2 Release Notes

JIRA 3.10.2 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.10.2 in Standard, Professional and Enterprise editions. This point release includes 24 bug fixes and improvements.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.10.2 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance since August 17, 2006.

If upgrading, please refer to the JIRA 3.10.2 Upgrade Guide.

Not using 3.10? Learn about all the new features you're missing out on!

JIRA 3.10.2 includes the following bug fixes.

JIRA Issues (25 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-14086	Setup page is accessable after JIRA instance has been setup already		Resolved	Fixed
	JRA-13094	administration - global settings - general configuration: User picker autocomplete option cannot be changed to On		Resolved	Fixed
	JRA-13210	Description in "Delete Issues" permission is inaccurate		Resolved	Fixed
	JRA-12600	CachedGenericConfigManager is not thread safe		Resolved	Fixed
	JRA-13272	Update Comment field on the attach screenshot page not wrapping at word boundaries		Resolved	Fixed
	JRA-13237	The WIKI Help mentions file:/// links buit they not longer work in major browsers		Resolved	Fixed
	JRA-12974	Sorting by work-ratio breaks if no issues have an original estimate		Resolved	Fixed
	JRA-10303	User Picker & Group Searcher Search Template does not respect Issue Type Context when Filtering		Resolved	Fixed
	JRA-13149	Bugzilla importer does not check if entered key is already used by a project		Resolved	Fixed
	JRA-13118	Windows installer uses existing Java JRE rather than built-in JRE		Resolved	Fixed
	JRA-13201	windows service installation script (service.bat) doesn't set the service correctly for JVM		Resolved	Fixed
	JRA-13285	"Ausgabehinweise" is a really really "literal" translation for "Release Notes" *yuck*		Resolved	Fixed
	JRA-13246	Minor typo mistake in English language pack		Resolved	Fixed
	JRA-12646	add 'projectrole' as an option for the 'type' attribute for the 'AddPermission' Jelly tag		Resolved	Fixed
	JRA-13224	Error when editing User Is In Group Custom Field Condition in workflow		Resolved	Fixed
	JRA-13232	Missing active row and columns links in dashboard portlets		Resolved	Fixed
	JRA-13158	2d Portlet loss of functionality to select row		Resolved	Fixed
	JRA-12881	Documentation of 2D portlet misleading		Resolved	Fixed
	JRA-13199	Easy NPE in Browse Version		Resolved	Fixed
	JRA-13209	moveissue.step1.desc in "Move Issue" error message		Resolved	Fixed

	JRA-13302	It is possible to see components without logging in		Resolved	Fixed
	JRA-13166	Empty dropdown box when trying to move subtask and changing Issue Type		Resolved	Fixed
	JRA-13151	Revert to 3.9 behaviour for stats page links for: Assignee, Version, Component, etc.		Resolved	Fixed
	JRA-13165	Calendar Turkish Lang File Cause JavaScript Error		Resolved	Fixed
	JRA-13217	The search description does not link to Browse Fix For Version and Browse Component page when searching versions or components		Resolved	Fixed

JIRA 3.10.2 Upgrade Guide

Upgrading from JIRA 3.10.1 to 3.10.2

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.10.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.10.1 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.10.1 in Standard, Professional and Enterprise editions. This point release includes 26 bug fixes and improvements.

JIRA 3.10.1 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance since August 1, 2006.

If upgrading, please refer to the JIRA 3.10.1 Upgrade Guide.

Not using 3.10? Learn about all the new features you're missing out on!

JIRA 3.10.1 includes the following bug fixes.

JIRA Issues (29 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-13041	System Info should include all system properties		Resolved	Fixed
	JRA-13022	Support Request form does not include database statistics		Resolved	Fixed
	JRA-13025	Make Screenshot applet fake user agent so that it works with Crowd SSO		Resolved	Fixed
	JRA-13198	show Server ID field on the 'Setup Wizard' page		Resolved	Fixed
	JRA-12985	Jelly doco for Create Issue tag says that default value of reporter tag is the current logged in user. But it ain't that simple.		Resolved	Fixed
	JRA-12994	Documentation of Jelly Login tag is misleading/incomplete		Resolved	Fixed
	JRA-13304	Permissions and Conditions should be linked in the documentation		Resolved	Fixed

	JRA-13314	Edited comments appear to come from the original case author and not from the person performing the edit		Resolved	Fixed
	JRA-13096	Unnecessary notification message		Resolved	Fixed
	JRA-13066	Worklog Creation not easily extensible.		Resolved	Fixed
	JRA-13171	Ajax issue picker leaks searchers in threadlocal variable which results in index exceptions on windows when re-indexing		Resolved	Fixed
	JRA-12887	CSV importer wizard breaks when custom field name contains a plus		Resolved	Fixed
	JRA-13117	CSV importer wizard breaks when field name contains unbalanced number of opening and closing brackets		Resolved	Fixed
	JRA-13034	Not very good translation for v.3.10.		Closed	Fixed
	JRA-13059	misnamed I18n properties for Worklog		Resolved	Fixed
	JRA-13120	Incorrect russian translation of summary field in issue navigator		Resolved	Fixed
	JRA-12993	Failed Jelly script displayed with most menus at the top of the screen missing		Closed	Fixed
	JRA-12898	Jelly createIssue tag does not set more than 3 versions		Closed	Fixed
	JRA-13042	OutOfMemoryError in Events and Issue Status admin pages when lots of issue types and workflows		Resolved	Fixed
	JRA-13071	JIRA will not restart after disabling services of the RPC JIRA Plugin		Resolved	Fixed
	JRA-13057	UnsupportedOperationException with hasPermissionToCreate when called with DocumentIssueImpl		Resolved	Fixed
	JRA-13060	Certain multiple select list not having size parameter being specified, causing usability problem to certain browsers (e.g. Opera)		Resolved	Fixed
	JRA-13049	JIRA crashes when subscribing to a filter, and not logged in.		Resolved	Fixed
	JRA-13104	Comment Text Box does not display in IE6 and 7		Resolved	Fixed
	JRA-12936	Login Panel should be dynamic with the width of logo		Closed	Fixed
	JRA-12977	Sub-task icon background is not transparent		Closed	Fixed
	JRA-12989	User Picker minor bug with the display of "Displaying users"		Resolved	Fixed
	JRA-12924	Verbose logging when filter subscription sends to nonexistent group		Closed	Fixed
	JRA-13115	Add Crowd SSO configuration instructions to JIRA's seraph-config.xml file		Resolved	Fixed

JIRA 3.10.1 Upgrade Guide

Upgrading from JIRA 3.10 to 3.10.1

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.9 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian is proud to announce **JIRA 3.9**, the latest release of our award winning issue tracking, workflow and project management software.

New features include:

- Ability to convert sub-tasks to issues (and vice versa)
- Convenient new scheduler for filter subscriptions
- Separate permissions for 'Delete Comment', 'Delete Attachment' and 'Delete Issue'
- Performance Improvements for Project Roles

This release also includes over 30 bug fixes.

To see a list of all new features and improvements — ask JIRA!

494 of your votes have been addressed in this release. As always, thank you for taking the time to cast your vote and tell us what is important to you. We appreciate your feedback.

JIRA 3.9 is a free upgrade for any customer who purchased/renewed JIRA after 9 May, 2006. This release can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.9 Upgrade Guide.

Convert sub-tasks to issues (and issues to sub-tasks)

In JIRA 3.9, sub-tasks can now be converted to issues, and vice versa.

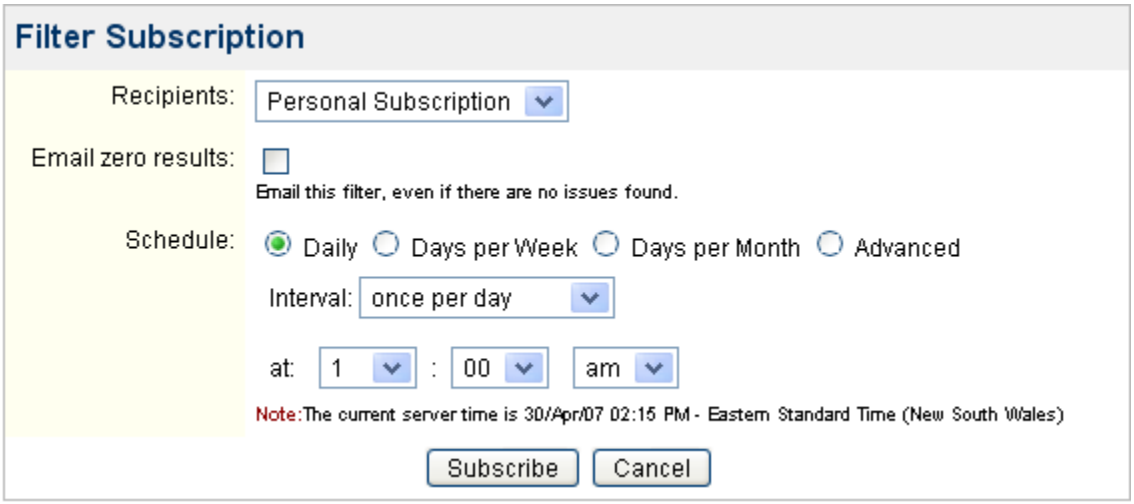
- Perhaps a particular sub-task has become important enough to be an issue in its own right, with its own sub-tasks? Simply go to the sub-task and click 'Convert to Issue'. You can now create sub-tasks for the converted issue.
- Or perhaps an issue should really have been created as a sub-task of an existing issue. No problem: simply go to the issue and — you've guessed it — click 'Convert to Sub-task'.

^Top

Convenient new scheduler for filter subscriptions

If you like to have your search results emailed to you, you will be pleased to see the new and improved scheduler in JIRA 3.9.

It's now even easier to choose exactly when and how often you would like to receive your emails, e.g. 'Every day at 1.00am', 'Every hour between 9.00am and 5.00pm, Monday to Friday'.



For the adventurous, we have also included a cron-based scheduler.

^Top

Separate permissions for 'Delete Comment', 'Delete Attachment' and 'Delete Issue'

You can now give people the ability to delete comments and/or attachments, without giving them the ability to delete entire issues.

JIRA 3.9 has four new project-level permissions:

- 'Delete All Attachments': This permission gives the user the ability to delete any attachments, regardless of who added them.
- 'Delete Own Attachments': This permission gives the user the ability to delete attachments that they created.
- 'Delete All Comments': This permission gives the user the ability to delete any comments, regardless of who added them.
- 'Delete Own Comments': This permission gives the user the ability to delete comments that they created.

Project Role Permission check performance improvements

The performance of permission checks against project roles has been significantly improved.

This improvement allows much faster load times for pages such as Dashboard, especially when several users are hitting JIRA at the same time. The performance improvement is most noticeable with large numbers of projects containing large numbers of project role user members. See JRA-12610 for details.

Previously this check was a CPU intensive operation that involved the intermediate creation of many intermediate objects - and degraded badly under concurrent access. The operation is now performed in constant time per project (basically a hash lookup).

^Top

JIRA 3.9 Upgrade Guide

Upgrading from JIRA 3.8.1 to 3.9

Please follow the JIRA general upgrade instructions. Additionally, please note the following:

In this version, there has been a change to the database which may cause problems for some customers.

The Recommended Upgrade Method

If you follow the recommended export/import upgrade procedure you should not experience any problems!

Pointing JIRA 3.9 at an existing, non-empty database

Some customers have a good reason for not following the recommended upgrade method. Using this method may result in database errors in your logs. You can avoid this if you modify your table structure manually, but the procedure is different depending on whether you have already started JIRA.

To avoid this, **BEFORE** you upgrade JIRA using this method, you can just drop the qrtz_cron_triggers table. This table has not been used by JIRA before 3.9, so it should be empty.

If you have **ALREADY** started JIRA 3.9 using your existing database, you may see the following log messages when JIRA starts up:

```
2007-04-18 15:31:53,345 main WARN [core.entity.jdbc.DatabaseUtil] Column  "CRON_EXPERSSION" of table "
public.qrtz_cron_triggers" of entity  "QRTZCronTriggers" exists in the database but has no
corresponding field
2007-04-18 15:31:53,347 main WARN [core.entity.jdbc.DatabaseUtil] Entity  "QRTZCronTriggers" has 3
fields but table "public.qrtz_cron_triggers"  has 4 columns.
```

The reason for this is that we have incorrectly changed a column in the `qrtz_cron_triggers` table. The intention was to fix a misspelling, but all we did was remove an underscore ("_")! The old column name is "CRON_EXPERSSION". The new column name is "CRONEXPERSSION". Note that both columns spell the word "expression" incorrectly.

To remove the error message, you must remove the old column as it is redundant. This column will not contain any data. The following table shows all columns in the qrtz_cron_triggers table. Columns that should be present are in **green** and columns that should be deleted are in **red**.

Keep	Keep	Keep	Delete
ID	TRIGGER_ID	CRONEXPERSSION	CRON_EXPERSSION

To delete the column, you can use SQL, but this may be slightly different between databases. Here's how it might look:

```
alter table qrtz_cron_triggers drop column CRON_EXPERSSION;
```

The data in this table

If you have users who have subscribed to issue filters, note that existing SimpleTriggers (time intervals) will be automatically converted into CronTriggers during the JIRA upgrade. In some cases, there may not be an exact mapping of time intervals to Cron Expressions, and approximations will be made (e.g. 'Every 5 weeks' will be converted to 'Once a month'). If this happens, the JIRA upgrade process will send an email to the user to inform them of the new schedule.

Upgrading from JIRA 3.8 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.9.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.9.3 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.9.3 in Standard, Professional and Enterprise editions. This point release includes:

- 7 bug fixes.
- professional French and German translations (see below)

JIRA 3.9.3 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance since June 28, 2006.

If upgrading, please refer to the JIRA 3.9.3 Upgrade Guide.

Not using 3.9? Learn about the new features you're missing out on!

What's new in JIRA 3.9.3?

JIRA 3.9.3 includes the following bug fixes and improvements:

JIRA Issues (10 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-12960	UserRoleActorFactory is not null-safe causing weird behaviour at some point, e.g. comment box in view issue screen		Resolved	Fixed
	JRA-12828	"Edit Issue Security Level" behaviour does not match documentation.		Resolved	Fixed
	JRA-12956	Version Management Documentation shows versions in reverse order		Resolved	Fixed
	JRA-12657	Add French and German translations		Resolved	Fixed
	JRA-12889	XSL Stylesheet broken for RSS view		Resolved	Fixed
	JRA-12888	Images in XSL Stylesheet are broken		Resolved	Fixed

	JRA-12855	Resolved issue not showing with strikethrough on Manage links page		Resolved	Fixed
	JRA-12923	UI bug: Choosing "Share with group" select list option does not auto-select the radio button		Resolved	Fixed
	JRA-12833	Subscription integrity checks are misnamed		Resolved	Fixed
	JRA-13030	Bad french translation : before/after avant/après		Resolved	Fixed

Professional French and German translations

The French and German language packs have been completely rewritten and are much more comprehensive than ever before. The administration sections of JIRA are now completely translated. To achieve this, we recently engaged a professional translation company to provide German and French versions of JIRA. These translations are now available in JIRA 3.9.3, and we hope they will make your experience with JIRA even better.

Thank you, danke and merci to all those people who have provided the previous translations over the years, and also to those who have recently been helping us to check the translations for style, consistency and correctness.

While we hope you enjoy the new more comprehensive translations, if the language changes are not ideal for you it is possible to use JIRA 3.9.3 with the old translations. Administrators can revert to the translations from JIRA 3.9.2 and earlier, simply by replacing the new language pack jar file with the corresponding jar file from the earlier version. The French jar file is `language_fr_FR.jar` and the German one is `language_de_DE.jar`, located in `atlassian-jira/WEB-INF/lib` in JIRA standalone.

JIRA 3.9.3 Upgrade Guide

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Upgrading from JIRA 3.9.2 to 3.9.3

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.9.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.9.2 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.9.2 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.9.2 in Standard, Professional and Enterprise editions. This point release includes 13 bug fixes.

JIRA 3.9.2 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance since June 18, 2006.

If upgrading, please refer to the JIRA 3.9.2 Upgrade Guide.

Not using 3.9? Learn about all the new features you're missing out on!

JIRA 3.9.2 includes the following bug fixes.

JIRA Issues (13 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-11284	When processing multiple "bugs" at the same time, the "Change Comment: " button is			Fixed

		no longer being "checked" automatically.		Resolved	
	JRA-12798	Edit Comment email notification mixes up updater with original commenter		Resolved	Fixed
	JRA-12871	CVS Log Handler no longer generates Issue Commented notifications		Resolved	Fixed
	JRA-12797	Edit Comment text email notification does not list 'Edited on' and 'Edited by' details		Resolved	Fixed
	JRA-12632	NullPointerException when search request is made during an import		Resolved	Fixed
	JRA-12785	windows installer failing to set windows service correctly		Resolved	Fixed
	JRA-12830	untranslatable parts to issue comment edited emails templates		Resolved	Fixed
	JRA-12779	Bad i18n messages		Resolved	Fixed
	JRA-12893	Applications Portlet Does Not use Resource Bundle for Display "Lead" text		Resolved	Fixed
	JRA-12403	"Manage attachments" not issue-operation-aware (attach screenshot)		Resolved	Fixed
	JRA-12799	Compiling the 'Project Table' portlet with resin using JDK 1.6 does not work		Resolved	Fixed
	JRA-12631	Exception thrown from xml rpc servlet during import gives misleading error.		Resolved	Fixed
	JRA-12784	User Picker does not handle single quote (') correctly		Resolved	Fixed

JIRA 3.9.2 Upgrade Guide

Upgrading from JIRA 3.9/3.9.1 to 3.9.2

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.8.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.9.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.9.1 Release Notes

Recommended Upgrade

JIRA 3.9.1 contains a security update and is highly recommended. This fix is related to issue level security schemes and a way that users can see details of issues that they are not meant to. If you do not have issue level security schemes or do not use Project Roles you do not need to worry. If you do we recommend you upgrade immediately.

Note that the bug list below does not contain details of the bug as it would reveal how to exploit it as well.

For installations running 3.7.x or 3.8.x who cannot upgrade to 3.9.1, there is a patch available.

Atlassian Software Systems is proud to announce the release of JIRA 3.9.1 in Standard, Professional and Enterprise editions. This point release includes 10 bug fixes and some internationalisation improvements.

JIRA 3.9.1 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance since May 30, 2006.

If upgrading, please refer to the JIRA 3.9.1 Upgrade Guide.

Not using 3.9? Learn about all the new features you're missing out on!

JIRA 3.9.1 includes the following bug fixes.

JIRA Issues (11 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-12671	Unassociated calculated fields are displayed on confirmation pages of Move and Convert issue, even when not in context		Resolved	Fixed
	JRA-12741	Multiselect Search Template doesn't honor HTML		Resolved	Fixed
	JRA-12494	Work Description on Work Logged Notification Email do not preserve carriage returns		Resolved	Fixed
	JRA-12843	Add a link to manage subscription in the subscription email		Resolved	Fixed
	JRA-12694	Update the code signing certificate used for the installer and screenshot applet		Resolved	Fixed
	JRA-12720	Component is always swapped while being deleted		Resolved	Fixed
	JRA-12393	Comment field jumps on "Preview" click when wiki-markup is enabled		Resolved	Fixed
	JRA-12667	DefaultRoleActor and ProjectRoleActor implementations should not hold references to the ProjectRole and Project objects		Resolved	Fixed
	JRA-12528	Can't remove an user from a custom project role		Resolved	Fixed
	JRA-12481	When viewing a saved filter, the 'Current View' links are incorect		Resolved	Fixed
	JRA-12737	Browse Project visual selection status is lost when selecting project tab panels		Resolved	Fixed

Important Security Patch for JIRA versions 3.7.x & 3.8.x

Please find attached an important security patch for JIRA 3.7.x and 3.8.x

If you are using Project Roles and have Issue Security schemes that use Project Roles and you cannot upgrade to 3.9.1 then **you must install this patch**.

Installation Instructions.

If you are using JIRA Standalone please do the following:

1. Download the attached patch zip file
2. Extract the contained files to <jira_install_dir>/atlassian-jira/WEB-INF/classes/ overwriting the files there
3. Restart JIRA

If you are using the WAR distribution of JIRA:

1. Download the attached patch zip file
2. Extract the contained files to <jira_install_dir>/atlassian-jira/WEB-INF/classes/ overwriting the files there
3. Run 'build.sh clean' on unix or 'build.bat clean' on windows
4. Run 'build.sh' on unix or 'build.bat' on windows
5. Redeploy the JIRA web app into your application server
6. Restart the application server

JIRA 3.9.1 Upgrade Guide

Upgrading from JIRA 3.9 to 3.9.1

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.8.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.8 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian is proud to announce **JIRA 3.8**, the latest release of our award winning issue tracking, workflow and project management software.

New features include:

- Editable comments
- Self-installer for JIRA
- CAPTCHA for new account signup
- Integration with Crowd
- Improvements to the Bugzilla importer

Feature preview:

- DHTML-loading of Issue screens

This release also includes more than 30 bug fixes.

To see a list of all new features and improvements — ask JIRA!

Weblogic Users

Please note that there is a known Weblogic and Firefox issue that affects JIRA 3.8.x when using Weblogic and Firefox. See the issue for more detail.

Upgrading

JIRA 3.8 is a free upgrade for any customer who purchased/renewed JIRA after 9 March, 2006. This release can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.8 Upgrade Guide.

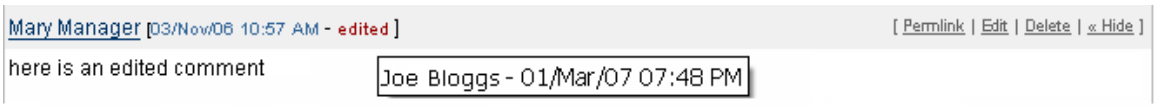
Editable comments

SPECIAL NOTE: Thank you to the 175 people who voted on this feature request. Your input is vital to planning the JIRA development roadmap, and we appreciate you taking the time to tell us what is important to you.

Issue comments can now be edited. To ensure that comments can only be edited appropriately, two new project permissions have been provided so that you can restrict the ability to edit comments:

- 'Edit Own Comments' -- this allows users to edit comments which they have created. This permission is typically granted to end-users.
- 'Edit All Comments' -- this allows users to edit comments which other people have created. This permission is typically granted to administrators.

If a comment has been edited, the word 'Edited' will appear in the comment trail. You can hover your mouse over the word 'Edited' to see who edited the comment and when, e.g.:



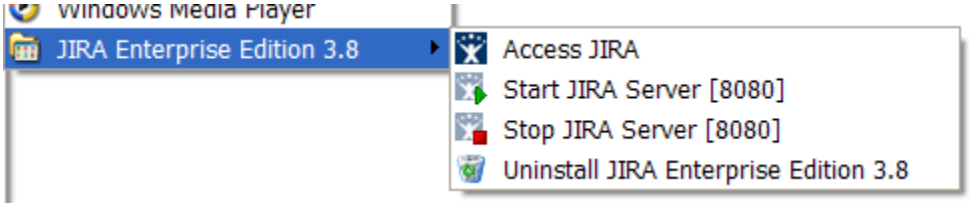
You can also configure email notifications to be sent when the 'Comment Edited' event occurs.

^Top

Self-installer for JIRA

New and existing customers can get the latest version of JIRA up and running on Windows within minutes, using our new installer. No more setting environment variables, installing Java, and running things from the command line. Even novice users will be able to access JIRA in as little as 5 clicks after the download.

For your convenience, we have even added controls to the Start Menu to make life as easy as possible:



We also bundle JIRA with the latest Sun JRE (Java 6.0), so there is no need for a separate download and installation. It all comes packaged and ready to run!

- Self extracting -- no need for Winzip or any other tools.
- Optional installation as a Windows Service.
- 'Start' and 'Stop' menu items, for both normal installation and Windows Service installation.
- Tested on Windows Vista.
- Detection of any other JIRA instances installed on the same machine.
- Port detection (detects if any other web servers are running on the machine and resolves conflicts).
- Uninstaller (not that you will be needing it).

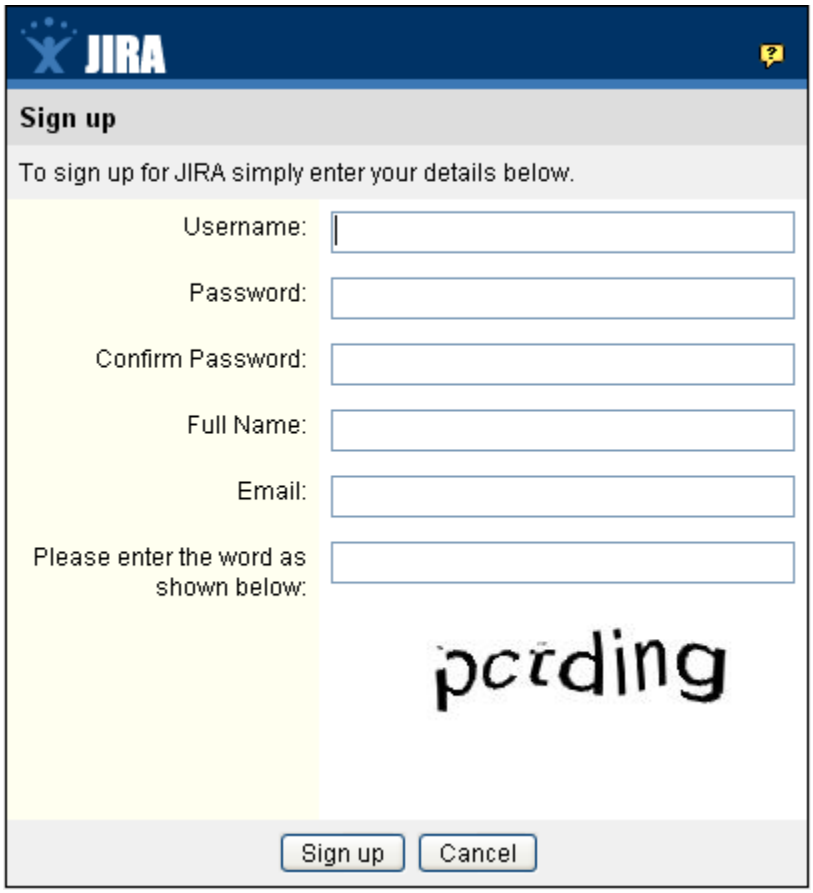
^Top

CAPTCHA for new account signup

If your JIRA server is accessible from outside your organisation's firewall, and you have enabled signup, then you may want to also enable *CAPTCHA*.

CAPTCHA helps ensure that only real humans (and not automated spam systems) can sign themselves up to JIRA. When CAPTCHA is enabled, visitors will need to recognise a distorted picture of a word (e.g. "pctding" in the screenshot below), and must type the word into a text field. This is

easy for humans to do, but very difficult for computers. We are hoping that this feature will help to [fight] evil JIRA spammers (see JRA-12293 for some of the background to this.)



We recommend anyone running a public JIRA instance (e.g. Codehaus, Apache, OpenSymphony) to enable this feature.

^Top

Integration with Crowd

JIRA can now be integrated with Atlassian Crowd, which is useful for organisations that have multiple user-repositories.

- How to integrate Crowd with JIRA

^Top

Improvements to the Bugzilla importer

JIRA's Bugzilla importer has been enhanced. When importing Bugzilla bugs and creating corresponding issues, JIRA will now:

- create Issue Links of type 'Duplicate' between issues that have been imported and marked as duplicates in Bugzilla. The 'Duplicate' link type will be automatically created if it doesn't exist.
- import Component Lead information.
- concatenate the 'URL' field (from Bugzilla) to the 'Environment' field in JIRA issues.

Many thanks to Vladimir Alexiev for his contributions.

^Top

DHTML-loading of Issue screens (Feature Preview)

To facilitate faster loading of issue screens (e.g. the "Edit Issue" screen and the "Resolve Issue" screen), we are working on a feature that will allow JIRA to re-load only those parts of the screen that have changed. We hope this will save a little of your valuable time, and improve your experience with JIRA.

This feature is shipped in JIRA 3.8, but as it has a few known problems (JRA-12348 and JRA-12349) it is disabled by default. The known problems should not affect many users, so we encourage you to turn it on and provide any feedback by adding comments to this page. We would especially like to hear if you believe the feature is useful or if you find any problems that we are not aware of.

To enable the feature, please navigate to Administration -> General Configuration and enable the 'Dynamic HTML for issue screens' option.
^Top

Feedback for DHTML-loading of Issue screens

Please add any feedback you have about the 'DHTML-loading of Issue screens' in JIRA 3.8 as a comment to this page.

We would be very interested to know whether you think the feature is useful and hear about any problems that you find.

Currently we know about the following issues:

1. JRA-12348
2. JRA-12349

JIRA 3.8 Upgrade Guide

Upgrading from JIRA 3.7.4 to 3.8

Please follow the JIRA general upgrade instructions. Additionally, please note the following:

1. The 'Assign To' field name has been changed to 'Assignee' consistently across JIRA. This means that users need to be aware that the column heading in the Excel export has changed to 'Assignee' from 'Assign To'. Please be aware of this if for example you are exporting JIRA data to Excel and running macros on it. The field has been renamed for the following Issue Navigator Views:
    - Excel (all)
    - Word (all)
    - Full Content
2. The `issuecommentedited.vm` e-mail template for the new Issue Comment Edited event has been added to the `WEB-INF/classes/email-template-id-mappings.xml` file. The id of the e-mail template used for sending Filter Subscriptions has changed to 10000. **If** you have manually modified the `WEB-INF/classes/email-template-id-mappings.xml` file in the version of JIRA you are upgrading from, please do not simply copy the old file to JIRA 3.8. You will need to port your changes to the `WEB-INF/classes/email-template-id-mappings.xml` file that is shipped with JIRA 3.8. If you have not changed the `WEB-INF/classes/email-template-id-mappings.xml` file, you do not need to worry about this.
3. Two columns have been added to the `jiraaction` table to support editable comments.

Upgrading from JIRA 3.7.3 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.8 Database Schema Changes

The table below summarises the changes to the database schema. Please note, that if you have developed any custom utilities which query or modify the JIRA database directly (i.e. without using the JIRA API), please check whether the utilities need to be updated.

New Database Table Columns

The following database columns have been added to the existing `jiraaction` table to support editable comments:

TABLE NAME	NEW COLUMN NAME
jiraaction	updateauthor
jiraaction	updated

JIRA 3.8.1 Release Notes

JIRA 3.8.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.8.1 in Standard, Professional and Enterprise editions. This point release includes 23 bug fixes and improvements, notably:

- **Integrity Checker can cause data corruption** — - JRA-12491
- **SSH Connection to CVS in CVS plugin does not close stdout** — JRA-12480 particular thanks go to David Delbecq from the Royal Meteorological Institute of Belgium for help finding this one.
- **OutOfMemoryErrors when reindexing if large numbers of custom fields and issues** — a thread-local cache for custom field values was expanding unbounded when reindexing all issues - JRA-12411

JIRA 3.8.1 can be downloaded here, and is of course free to all customers who purchased their JIRA licence or maintenance within the last 12 months.

If upgrading, please refer to the JIRA 3.8.1 Upgrade Guide.

Not using 3.8? Learn about all the new features you're missing out on!

JIRA 3.8.1 includes the following bug fixes.

JIRA Issues (23 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-12338	If a service is attempted to be loaded that JIRA cannot find it loads the UnloadableJiraServiceContainer instead which interferes with shutdown		Resolved	Fixed
	JRA-12411	OutOfMemoryError during reindex all (due to EagerLoadingOfbizCustomFieldPersister's caching of custom field values)		Resolved	Fixed
	JRA-12410	Deleting a custom field which has an issue security scheme or permission scheme on it causes system error		Resolved	Fixed
	JRA-12480	CVS module of JIRA closes STDOUT		Resolved	Fixed
	JRA-12345	To provide easier configuration between Crowd and JIRA the attached crowd-ehcache.xml file will need to be added to the jira release		Resolved	Fixed
	JRA-12507	Update splitting JIRA instances doc to mention attachments		Resolved	Fixed
	JRA-12501	Update the mail documentation to mention that system properties override mail server settings		Resolved	Fixed
	JRA-12343	Mail is not correctly finding a user that exists - then User creation fails with DuplicateEntityException		Resolved	Fixed
	JRA-12491	Caching in integrity checker can make integrity checker corrupt workflow data		Resolved	Fixed
	JRA-11641	Please allow ability to set mail encoding separtely from html/DB enoding		Resolved	Fixed
	JRA-12565	Example CloseParentIssueFunction		Closed	Fixed
	JRA-12398	Clicking on user prompt in Manage Watch List in Internet Explorer 7.0.5730.11 results in an error		Resolved	Fixed
	JRA-12371	Screenshot applet link available in linux but cant paste the image from the clipboard		Resolved	Fixed
	JRA-12339	Charting Portlets not being rendered in Printable View		Resolved	Fixed
	JRA-11809	Component Deletion is not logged in tickets which had their component changed (audit trail issue for SOX)		Resolved	Fixed
	JRA-12335	Project Role Modifications not reflected in Issue Security Scheme		Resolved	Fixed
	JRA-12381	Data anonymiser does not blank out SMTP server username and password		Resolved	Fixed
	JRA-11858	Remove the predefined "Users" role		Resolved	Fixed
	JRA-12349	Resizing of window does not work with the AJAXy issue operation screens			Fixed

				Resolved	
	JRA-12350	CSS is not correctly applied to pages loaded via AJAX issue operation links		Resolved	Fixed
	JRA-12372	Order Event Types logically		Resolved	Fixed
	JRA-12387	UserPicker / GroupPicker pop-up windows break on IE7 due to a space in the window title		Resolved	Fixed
	JRA-12348	AJAX Issue operations do not resize iframe correctly when there is text that wraps		Resolved	Fixed

JIRA 3.8.1 Upgrade Guide

Upgrading from JIRA 3.8 to 3.8.1

Please follow the JIRA general upgrade instructions.

> **Charting Plugin must be upgraded to v1.3.5**
>
> Please note that the version of JFreeChart included in JIRA 3.8.1 is **not compatible** with older versions of the Charting Plugin. If you have the Charting Plugin installed, please make sure you upgrade it to version 1.3.5 or above.
>
> The updated JFreeChart 1.0.4 version is **not** backwards compatible with the previous 1.0.0pre2 version, so if you have any plugins that utilise JFreeChart, please make sure you test them before upgrading.

Upgrading from JIRA 3.7.4 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.7 Release Notes

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

Atlassian is proud to announce **JIRA 3.7**, the latest release of our award winning issue tracking, workflow and project management software.

New features include:

- Project Roles
- 'Charting' View for Issue Navigator
- RSS Improvements
- User Properties
- SVN Project Panel plugin
- SVN Commit Acceptance plugin
- And much more...

This release also includes over **100** bug fixes and **60** improvements. This version is a free upgrade for any customer who purchased/renewed JIRA after 13 December, 2005.

Upgrading

The 3.7 release can be downloaded from the JIRA Download Center. Before upgrading, please refer to the JIRA 3.7 Upgrade Guide.

Project Roles

In JIRA 3.7, you can configure project roles (e.g. developer, tester, administrator), and assign users/groups to these roles on a per-project basis. If you have more than a few projects, this will significantly simplify administration.

Users can belong to different project roles in different projects, e.g.:

Manage Project Role Membership for Project: ABC

On this page you can manage project role membership for the **ABC** project.

Role	Users
Administrators A project role that represents administrators in a project	Mary Manager Edit
Developers A project role that represents developers in a project	Peter Programmer Edit
Users A project role that represents users in a project	Sally User Edit

Manage Project Role Membership for Project: DEF

On this page you can manage project role membership for the **DEF** project.

Role	Users
Administrators A project role that represents administrators in a project	Peter Programmer Edit
Developers A project role that represents developers in a project	Sally User Edit
Users A project role that represents users in a project	Mary Manager Edit

Ease of management
Does your system currently contain multiple, project-specific groups? Once you upgrade to JIRA 3.7, your permission schemes and notification schemes can use project roles instead of groups. By implementing project roles, you may be able to greatly reduce the number of groups, permission schemes and notification schemes in your JIRA system.

Tools are provided to help you migrate your permission schemes and notification schemes from using groups to using project roles, for example:

The old

The new

Delegated administration
In JIRA Enterprise, a project administrator* can assign users and groups to project roles for their project. If their project's permission scheme and notification scheme are using project roles, the project administrator can control who may access their project and who receives email notifications. In Professional and Standard Editions the global administrator permission is required to manage role membership of a project.

Global access to JIRA is still controlled via groups, which are managed by JIRA global administrators.

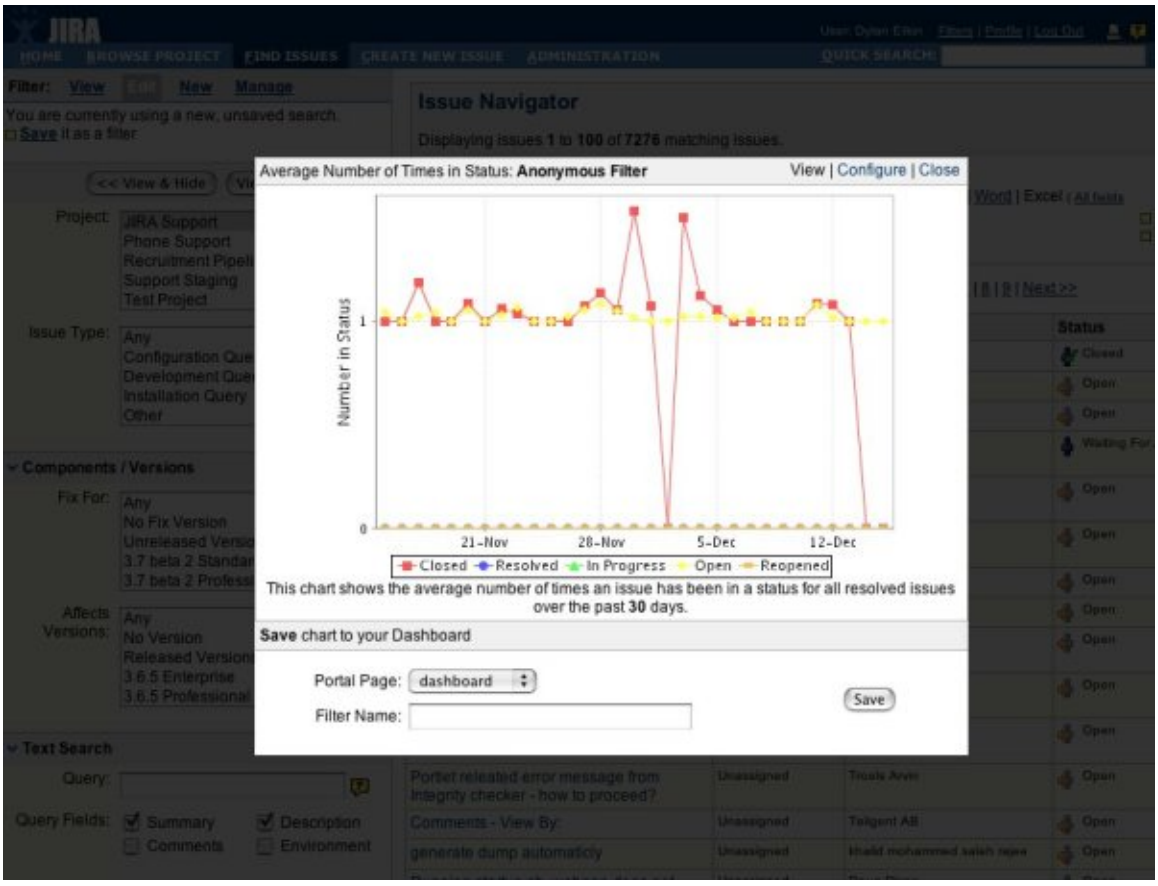
**A project administrator is someone who has the 'Administer Project' permission, but not neceessarily the global 'JIRA Administrator' permission.*

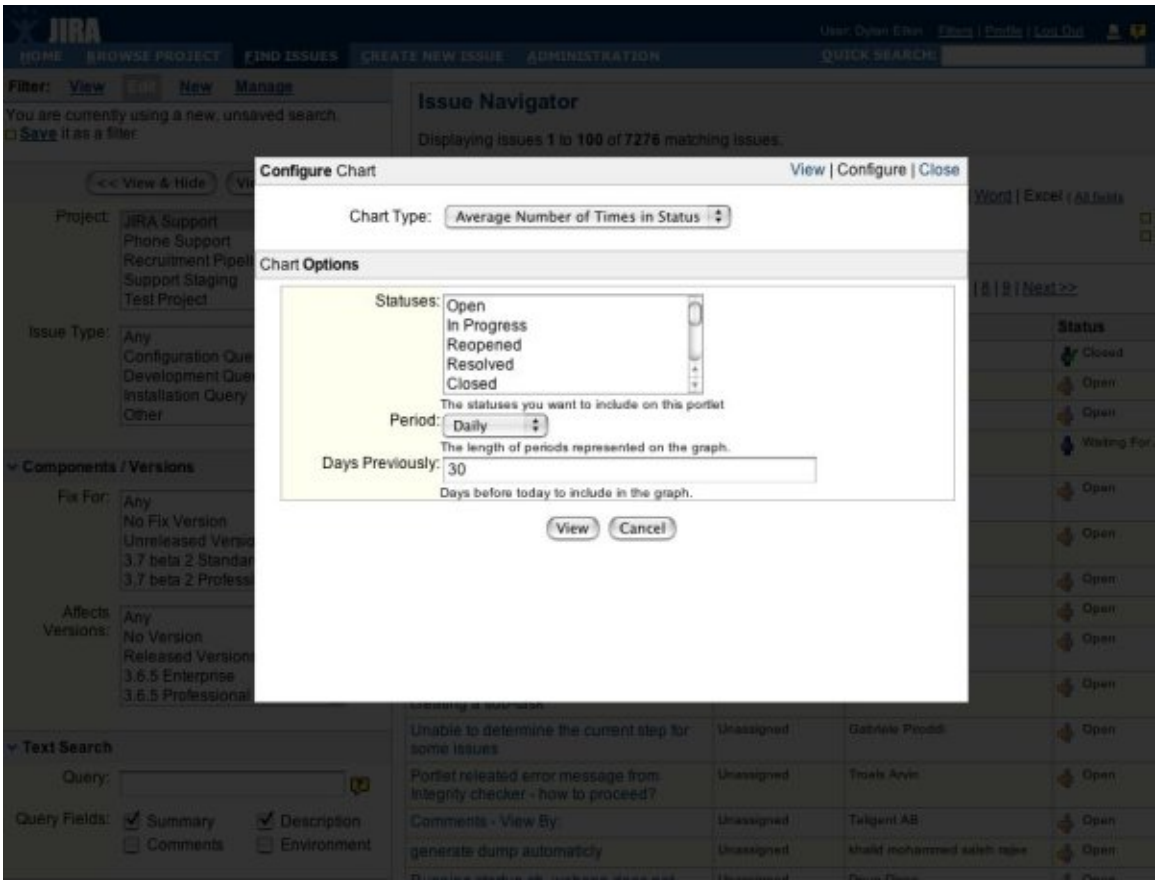
^Top

'Charting' View for Issue Navigator

The JIRA Charting Plugin now integrates with your Issue Navigator. When viewing search results you can click the 'Chart' option in the Issue Navigator views to popup an instant chart view.

From the chart view popup you have the option to configure any chart that is available via the charting plugin. Once happy with your chart and its configuration you can, in one step, create a named filter and save the chart to any page on your dashboard.





RSS Improvements

RSS Feeds got a completely new look:

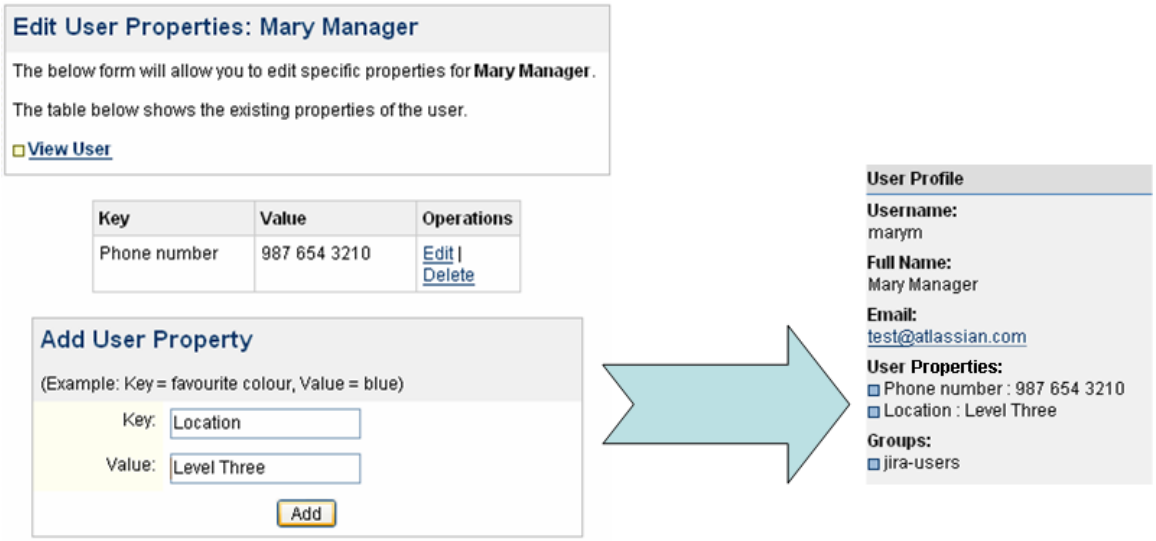
- **New RSS 2 compliant feed** — this feed will enable your RSS reader to recognise issue updates correctly, rather than report updates as newly created issues.
- **Much more readable RSS** — the new RSS feed presents issues in a much more readable format. If, with previous JIRA releases, you have been pointing your RSS reader at JIRA's XML feed, we strongly suggest switching to the new RSS feed.
- **Comments RSS feed** — an RSS feed that shows comments which have been recently added to issues matching your search criteria. If you have ever wanted to see new comments added to issues you are interested in, this feed is what you have been looking for:

^Top

User Properties

Ever wanted to record some additional information about a user? For instance, you might want to record their phone number, location, department, cost centre, etc.

In JIRA 3.7 you can easily add user properties of your choice. Once a user property is added, it is visible (to administrators) in the User Profile:

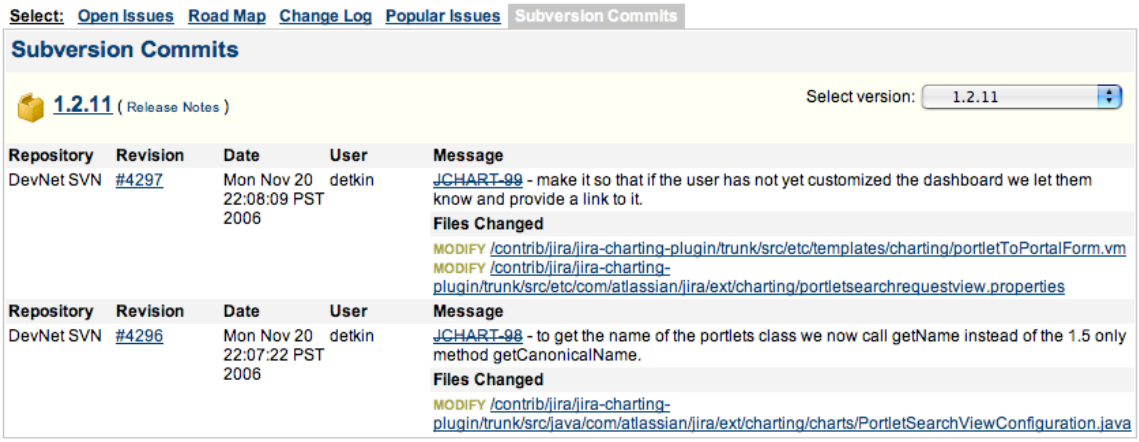


^Top

SVN Project Panel plugin

The JIRA Subversion Plugin now features a new project tab, which shows all commits made against a particular project or a project version, giving you a summary of recent developer activity on the project.

This new tab is displayed on the Browse Project page for each project:



You can download this plugin from its home page.

^Top

SVN Commit Acceptance plugin

In some environments, developers like the idea that there must be a one-to-one correlation between issues and check-ins. The new SVN Commit Acceptance plugin gives JIRA the ability to approve or deny any check-in made to CVS or SVN. We've started by allowing administrators to check three of the most common uses:

- Does this commit reference a valid issue key?
- Is that issue open?
- Does that issue belong to the committer?

Unable to render embedded object: File (commit-acceptance-plugin-configurator.png) not found.

This plugin has two parts: an RPC plugin and a client-side perl or python script. The perl script uses XML-RPC to call the plugin, passing in the committer and the commit message. The plugin checks any of the rules that the administrator has configured, and returns a yes or no, which

causes the perl script to allow or disallow the commit.

You can download this plugin from its home page.

^Top

Other Improvements

JIRA 3.7 features a large number of other improvements, the most notable include:

1. **Internet Explorer 7 compatibility** — JIRA 3.7 is compatible with the latest browsers, including Internet Explorer 7. You may want to check out the keyboard shortcuts.
2. **Improved search robustness** — In previous JIRA releases, each search caused additional memory usage and, during peak-hours, many concurrent searches could cause JIRA to run out of memory. In JIRA 3.7 the searching sub-system has been revamped so that many more searches can be executed concurrently in *constant memory*.
3. **AJAX-loading of dashboard portlets** — Portlets which can be slow to load are now loaded in two parts, making the Dashboard display much faster. The first (quick) part of the portlet is loaded with the Dashboard; the second is loaded as a separate request. For example, the List All Filters portlet shows the filter names first, and then fetches the count of matching issues for each filter.
4. **HSQLDB Upgrade** — The built-in database (which ships with JIRA Standalone) has been upgraded to a more stable version (1.8.0.5).
5. **Tomcat Upgrade** — The built-in application server (which ships with JIRA Standalone) has been upgraded to version 5.5.15.
6. **Performance improvements** — If you're using JIRA 3.6.3 or above, you have already seen some performance improvements. With JIRA 3.7, we've added:
    - Excel and XML exports, and RSS feeds, now use constant memory, which means that you are able to exports thousands of issues without affecting other JIRA users, or causing JIRA to run out of memory.
    - Faster page loads — all JavaScript and CSS files can now be cached by your browser, which makes every page in JIRA load faster.
7. **Time tracking improvements**:
    - It is now possible to import issue estimates via CSV.
    - It is possible to configure how the time tracking estimates and logged work information is formatted. You can choose 'days', 'hours' or 'pretty'. When choosing a specific time unit (e.g. days or hours) the information will be shown as fractions of the chosen unit (e.g. 6.5 hours). `Pretty` format preserves the way JIRA behaved in previous releases, i.e. breaks down the time into minutes, hours, days and weeks.
8. **Sub-task fields are now configurable** — You can now choose which sub-task fields are displayed on the sub-task 'Quick Create' form (shown on parent's 'View Issue' page). This is done via the jira-application.properties file.
9. **'Preset Filters' are now pluggable** — The 'Preset Filters' (which appear on the Project portlet and in the 'Browse Project' menu) are now pluggable. This means you can:
    - Add new 'Preset Filter' links. For example, add a link to a search that will find all issues in a custom status (e.g. Awaiting QA Approval).
    - Choose the display order of 'Preset Filter' links.
    - Place conditions on when a filter is visible (e.g. 'Assigned to me' filter is only shown to logged in users).
      For details, please see How to create a custom preset filter.
10. **Navigation menus are now pluggable** — JIRA 3.7 gives the ability for plugins to inject links to various sections of JIRA's UI. For example:
    - Main navigation bar (which appears accross the top of most JIRA pages).
    - Navigation pop-ups (e.g. History, Filters, Profile, etc).
    - Administration Menu (links appearing down the left-hand side in the administration section).
      This functionality greatly enhances how a plugin can interact with JIRA users. For example, plugins can now display a custom link in JIRA's Administration section and allow JIRA Administrators to provide configuration information via JIRA's UI, rather than having to tweak the `properties` file.
      The custom links are called 'Web UI Modules' in JIRA. For details on how to create these, please see this document.
11. **Pluggable Issue Navigator Views** — Issue Navigator views show results of a search in various formats. All built-in Issue Navigator views (e.g. Excel, Word, RSS) have been converted to plugins. This improvement creates the following advantages:
    - It is much easier to customise Issue Navigator views to your needs. For example, you can change the way that issue information is exported to MS Word.
    - By building a new plugin, you can show the results of a search in any format you wish.
12. **Pluggable Single Issue Views** — similar to Issue Navigator Views, the single-issue views (Word, XML) have been turned into plugins. This provides the ability to customise how the issue is exported to various formats, and to add custom export formats.
13. **Issue Operations turned into plugins** — all issue operation links ('Assign Issue', 'Attach File', 'Edit Issue') have been turned into plugins, which allows the JIRA Administrator to easily disable unwanted operations. For example, if you do not need the clone issue functionality, simply disable the 'Clone Issue' operation.

To see a list of all new features and improvements — ask JIRA!

^Top

Issue Operations plugin

To remove an issue operations link (e.g. **Assign this issue**), simply navigate to 'Plugins' (in the 'Adminstration' menu) and click 'Disable module':

JIRA 3.7 Upgrade Guide

Once you have upgraded to JIRA 3.7, downgrading to a previous version is not a straightforward task and is not recommended.

JIRA 3.7 Upgrade Notes

This page lists a few things to be aware of when upgrading from previous releases of JIRA to JIRA 3.7. To perform the actual upgrade, see the upgrade documentation.

Note: If you are upgrading from a pre-3.6.5 release, please also refer to the relevant JIRA 3.x Upgrade Guides.

Please note that JIRA 3.7 requires JDK 1.4 or above. Support for JDK 1.3 has been discontinued.

Please note that some new functionality will not be available if you are running JIRA on WebLogic or Orion. The List All Filters portlet will not be able to fetch the issue counts for each issue. The new 'Charting' View will also be unavailable. The support for WebLogic and Orion will be added in JIRA 3.7.1.

Database Schema Changes

Due to the upgrade of HSQLDB, and to improve compatibility with Firebird and Frontbase, various database tables and columns have been renamed. For more details on the changes please see the JIRA 3.7 Database Schema Changes document.

Therefore, the easiest way to upgrade to JIRA 3.7 is to follow the Upgrading JIRA safely instructions.

If in the past, instead of performing an XML backup and restore, you have been upgrading by "pointing" new version of JIRA at an old database, this is still possible, however the procedure is more complicated. You will need to use SQL scripts to perform database schema changes. For more information please see the SQL Scripts for 3.6.x to 3.7 schema upgrade document.

If you are using HSQLDB with JIRA, you must follow Upgrading JIRA safely instructions (i.e. perform a full XML backup and restore from XML), as simply copying the `.script` file will not work. The format of the `.script` file has changed between the HSQLDB versions, and therefore, copying the `.script` file will result in the following error on startup.

Request Context Changes

In order for plugins, customfields and portlets to function better outside of a web-context (e.g.: displaying a customfield in an e-mail), all direct references to the HttpServletRequest have been replaced by a VelocityRequestContext. If you have deployed your own plugins, customfields or portlets that use the HttpServletRequest directly (i.e.: any references to `${req}`) than they should be changed to use the new `${requestContext}` object. The `${requestContext}` is an implementation of the `VelocityRequestContext` interface.

Currently the `${requestContext}` supports the following properties:

- `${requestContext.baseUrl}` - Returns the same as HttpServletRequest.getContextPath() or the base URL configured in your JIRA instance if no HttpServletRequest is available
- `${requestContext.requestParameters}` - Returns an implementation of `RequestContextParameterHolder` or null if no HttpServletRequest is available
- `${requestContext.requestParameters.servletPath}` - Returns the same as HttpServletRequest.getServletPath()
- `${requestContext.requestParameters.requestURL}` - Returns the same as HttpServletRequest.getRequestURL()
- `${requestContext.requestParameters.queryString}` - Returns the same as HttpServletRequest.getQueryString()

Integrity Checks

In JIRA 3.7 Database Integrity Checks (available from the Administration section) have been re-written to run as multiple transactions, which increased the throughput of the system while the checks are running. In large JIRA 3.6 (and earlier) installations, integrity checks could cause database lock escalation and stop users from performing operations (e.g. viewing issues).

Please note, that due to the change, each integrity check became about 10% slower.

As integrity checks are quite database intensive operations, it is still recommended to run them during off-peak hours (i.e. while the system is not under heavy load).

Change of commentLevel to groupLevel in the Comment and TransitionWorkflow jelly tags

We have changed the AddComment and TransitionWorkflow jelly tag attribute that specifies the group visibility level from 'commentLevel' to be 'groupLevel'. If you have existing jelly tags that use this attribute it will need to change. This was done so that we could introduce the 'roleLevel' attribute which allows you to specify a project role based visibility. Only one of the two attributes can be specified at a time.

Change of level to grouplevel in the XML view of a Comment

1. We have changed the XML view of a comment, as seen in the XML view of an Issue to contain either a 'grouplevel' attribute or a 'rolelevel' attribute. This attribute defines the visibility level specified on the comment. In the past the 'grouplevel' attribute was simply 'level'. If you have any existing custom code that expects the 'level' attribute in the Comment XML it must change to expect 'grouplevel'.
2. In previous versions of JIRA the XML view of the <comment> tag level attribute was always shown, even if there was no value for the level, it was rendered as an empty attribute. We have changed it so that the attributes themselves (grouplevel and rolelevel) do not display if there is no value.

Change to the RemoteComment object used via SOAP/RPC plugin

The RemoteComment object and therefore the remote SOAP/RPC api has changes to almost all properties. The 'roleLevel' attribute was added and the following attributes have changed:

1. level -> groupLevel
2. datePerformed -> created
3. username -> author

ActionManager removed

The `ActionManager` interface has been removed and its functionality has been delegated to new interfaces. For details please refer to ActionManager Removed documentation

Removal of 'Backend Actions'

1. We have removed the 'Backend Action' `com.atlassian.jira.action.action.WorklogCreate` if you were using this class in a plugin or custom code you will now need to use the `com.atlassian.jira.issue.worklog.WorklogManager` this now has method calls to return worklogs for a given user+issue and also create worklog entries.
2. We have removed the 'Backend Action' `com.atlassian.jira.action.action.ActionCreate` if you were using this class to create comments you will need to modify your code to use one of the create methods on the com.atlassian.jira.bc.issue.comment.CommentService

Issue Events

We have modified the `com.atlassian.jira.event.issue.IssueEvent` class to no longer use GenericValues. The GenericValue representing the comment is replaced by `com.atlassian.jira.issue.comments.Comment` class and the GenericValue representing the worklog is replaced by `com.atlassian.jira.issue.worklog.Worklog` class. If you have a custom listener in a previous version of JIRA this will need to be updated to use the newer IssueEvent class and `com.atlassian.jira.event.issue.IssueEventDispatcher.dispatchEvent(...)` methods.

Renaming of XML export file

By popular request, the XML filename (that is, the default filename when you choose to save the XML view in the Issue Navigator) has been changed from `issuenavigator.jspa` to `SearchRequest.xml`. Should you have any external systems or programs that utilise the exported XML file, please be aware of the changed filename.

Confluence Users Only - Pre 2.2.10 Confluence Must Be Patched To Use JIRA Issues Macro
Unable to render {include} Couldn't find a page to include called: DOC:JIRA 3.7 Link Format Change

JIRA 3.7 Downgrade Notes

Once you have upgraded to JIRA 3.7, downgrading to a previous version is not a straightforward task and is not recommended. Please be aware that in JIRA 3.7 the database schema has changed.

If upgrade to JIRA 3.7 fails, the best way to proceed is to go back to the previous version of JIRA you were using, and to the latest **pre-upgrade** data that you have. The exact steps for doing this depend on how you have upgraded JIRA.

If you have created a new database for JIRA 3.7 by following the Upgrading JIRA safely instructions, you should be able to simply shutdown JIRA 3.7 and bring up the old version of JIRA your were using. The old version should be configured to use its old (unupgraded) database.

If you have upgraded JIRA by pointing JIRA 3.7 to an older database (and ran the SQL Scripts to upgrade the database schema), then you will need to:

1. Create a new database
2. Configure the old version of JIRA you were using to point at the new (empty) database
3. Restore the latest **pre-upgrade** backup that you have
4. Start the old JIRA installation

ActionManager Removed

From JIRA 3.7, the ActionManager has been refactored into several other interfaces, these include the CommentManager, WorklogManager, ChangeHistoryManager, RepositoryManager and IssueTabPanel. The following table is a mapping of the old ActionManager methods to the new refactored ones (including the new java interface the method resides in).

You will notice that the new methods in JIRA 3.7 take in the Issue object as opposed to the GenericValue.

ActionManager method (Pre JIRA 3.7)	Corresponding method (Post JIRA 3.7)	Corresponding Interface
List **getComments**(GenericValue issue, User user)	List **getCommentsForUser**(Issue issue, User user)	CommentManager
List **getWorklog**(GenericValue issue, User user)	List **getWorklogsForUser**(Issue issue, User user)	WorklogManager
List **getChangeHistory**(GenericValue issue, User remoteUser)	List **getChangeHistoriesForUser**(Issue issue, User remoteUser)	ChangeHistoryManager
List **getWorkflow**(GenericValue issue, User remoteUser)	removed	-
List **getActions**(GenericValue issue, User remoteUser)	List **getActions**(Issue issue, User remoteUser)	AllTabPanel
List **getCommits**(GenericValue issue, User remoteUser)	Map **getCommits**(Issue issue, User remoteUser)	RepositoryManager
Email **createEmail**(GenericValue issue, Message mimeMessage)	removed	-
List **getEmails**(GenericValue issue, User remoteUser)	removed	-
List **getPluginModuleActions**(String moduleKey, GenericValue issue, User remoteUser)	List **getActions**(Issue issue, User remoteUser)	IssueTabPanel

JIRA 3.7 Database Schema Changes

Due to the upgrade of HSQLDB, and to improve compatibility with Firebird and Frontbase, various database tables and columns have been renamed.

The table below summarises the changes to the database schema. Please note, that if you have developed any custom utilities which query or modify the JIRA database directly (i.e. without using the JIRA API), please check whether the utilities need to be updated.

Tables

The following database table has been renamed:

OLD TABLE NAME	NEW TABLE NAME
version	projectversion

Columns

The following database columns have been renamed. Their old and new names, as well as the database table they belong to are shown below:

TABLE NAME	OLD COLUMN NAME	NEW COLUMN NAME
notification	type	notif_type

notification	parameter	notif_parameter
mailserver	type	server_type
jiraeventtype	type	event_type
schemepermissions	type	perm_type
schemepermissions	parameter	perm_parameter
fieldlayout	type	layout_type
schemeissuesecurities	type	sec_type
schemeissuesecurities	parameter	sec_parameter
portletconfiguration	position	positionseq

Special note for MS SQL Server

Additionally to the schema changes described above, for MS SQL Server, all columns of type `TEXT` have been changed to `NTEXT` to ensure that international characters can be safely stored.

Possible upgrade problems + solutions

HSQL DB Upgrade

JIRA 3.7 has been upgraded to use HSQL DB version 1.8. HSQL DB is the in-memory database that ships with the Standalone distribution of JIRA.

If you have upgraded to JIRA 3.7 and have tried to copy across the HSQL DB script files from your 3.6.x or earlier instance you will see an error like the following:

```
2006-09-20 16:33:49,858 [core.entity.jdbc.DatabaseUtil] Unable to esablish a connection with the
database... Error was:org.apache.tomcat.dbcp.dbcp.SQLNestedException: Cannot create
PoolableConnectionFactory (error in script file line: 104 Unexpected token: POSITION in statement
[CREATE TABLE PORTLETCONFIGURATION(ID BIGINT NOT NULL PRIMARY KEY,PORTALPAGE BIGINT,PORTLET_ID
VARCHAR,COLUMN_NUMBER INTEGER,POSITION])
 2006-09-20 16:33:49,858 [core.entity.jdbc.DatabaseUtil] Could not get table name information from the
database, aborting.
 2006-09-20 16:33:51,493 WARN [jira.config.properties.ApplicationPropertiesImpl] Exception getting
property 'jira.extension.container.provider' from database. Using default
 2006-09-20 16:33:51,729 ERROR [ContainerBase.[Catalina].[localhost].[/]] Exception sending context
initialized event to listener instance of class com.atlassian.jira.upgrade.ConsistencyLauncher
 com.opensymphony.module.propertyset.PropertyImplementationException: Unable to esablish a connection
with the database. (Cannot create PoolableConnectionFactory (error in script file line: 104 Unexpected
token: POSITION in statement [CREATE TABLE PORTLETCONFIGURATION(ID BIGINT NOT NULL PRIMARY
KEY,PORTALPAGE BIGINT,PORTLET_ID VARCHAR,COLUMN_NUMBER INTEGER,POSITION]))
         at
com.opensymphony.module.propertyset.ofbiz.OFBizPropertySet.getKeys(OFBizPropertySet.java:100)
         at
com.opensymphony.module.propertyset.AbstractPropertySet.getKeys(AbstractPropertySet.java:292)
         at
com.opensymphony.module.propertyset.PropertySetCloner.cloneProperties(PropertySetCloner.java:114)
         at com.opensymphony.module.propertyset.PropertySetManager.clone(PropertySetManager.java:61)
         at
com.atlassian.jira.propertyset.JiraCachingPropertySet.init(JiraCachingPropertySet.java:411)
         at
com.opensymphony.module.propertyset.PropertySetManager.getInstance(PropertySetManager.java:45)
         at
com.opensymphony.module.propertyset.PropertySetManager.getInstance(PropertySetManager.java:22)
         at
com.atlassian.jira.config.properties.PropertiesManager.loadPropertySet(PropertiesManager.java:79)
         at com.atlassian.jira.config.properties.PropertiesManager.<init>(PropertiesManager.java:25)
         at
com.atlassian.jira.config.properties.PropertiesManager.getInstance(PropertiesManager.java:35)
         at
com.atlassian.jira.config.properties.ApplicationPropertiesImpl.getPropertiesManager(ApplicationProperties
at
com.atlassian.jira.config.properties.ApplicationPropertiesImpl.getStringsWithPrefix(ApplicationProperties
```

```
at com.atlassian.jira.plugin.JiraPluginStateStore.loadPluginState(JiraPluginStateStore.java:41)
         at com.atlassian.plugin.DefaultPluginManager.<init>(DefaultPluginManager.java:29)
         at com.atlassian.jira.plugin.JiraPluginManager.<init>(JiraPluginManager.java:28)
         at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
         at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
         at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
         at java.lang.reflect.Constructor.newInstance(Constructor.java:494)
         at
org.picocontainer.defaults.InstantiatingComponentAdapter.newInstance(InstantiatingComponentAdapter.java:1
at
org.picocontainer.defaults.ConstructorInjectionComponentAdapter.instantiateComponent(ConstructorInjectior
at
org.picocontainer.defaults.InstantiatingComponentAdapter.getComponentInstance(InstantiatingComponentAdapt
at
org.picocontainer.defaults.DecoratingComponentAdapter.getComponentInstance(DecoratingComponentAdapter.jav
at
org.picocontainer.defaults.SynchronizedComponentAdapter.getComponentInstance(SynchronizedComponentAdapter
at
org.picocontainer.defaults.DecoratingComponentAdapter.getComponentInstance(DecoratingComponentAdapter.jav
at
com.atlassian.jira.config.component.ProfilingComponentAdapter.getComponentInstance(ProfilingComponentAdap
at org.picocontainer.defaults.DefaultPicoContainer.getComponentInstance(DefaultPicoContainer.java:298)
         at com.atlassian.jira.ComponentManager.quickStart(ComponentManager.java:267)
         at com.atlassian.jira.ComponentManager.start(ComponentManager.java:254)
         at
com.atlassian.jira.upgrade.ConsistencyLauncher.contextInitialized(ConsistencyLauncher.java:25)
         at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:3669)
         at org.apache.catalina.core.StandardContext.start(StandardContext.java:4104)
         at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
         at org.apache.catalina.core.StandardHost.start(StandardHost.java:718)
         at org.apache.catalina.core.ContainerBase.start(ContainerBase.java:1012)
         at org.apache.catalina.core.StandardEngine.start(StandardEngine.java:442)
         at org.apache.catalina.core.StandardService.start(StandardService.java:450)
         at org.apache.catalina.core.StandardServer.start(StandardServer.java:683)
         at org.apache.catalina.startup.Catalina.start(Catalina.java:537)
         at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
         at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
         at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
         at java.lang.reflect.Method.invoke(Method.java:585)
```

```
        at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:271)
        at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:409)
```

If you see this error, JIRA will not be able to startup.

Solution

To resolve this problem you will need to 'safely' upgrade JIRA. The detailed instructions for doing this can be found in JIRA's online documentation.

A quick guide on doing this follows:

1. Startup your old version of JIRA, pointing it at your current HSQL DB scripts files
2. [Perform an XML backup of JIRA]
3. Shutdown you old 3.6.x JIRA instance.
4. Startup your new 3.7 instance of JIRA using a new script location

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
 username="sa"
 password=""
 driverClassName="org.hsqldb.jdbcDriver"
 url="jdbc:hsqldb:${catalina.home}/database/jiradb37"
 minEvictableIdleTimeMillis="4000"
 timeBetweenEvictionRunsMillis="5000"
/>
```

5. Perform an XML import

SQL Scripts for 3.6.x to 3.7 schema upgrade

Audience	People who are upgrading from pre-JIRA 3.6.x to post-3.7, and cannot use the recommended 'XML backup/restore' upgrade method because it would take too long.

If you are upgrading JIRA by the recommended method, ignore this page.

Background

In general, there are two ways to upgrade JIRA's database, both of which are described in the Upgrading JIRA document:

- **XML backup/restore (recommended)** — doing a full XML export and import into the new database.
- **Connect JIRA to a copy of your old database** — connecting the new JIRA to the old database, and letting it automatically upgrade the database tables.

In 3.7.x, **the 'Connect JIRA to a copy of your old database' method will not work**. If you point JIRA 3.7.x to an older database, JIRA will print a warning and refuse to do anything. This is because a large number of database schema changes were made between 3.6 and 3.7 (see JIRA 3.7 Database Schema Changes) and these changes are too great for JIRA's database engine to upgrade automatically.

So, the **vast majority of users** should follow the XML backup/restore method described in the Upgrading JIRA document.

However, there are a handful of users with large installations, for which a full export/import is impractical as it takes a relatively long time. These users may need to use the 'Connect JIRA to a copy of your old database' method — **This page is intended for these users.** We have provided SQL scripts for each database, which will make the required changes to a 3.6.x database so it can be upgraded without a full export/import.

These scripts will only work on **JIRA 3.6.x** databases (they refer to a table only added in 3.6). If you are upgrading from an earlier release, please:

1. Download JIRA 3.6.5 Standalone
2. Back-up your database, and create a copy to be upgraded to 3.7.
3. Configure it to point to your 3.7 copy of the database
4. (optionally) Edit `atlassian-jira/WEB-INF/classes/jira-application.properties` and set `jira.autoexport=false` to speed up the process.
5. Start JIRA Standalone. By watching the logs (`atlassian-jira.log` or `logs/catalina.out`, you will see JIRA automatically upgrading tables to the 3.6.x format.
6. Proceed with the instructions below.

If you are using HSQLDB with JIRA, you **must** follow the 'XML backup/restore' instructions in the Upgrading JIRA guide, as simply copying the `.script` file **will not work**. The format of the `.script` file has changed between the HSQLDB versions, and therefore, copying the `.script` file will result in the following error on startup.

DB2 upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your DB2 database: **db2 backup database sample to /home/db2/backups**
3. Download the following script db2_3.7_migration.ddl and modify the connect statement within the file
4. Execute the script using the following command: **db2 +c -t -v -f db2_3.7_migration.ddl**
   If you see errors like:

```
insert into SCHEMEISSUESECURITIES (select ID, SCHEME, SECURITY, TYPE, PARAMETER from
TMP_SCHEMEISSUESECURITIES)
SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a
query is an empty table.  SQLSTATE=02000
```

   This is OK, it simply means that the inner SQL query did not return any data to be inserted into the new temporary table. This can occur if you are not using certain features in JIRA.

5. Point your new instalation of JIRA 3.7 at your DB2 database and watch for any errors during the statup sequence.
6. If you see any other errors please contact support for further assistance.

PostgreSQL 8+ upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your PostgreSQL database **pg_dump -d *database name* > *backup filename*.sql**
   - for example **pg_dump -d jiradb > jiradb_365_14112006.sql**
3. Download the following script postgres_3.7_migration.sql
4. Execute the script using the following command: **psql -Ujirauser -v schema_name=public -d jiradb -f postgres_3.7_migration.sql**
   a. Where -U is the username, -v is the name of your schema -d is the database -f is the location of the script file
5. Point your new instalation of JIRA 3.7 at your database and watch for any errors during the startup sequence.
6. If you see any errors please contact support for further assistance.

Oracle 10g upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your Oracle database. There are multiple strategies here, so we will leave this up to your DBA.
3. Download the following script oracle_3.7_migration.sql
4. Connect to SQL*Plus and execute the following script:

```
SQL*Plus: Release 10.2.0.1.0 - Production on Thu Oct 19 12:56:11 2006
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> @/home/oracle/oracle_3.7_migration.sql
```

5. If you see any errors please contact support for further assistance.

6. Point your new instalation of JIRA 3.7 at your DB2 database and watch for any errors during the statup sequence.

Microsoft SQL Server upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your SQL Server database: **osql -U *username* -P *password* -Q "BACKUP DATABASE *db_name* TO DISK = *backup_path_and_filename*"**
    - for example **osql -U sa -P secret -Q "BACKUP DATABASE jiradb TO DISK = 'C:\MyBackup.dat'"**

1. Download the following script sqlserver_3.7_migration.sql
2. Execute the script: **osql -U *username* -P *password* -d *db_name* -i mssql_3.7_migration.sql**
    - for example **osql -U sa -P secret -d jiradb -i sqlserver_3.7_migration.sql**
3. If everything goes well the following should be displyed

```
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'NOTIF_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'NOTIF_PARAMETER'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'SEVER_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'EVENT_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'PERM_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'PERM_PARAMETER'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'LAYOUT_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'SEC_TYPE'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'SEC_PARAMETER'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The COLUMN was renamed to 'POSITIONSEQ'.
Caution: Changing any part of an object name could break scripts and stored procedures.
The object was renamed to 'projectversion'.
```

4. Point your new instalation of JIRA 3.7 at your SQL Server database and watch for any errors during the statup sequence.

Sybase upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your SQL Server database
    - for example using **isql** tool

```
1> dump database db_name to "backup_path_and_filename"
2> go
```

3. Download the following script sybase_3.7_migration.sql
4. Execute the script: **osql -U *username* -P *password* -D *db_name* -i sybase_3.7_migration.sql**
    - for example **osql -U sa -P -D jiradb -i sybase_3.7_migration.sql**
5. If everything goes well the following should be displyed

```
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Column name has been changed.
(return status = 0)
Object name has been changed.
(return status = 0)
```

6. Point your new instalation of JIRA 3.7 at your Sybase database and watch for any errors during the statup sequence.

MySQL upgrade notes

1. Shutdown your JIRA instance
2. Perform a backup of your MySQL database: **mysqldump --opt *db_name* > *db_name*.sql**
    - for example **mysqldump --opt jiradb > jiradb_before37.sql**
3. Download the following script mysql_3.7_migration.sql
4. Execute the script: **mysql --user=*username* --password=*password* *db_name* < mysql_3.7_migration.sql**
    - for example **mysql --user=root --password=password jiradb < mysql_3.7_migration.sql**
5. Point your new installation of JIRA 3.7 at your MySQL database and watch for any errors during the statup sequence.

Attachments

Name	Size	Creator (Last Modifier)	Creation Date	Last Mod Date	Comment
sybase_3.7_migration.sql	0.5 kB	Dylan Etkin	Sep 20, 2006	Sep 20, 2006	
mysql_3.7_migration.sql	0.7 kB	Dylan Etkin	Sep 20, 2006	Sep 20, 2006	
sqlserver_3.7_migration.sql	12 kB	Jed Wesley-Smith	Mar 22, 2007	Mar 22, 2007	
db2_3.7_migration.ddl	4 kB	Justin Koke	Oct 18, 2006	Oct 18, 2006	
oracle_3.7_migration.sql	0.7 kB	Justin Koke	Oct 18, 2006	Oct 18, 2006	
postgres_3.7_migration.sql	0.9 kB	Justin Koke	Oct 18, 2006	Oct 18, 2006	

JIRA 3.7.4 Release Notes

JIRA 3.7.4 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.7.4 in Standard, Professional and Enterprise editions. This point release includes 31 bug fixes and improvements, notably:

- **JIRA startup** — speed has been improved for Oracle - JRA-12049
- **Project roles** — a few minor bug fixes.
- **Getting ready for Crowd** — updates to JIRA infrastructure in preparation for integrating with Crowd in JIRA 3.8.

JIRA 3.7.4 can be downloaded here, and of course is free to all customers who purchased their JIRA licence or maintenance within the last 12 months.

If upgrading, please refer to the JIRA 3.7.4 Upgrade Guide.

Not using 3.7? Learn about all the new features you're missing out on!

> **Weblogic Users**
>
> Please note that there is a known Weblogic and Firefox issue that will affect JIRA 3.7.x when using Weblogic and Firefox. See the issue for more detail.

JIRA 3.7.4 includes the following bug fixes.

JIRA Issues (36 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-12128	Wrong selection when using "Group Custom Field Value" while custimzing a Notification Scheme		Resolved	Fixed
	JRA-12041	In Professional, permission scheme page implies that individual users can be assigned permissions (they can't)		Resolved	Fixed
	JRA-12195	Moving issue increments key of subsequently created issue		Resolved	Fixed
	JRA-12014	NPE in 500 page		Resolved	Fixed
	JRA-11514	org.apache.velocity.runtime.exception.ReferenceException: reference : template = templates/jira/issue/table/issuetable.vm [line 3,column 63] : $column.getHtml($celldisplayparams, $issue) is not a valid reference.		Resolved	Fixed
	JRA-11058	Exception should not be logged when invalid duration is entered		Resolved	Fixed
	JRA-12061	StackOverflowError when starting JIRA on Weblogic 8.1 and Orion after an incorrect upgrade		Resolved	Fixed
	JRA-12160	Need to modify trackback documentation		Resolved	Fixed
	JRA-12220	Move the Create New Translation JIRA doc to confluence		Resolved	Fixed
	JRA-12100	Quick Search - "updated:-1y,-1m"		Closed	Fixed
	JRA-11994	When adding a createOrComment mail handler with no params you get a NPE when the handler tries to handle the mail		Resolved	Fixed
	JRA-12871	CVS Log Handler no longer generates Issue Commented notifications		Resolved	Fixed
	JRA-12124	CSV Import Wizard does not work with custom fields anymore		Resolved	Fixed
	JRA-12153	Russian: FixForVersion translated incorrectly (again)		Resolved	Fixed
	JRA-12130	"Edit issue" resets the assigned component		Resolved	Fixed
	JRA-12036	Clicking Printable link in Issue Navigator then going back breaks the Filter		Resolved	Fixed
	JRA-12015	Issue Navigator should better handle mistyped ids		Resolved	Fixed
	JRA-12049	JIRA startup for 3.7.x is slow against some databases		Resolved	Fixed

	JRA-12113	Signup link visible when external user and/or password management is ON.		Resolved	Fixed
	JRA-12060	Guard project role group membership edits by the global permission Browse Users		Resolved	Fixed
	JRA-12107	CLONE -Error with possible Watching Portlet Configuration		Resolved	Fixed
	JRA-12086	Entering large values for number of issues to display for a portlet causes exception		Resolved	Fixed
	JRA-11956	Exception is thrown when editing Project Role Memberships with Oracle		Resolved	Fixed
	JRA-12174	Default project roles not populated via SOAP upon project creation		Resolved	Fixed
	JRA-12067	When you choose to create a Word document from the issues, JIRA creates one Word table per issue. The tables created in the Word document no longer have their lines drawn out, ie the lines are now invisible (previously, the lines were black, thick lines)		Resolved	Fixed
	JRA-12155	Deleting project role "breaks" workflow condition page		Resolved	Fixed
	JRA-12082	Assign Users to Project Role handles case sensitivity in username wrong - association that cannot be deleted		Resolved	Fixed
	JRA-12081	On 500page.jsp, only show services info to admins		Resolved	Fixed
	JRA-12018	red or green color background around sub task number is missing/incompatible in Internet Explorer 6.0 and 7.0		Resolved	Fixed
	JRA-11860	View User throws a NullPointerException when using Crowd (0.3.3 and below)		Resolved	Fixed
	JRA-12066	Comment dates don't display properly in 3.7.3		Resolved	Fixed
	JRA-12133	NPE in ProjectActionSupport when moving issue before visiting its View Issue page - does not affect normal usage		Resolved	Fixed
	JRA-12148	Component and Versions displayed on the manage links or attachment screens are wrong		Resolved	Fixed
	JRA-12184	Release Notes don't escape HTML special characters in TextArea		Resolved	Fixed
	JRA-11825	Edit Transition page throws Exception if name contains invalid characters but still updates the transitions name		Resolved	Fixed
	JRA-12052	Update the caching library needed for crowd		Resolved	Fixed

JIRA 3.7.4 Upgrade Guide

Upgrading from JIRA 3.7.3 to 3.7.4

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.7.2 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.7.3 Release Notes

JIRA 3.7.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Recommended Upgrade

JIRA 3.7.3 contains a security update and is highly recommended, especially for instances of JIRA that are accessible via the Internet.

Atlassian Software Systems is proud to announce the release of JIRA 3.7.3 in Standard, Professional and Enterprise editions. This point release includes 15 bug fixes and improvements, notably:

- **Excel formatting** has been rectified (JRA-11928, JRA-8899).
- **French translations** have been rolled back (temporarily) to pre-3.7.1 (JRA-11973, JRA-11873).
- **Security Update** - a problem was fixed where, at worst, a malicious user could indirectly email (and obtain a list of) other users. No other project information could be leaked, apart from the users' emails. This bug affects all previous versions.

JIRA 3.7.3 can be downloaded here, and of course is free to all customers who purchased their JIRA licence or maintenance within the last 12 months.

If upgrading, please refer to the JIRA 3.7.3 Upgrade Guide.

Not using 3.7? Learn about all the new features you're missing out on!

Weblogic Users

Please note that there is a known Weblogic and Firefox issue that will affect JIRA 3.7.x when using Weblogic and Firefox. See the issue for more detail.

JIRA 3.7.3 includes the following 15 bug fixes and improvements.

JIRA Issues (15 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-11204	CreateOrCommentHandler: Attachment with invalid filename causes loop		Resolved	Fixed
	JRA-11175	File names in attachments are truncated just before the ";" character		Resolved	Fixed
	JRA-11950	When comments that are restricted to Roles are deleted, their body appears in change history.		Resolved	Fixed
	JRA-11840	Fix implementation of CustomFieldImpl.compare(Issue, Issue) method		Resolved	Fixed
	JRA-11963	Outdated maven repository url		Resolved	Fixed
	JRA-11928	Excel export is not formatted correct		Resolved	Fixed
	JRA-11873	French I18n : Issue is now a "Problème" !		Resolved	Fixed
	JRA-11973	Can't edit general configuration with French locale		Resolved	Fixed
	JRA-11974	Button titles are hard-coded in English on manageversions.jsp		Resolved	Fixed
	JRA-8899	Export to Excel doesn't export all records unless you are on Page 1		Resolved	Fixed

	JRA-11546	CreateIssue do not set Reporter to specified value		Resolved	Fixed
	JRA-10804	SOAP's UserService::createUser() should add user to all groups with USE permission		Resolved	Fixed
	JRA-11999	Add XFire (Crowd) dependency libraries and adjust configuration file to have stubbed out Crowd configurations.		Resolved	Fixed
	JRA-11992	Change the description of the global Browse Users permission		Resolved	Fixed
	JRA-11906	Screenshot paste in Firefox on mac os x fails		Resolved	Fixed

JIRA 3.7.3 Upgrade Guide

Upgrading from JIRA 3.7.2 to 3.7.3

Please follow the JIRA general upgrade instructions.

Upgrading from JIRA 3.7.1 and earlier

In addition to the above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

JIRA 3.7.2 Release Notes

JIRA 3.7.2 Release Notes

Recommended Upgrade

This upgrade is strongly recommended for all users of JIRA 3.7.x as it contains a fix for a critical search indexing issue. Any users of JIRA 3.7 or 3.7.1 should upgrade immediately.

Atlassian Software Systems is proud to announce the release of JIRA 3.7.2 in Standard, Professional and Enterprise editions. This point release includes 32 bug fixes and improvements, including:

- Most notably, 3.7.2 includes a fix for a critical search indexing issue, which makes it a **highly recommended upgrade** for JIRA 3.7 and 3.7.1 users.
- A few small bugs in the CSV importer have been fixed (JRA-11847, JRA-11842).
- Deleting role members in Sybase now works (JRA-11890).

JIRA 3.7.2 can be downloaded here, and of course is free to all customers who purchased their JIRA license or maintenance within the last 12 months.

Not using 3.7? Learn about all the great new features you're missing out on!

If upgrading, please read refer to the JIRA 3.7.2 Upgrade Guide.

Weblogic Users

Please note that there is a known Weblogic and Firefox issue that will affect JIRA 3.7.x when using Weblogic and Firefox. See the issue for more detail.

JIRA 3.7.2 includes the following 32 bug fixes and improvements.

JIRA Issues (29 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**

	JRA-11702	Integrity Checker itself dies unexpectedly due to a bug in the Bugzilla importer		Resolved	Fixed
	JRA-11853	Group to Role mapper breaks - URL too long?		Resolved	Fixed
	JRA-11911	Able to add attachements altough session is time out		Resolved	Fixed
	JRA-11381	Unsynchronized HashMap in IssueLevelSecurities class		Resolved	Fixed
	JRA-11899	IssueActionComparator causes a ClassCastException		Resolved	Fixed
	JRA-11836	Create integrity checker for links to non-existing issues		Resolved	Fixed
	JRA-11452	-Dmail.debug property does not work for POP/IMAP		Resolved	Fixed
	JRA-11924	MailFetcherService does not delete mail message if it is unable to forward the mail because of an exception		Resolved	Fixed
	JRA-11861	Doc out of order exception when merging		Resolved	Fixed
	JRA-6199	Unable to import Comment Author and Comment Date		Resolved	Fixed
	JRA-11835	Improve CSV importer wizard error messages		Resolved	Fixed
	JRA-11847	CSV import wizard implictly ignores field mappings in the configuration file if they are not from the CSV header		Resolved	Fixed
	JRA-11842	CSV Import wizard does not check if a summary field has been mapped		Resolved	Fixed
	JRA-11865	Default Jira download should support Tomcat 55 rather than Tomcat 50		Resolved	Fixed
	JRA-9084	Jira does not work in Arabic		Resolved	Fixed
	JRA-11857	Danish Property Files for 3.7		Resolved	Fixed
	JRA-10287	Hide/Show text for collapsible fields are not translated		Closed	Fixed
	JRA-11661	Not Logged In User - The Pie Chart Report does not pass the first step		Resolved	Fixed
	JRA-11832	tooltip of the project portlets priority bar is wrong for issues with no priority		Resolved	Fixed
	JRA-11390	Single letter Project keys fail		Resolved	Fixed
	JRA-11878	SOAP - JiraSoapService.getComments(token, issueKey) NPE		Resolved	Fixed
	JRA-11445	AddComment in SOAP does not validate if the provided group exists.		Resolved	Fixed
	JRA-11676	Change log not updated after attaching files via remote API		Resolved	Fixed
	JRA-11890	Deleting user or group from Role breaks on Sybase (and other column-case-sensitive databases)		Resolved	Fixed

	JRA-11831	Upgrade Standalone to latest Tomcat 5.5.x		Resolved	Fixed
	JRA-11071	Should not allow a user to delete the last group with the global permission administrator		Resolved	Fixed
	JRA-11730	User profile shows groups, not roles		Resolved	Fixed
	JRA-11888	Group dropdown not in alphabetical order when commenting on issue		Resolved	Fixed
	JRA-11727	Select component assignee page should display a nice message when no components exist		Resolved	Fixed

JIRA 3.7.2 Upgrade Guide

JIRA 3.7.2 Upgrade Guide

This page contains specific information you need to know when upgrading from JIRA 3.7.1 to JIRA 3.7.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- 3.7.2 will automatically perform a full reindex when upgrading. For more details please see JRA-11861

JIRA 3.7.1 Release Notes

JIRA 3.7.1 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.7.1 in Standard, Professional and Enterprise editions. This point release includes 34 bug fixes and improvements, including many user interface and consistency fixes. It can be downloaded here.

Not using 3.7? Learn about all the **new features you're missing out on**!

If upgrading from an earlier version please read through the JIRA 3.7.1 Upgrade guide.

> **Weblogic Users**
>
> Please note that there is a known Weblogic and Firefox issue that will affect JIRA 3.7.x when using Weblogic and Firefox. See the issue for more detail.

JIRA 3.7.1 includes the following 34 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '12531' does not exist for the field 'fixVersion'.

JIRA 3.7.1 Upgrade guide

JIRA 3.7.1 Upgrade Guide

This page contains specific information you need to know when upgrading from JIRA 3.7 to JIRA 3.7.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.7 to JIRA 3.7.1.

JIRA 3.6 Release Notes

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

Atlassian is proud to announce the latest release of the award winning issue tracking and project mamagement software - **JIRA 3.6**. This release offers a range of new features and improvements throughout JIRA - from the introduction of custom events to wiki-style link aliasing - along with a number of bug fixes.

As always, this release can be downloaded from the JIRA Download Center - and dont forget to check out the upgrade guide if upgrading from a previous version!

Contents

- New Features
- Improvements
- Bug Fixes
- Upgrading

New Features

To see a full list of new features and improvements added - ask JIRA!

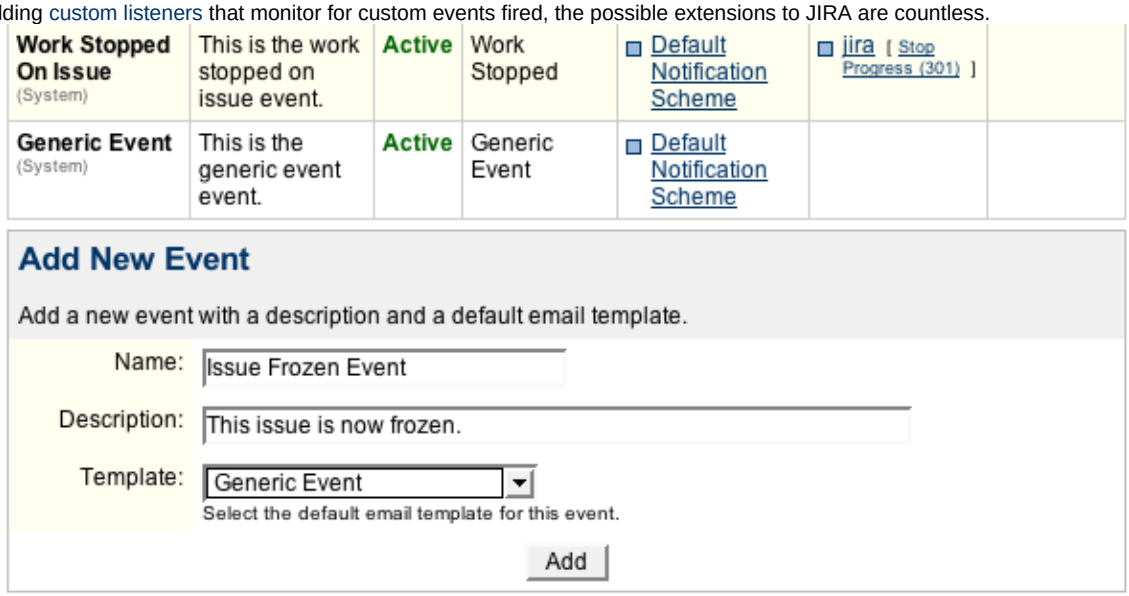
- Custom Events
- Group Picker Custom Field
- Per-Issue Group Notifications & Permissions
- Wiki-style Linking
- "I'm Feeling Lucky" Quick Search
- Collapsible Fields
- Nestable Conditions
- Charting Plugin Updates

Custom Events

JIRA uses an event-listener mechanism to alert the system that something has happened and allow it to perform an action based on that event. In both Professional and Enterprise editions, this release presents the ability to add custom events to the system - providing a pivotal extension point for notification and workflow schemes.

For instance, a custom event can be added to the system and associated with a workflow post-function - fired on completion of the event. A notification scheme can then be configured to email particular recipients once this event has been fired. With custom events, the notification and workflow schemes can be configured extensively to respond to specific custom events added to the system.

Further, by adding custom listeners that monitor for custom events fired, the possible extensions to JIRA are countless.

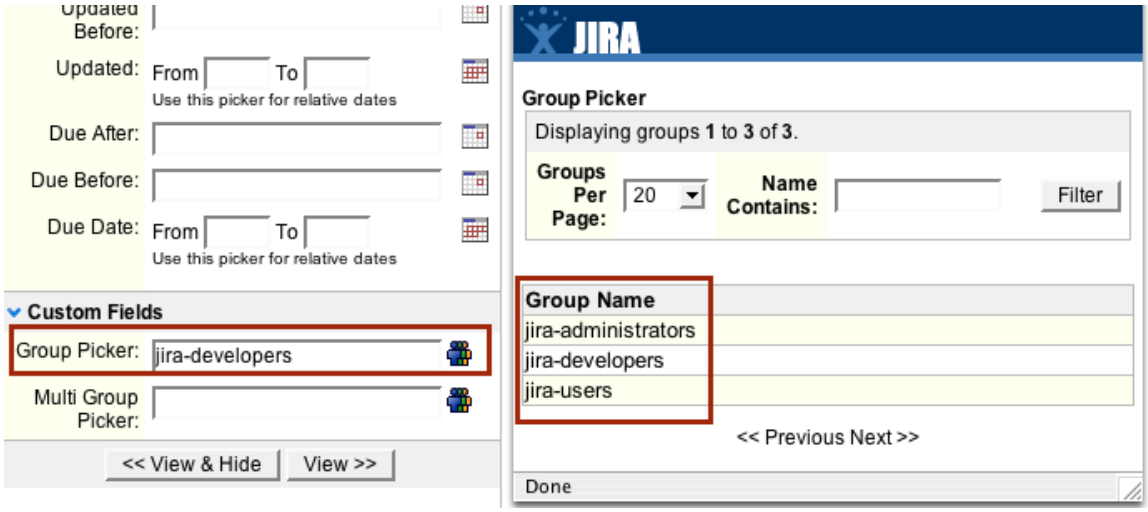


Group Picker Custom Field

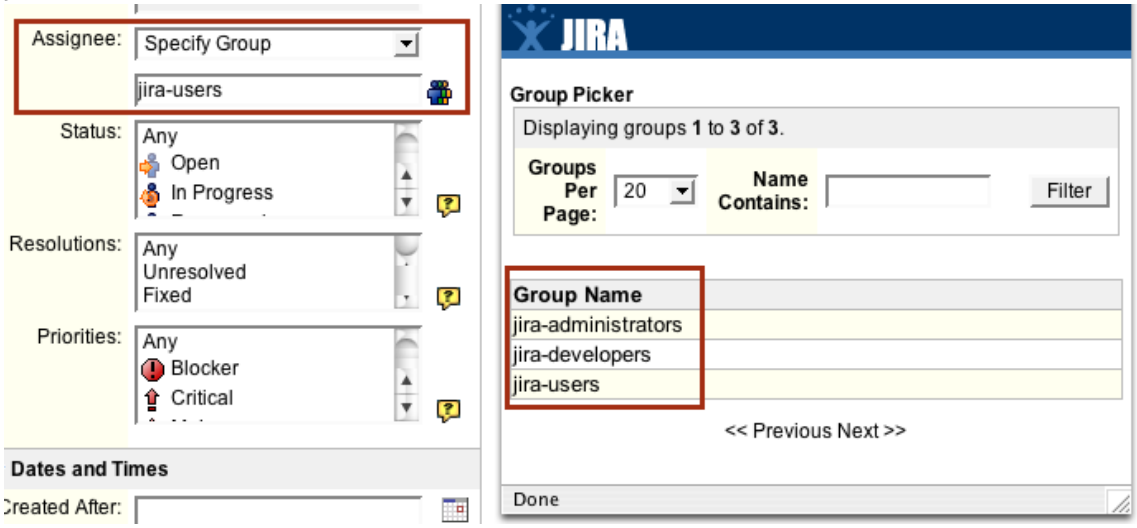
JIRA custom fields have become invaluable tools in extending the data associated with an issue. This release introduces two new custom fields into the fold:

- Group Picker
- Multi Group Picker

The **Group Picker** allows a JIRA-defined group to be associated with an issue while the **Multi Group Picker** allows the association of multiple JIRA-defined groups with an issue. It is possible to use these fields in an issue search as filter criteria in the Issue Navigator.



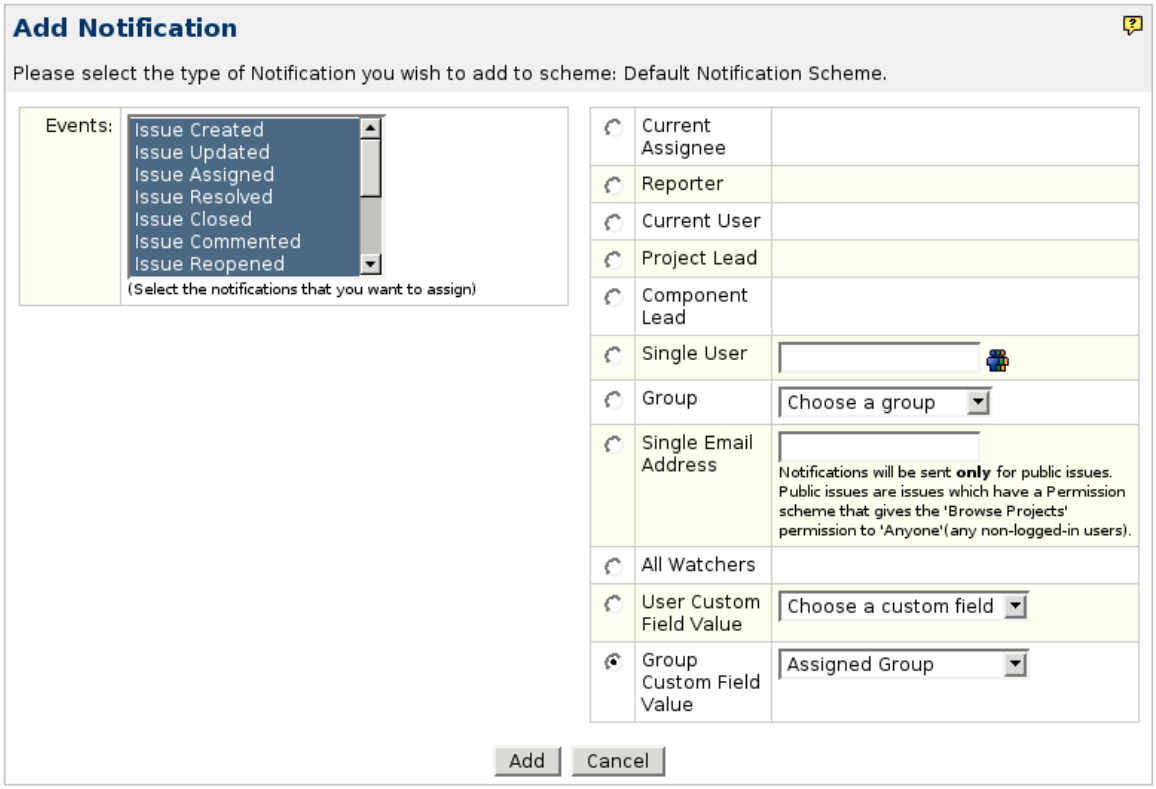
The assignee selector field in the Issue Navigator has also been updated with the Group Picker link - allowing the group to be selected from the available list.



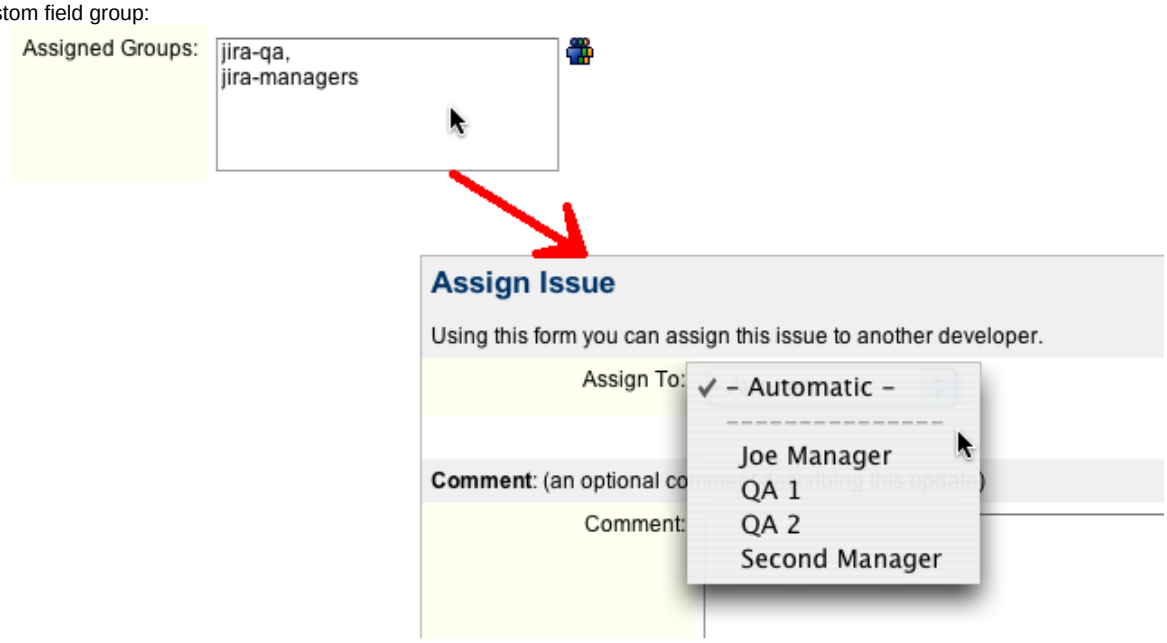
Per-issue Group Notifications and Permissions

Building on the group picker custom field, one can now send notifications to members of a group determined by a custom field.

For example, add an **Assigned Group** custom field, and edit the notification scheme to send all notifications to the group's members:



Now the "Assigned Group" on each issue will be notified of changes.
As with notifications, one can now grant a permission to members of a group(s) selected by a custom field. Continuing our **Assigned Group** example, by granting the **Assigned Group** the **Assignable** permission, the possible assignees of an issue will be the members the **Assigned Group** custom field group:



You can also restrict workflow operations to members of a custom field's group, with a custom field condition.

Preset group lists

Per-issue groups can also be chosen via select-lists, in addition to the group picker. Say you wish to restrict the **Assigned Group** to `jira-managers` or `jira-qa`. Simply create a select-list custom field with these two values, and add the select-list to the notification and permission schemes.

Wiki-Style Linking

Adding to the Wiki-style renderingfunctionality, users can now enrich their description, environment and comment entries with pertinent links to

other JIRA data:

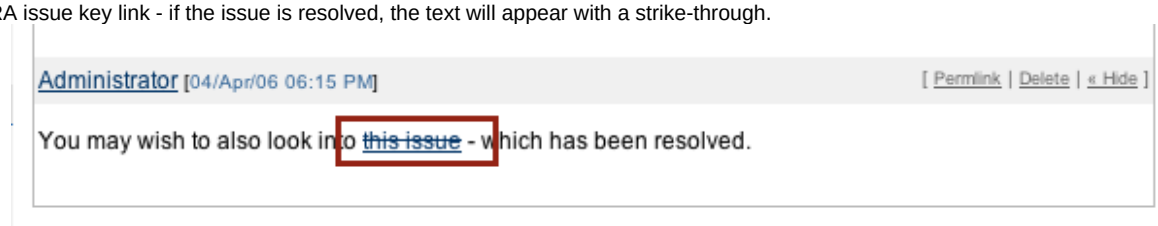
- User Aliasing
- Issue Aliasing

User Aliasing

Using the syntax *[~username]*, the username text will appear linked to the user profile of that user.

Issue Aliasing

Using the syntax *[some text here|TST-1]*, the text *some text here* will appear as a link to the specified issue. This text adopts the behaviour of a regular JIRA issue key link - if the issue is resolved, the text will appear with a strike-through.



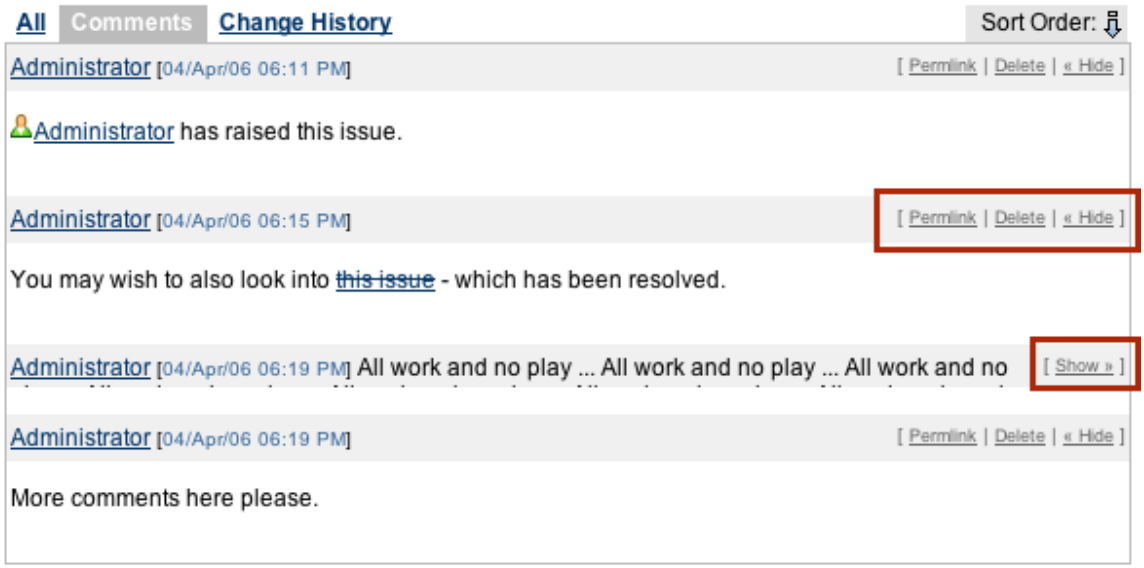
"I'm Feeling Lucky" Quick Search

Letting the user skip the results screen in the Issue Navigator of a search, the 'I'm Feeling Lucky' search presents the user with the first result associated with the search query. This seach can be engaged by pressing **Control + Enter** after submitting the search query to the 'Quick Search' query box.

Collapsible Fields

Enhancing the JIRA user interface, it is now possible to quickly configure the level of detail displayed while viewing an issue through collapsible and expandable issue fields.

The environment, description, individual comment fields and any textarea custom field can be *hidden* or *displayed* by simply clicking on the relevant link for a particular field - allowing fields with large amounts of data to be temporarily hidden while viewing other entries against that issue. JIRA only adds these options to fields that include a large amount of information.

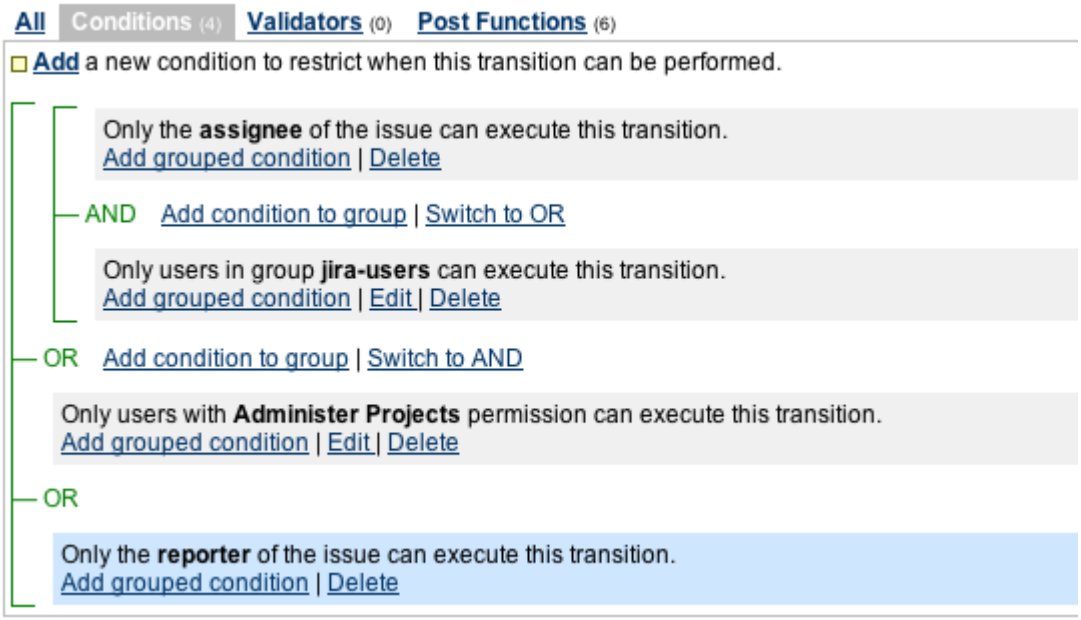


Nestable Conditions

Workflow conditions allow workflow designers the ability to restrict the availability of a workflow transition. The criteria of the condition must be met in order for the workflow transition to become available.

Workflow conditions can now be configured to combine criteria with boolean **OR** statements - allowing condition specification to match the workflow design more closely. For example, a condition can be constructed whereby it is only met if the user is the assignee of the issue **OR** the user is a member of the **jira-users** group.

Extending this concept, more complex condition criteria can be constructed with the ability to *nest* conditions in groups. Each group can be combined with other individual conditions or groups with the boolean **AND** or **OR** statements.



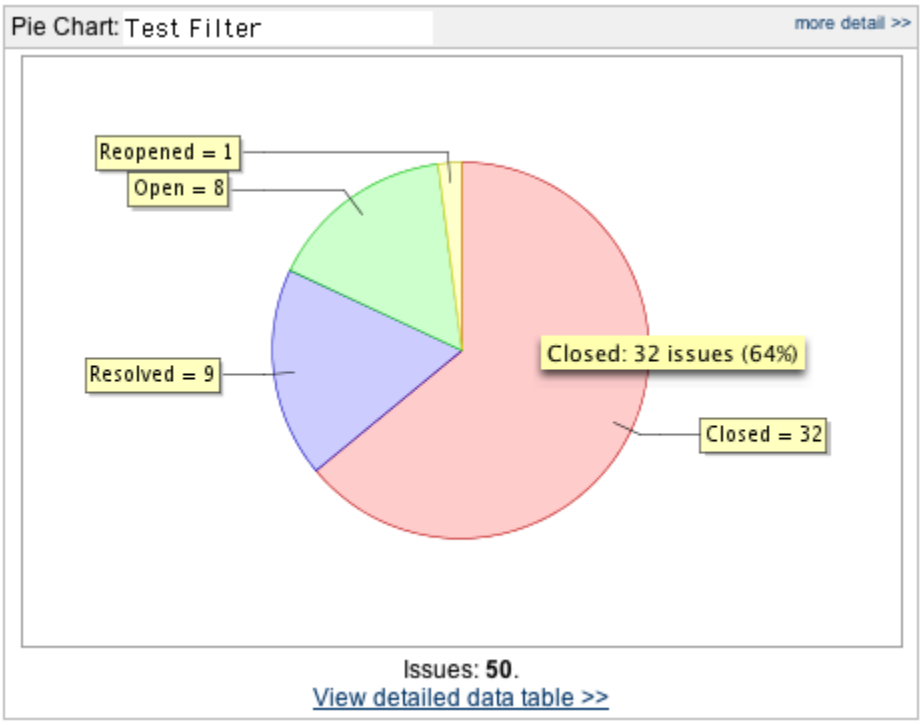
Charting Plugin Updates

The latest version of the Charting Plugin includes two new charting options:

- Pie Chart
- Average-Age Open Issues

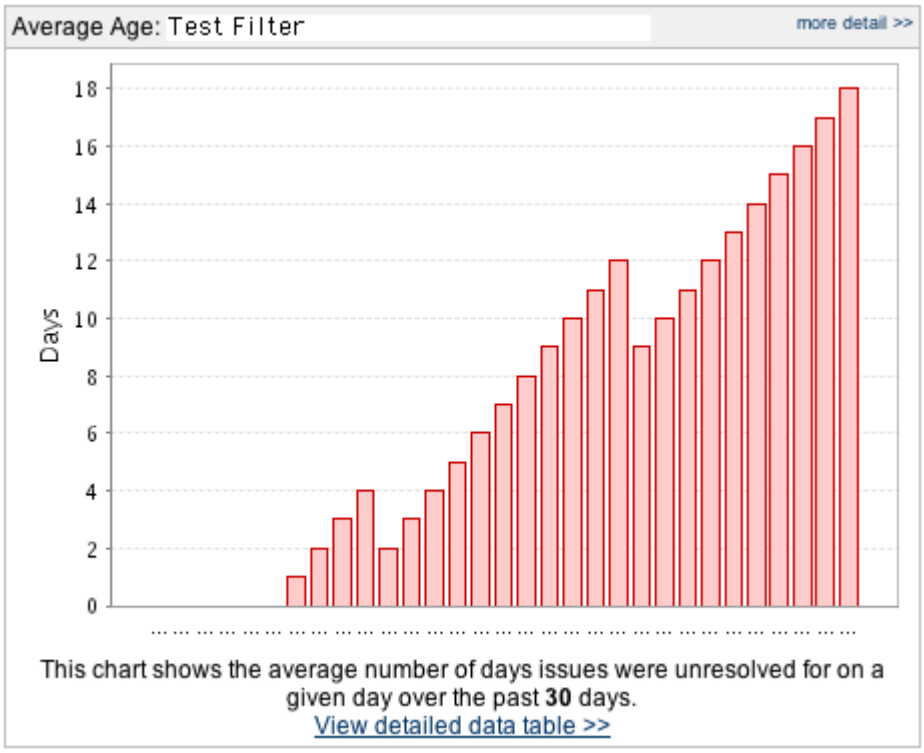
Pie Chart

Reports and portlets can be generated displaying data based on a statistic type (*e.g. Status, Priority, etc.*) of issues from a project or specified filter in pie-chart format.



Average-Age Open Issues

This chart displays the average-open-age of issues over a specified period with a configurable interval. This chart graphically conveys the trend for the average amount of time that issues remain unresolved.



Improvements

- Email Notification & Internationalization
- Performance
- Clone Issue Extensions
- Disable Notification for Bulk Operation
- Banner Visibility
- *... and many more ...*

Email Notification & Internationalisation

JIRA is shipped with over 15 language bundles - allowing the user to configure the language JIRA is displayed in globally and on a per-user basis. The internationalisation coverage now includes the email templates used in issue event notifications. Each template has been converted to allow full translations of the body of the email for all locales.

Taking JIRA into a truly global team environment, JIRA can now send individual email notifications to each recipient in *their user-profile selected language*. Hence, global team members located in regional sites around the world can be updated of issue events and updates with individually-tailored email notifications in their desired language.

This functionality becomes complete with the addition of properties files for the locales in use.

The translation process is greatly supported through the *much appreciated* efforts of the JIRA community. Many thanks to Gerd Gueldenast who has already provided a German translation for the email template properties - which is included in this release.

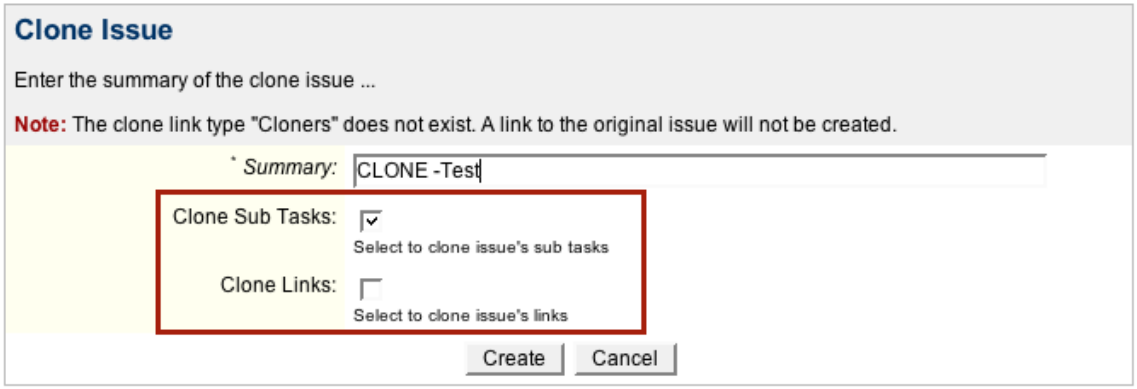
Performance

Improving overall response times while navigating through JIRA, this release introduces a caching servlet for all JavaScript and CSS resources. JIRA will cache these resources on the client system until the instance of JIRA is restarted. In-house testing and initial reports indicate that this change results in a significant performance improvement throughout JIRA.

Clone Issue Extensions

The **Clone Issue** functionality has been expanded with further configuration options. It is now possible to specify the level of detail cloned through:

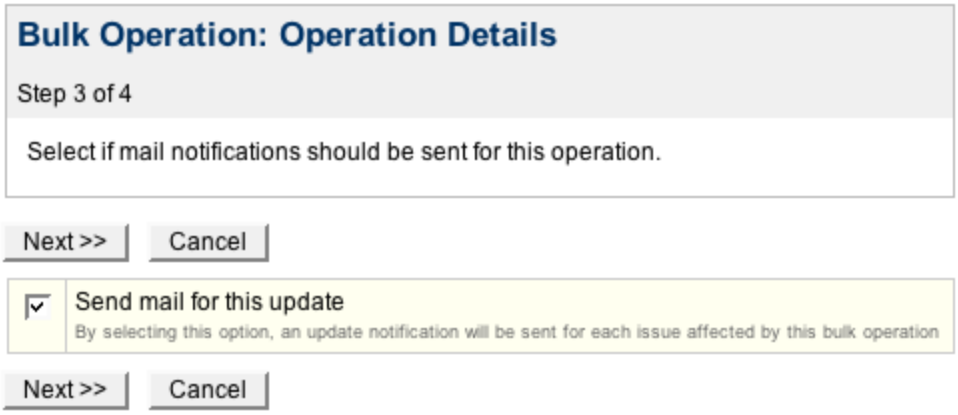
- Clone issue links - links between the clone issue and those linked to/from the original cloned issue will be recreated in the clone issue
- Clone sub-tasks - sub-tasks associated with the original cloned issue will be re-created for the clone issue



Disable Notification for Bulk Operation

JIRA administrators (and project administrators of the selected issues) can now configure whether notifications are sent for a bulk operation - avoiding mass emailing of all changes made during a bulk operation.

With this configurability, it is possible to complete updates on a collection of issues without generating unwanted email notification noise. The possible scenarios where this option can be applied are many - e.g. adding a new custom field and setting a default value, updating old issues ...



Banner Visibility

The announcement banner allows pertinent information to be displayed on all JIRA pages. This feature has been extended to allow configuration of the banner visibility level:

- **Public** - show the banner to anyone
- **Private** - show the banner to logged-in users only

... and many more ...

A number of other notable improvements included in this release ...

Mail Server & Bulk Precedence

The JIRA mail service can be configured to ignore emails with a *Precedence: bulk* header. This improvement eliminates potential issue/comment generation loops through the mail service from auto-generated emails.

Statable VersionPicker Field

JIRA provides a vast array of portlets that can populate your dashboard with instant access to issue information. The **Version Picker** custom field has been extended to allow inclusion in the 2-Dimensional Filter Statistics portlet - a portlet displaying the results of a search in table format with configurable axes.

Quick Sub-Task Creation Form

JIRA can now be configured so that the quick sub-task creation form is present on the *View Issue* screen at all times - even if the issue does not currently have any sub-tasks.

Issue Link Configuration

The issue link view can be configured to include various issue fields (e.g. issuetype, issuekey, etc.) - providing further details of the linked issue without having to navigate to that issue. Further, it is also possible to specify the sort order for this table.

Bug Fixes

This release includes a number of bug fixes - just ask JIRA to view the entire list!

Upgrading

In order to complete a successful upgrade, please refer to our Upgrade Guides. If you are upgrading from JIRA 3.5.3 please refer to the JIIRA 3.6 Upgrade Guide.

If you are upgrading from a pre-3.5.3 release, please refer to the relevant JIRA 3.x Upgrade Guides.

JIRA 3.6 Upgrade Guide

JIRA 3.6 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.x from JIRA 3.5.x. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Database Intensive Upgrade Task

To introduce the Custom events to JIRA, it was necessary to upgrade a large data set within JIRA's database for 3.5.x and earlier releases. Depending on the size of your JIRA data the upgrade task (number 150) might get your DBMS to do a lot of work which might take some time. The exact amount of time also depends on the processing power of the machine running JIRA's database.

Please be patient with the upgrade task and do not restart JIRA while the upgrade is in progress. The upgrade task will report on its progress to JIRA's log file as it upgrades your data.

The following is the sample output that the upgrade task will produce. As you can see the upgrade task took roughly 5 and a half minutes to modify over 660,000 records in the database.

```
11:14:09 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Phone Support Workflow
v.6'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow v.3'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Phone Support Workflow
v.7'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Test'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Copy of Support
Workflow'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow v.4'.
11:14:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Inspecting workflow 'Support Workflow'.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150]
************************************************************
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating 660453 records in the
'NotificationInstance' table.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] This might take a long time. Please do NOT
stop JIRA.
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150]
************************************************************
11:14:18 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_CREATED'.
11:15:12 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_UPDATED'.
11:15:51 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_ASSIGNED'.
11:16:10 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_RESOLVED'.
11:16:46 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_CLOSED'.
11:16:57 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_COMMENTED'.
11:18:57 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_REOPENED'.
11:19:17 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_DELETED'.
11:19:26 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_MOVED'.
11:19:31 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKLOGGED'.
11:19:37 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKSTARTED'.
11:19:41 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_WORKSTOPPED'.
11:19:43 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Updating records of type
'NOTIFICATION_ISSUE_GENERICEVENT'.
11:21:23 INFO [jira.upgrade.tasks.UpgradeTask_Build150] Update of 'NotificationInstance' records
finished.
```

Workflow Post Functions

Applies to	users with custom workflow XMLs saved on disk - external to JIRA

JIRA stores its workflows in the database. During the upgrade, these workflows will be upgraded automatically. However, if you have stored your workflows on disk (outside the database), you will need to follow these instructions to upgrade the workflows manually.

Previously, workflow post functions referenced the event to fire through a string value of the event name. All post functions now reference the event through a numeric ID value. As mentioned, all workflows stored within JIRA will be automatically updated. However, all workflows saved to disk - external to JIRA - should be updated manually as follows. The actual workflow XML file should be updated as follows:

For each workflow post function that accepts the event ID as an argument:

1. The value of the **name** attribute of the **arg** tag has to be changed from **eventType** to **eventTypeId**
2. The body of the **arg** tag has to change according to the following table:

Event Name	Event Type Id

created	1
updated	2
assigned	3
resolved	4
closed	5
commented	6
reopened	7
deleted	8
moved	9
worklogged	10
workstarted	11
workstopped	12
genericevent	13

By default, the only post functions that accept event IDs are *FireIssueEventFunctions.* Therefore, unless you have implemented your own custom post function that also deals with events, you will only need to update the arg tags for the *FireIssueEventFunctions* everywhere in the workflows.

For example, *FireIssueEventFunction* for create issue workflow transition looked like:

```
<function type="class">
    <arg name="class.name">com.atlassian.jira.workflow.function.event.FireIssueEventFunction</arg>
     <arg name="eventType">created</arg>
</function>
```

and needs to be changed to:

```
<function type="class">
    <arg name="class.name">com.atlassian.jira.workflow.function.event.FireIssueEventFunction</arg>
     <arg name="eventTypeId">1</arg>
</function>
```

Custom Events

Applies to	users who have **modified JIRA source code or added custom code to define new notification events**. Also of interest to users wishing to define new notification templates

Releases before JIRA 3.6 did not allow users create custom events. If you have modified the JIRA source to add custom events - please follow these instructions.

If you have previously defined a custom event within JIRA - it is necessary to add appropriate entries to the following files:

- `system-event-types.xml`- used to install and upgrade all event types within the system to the new 3.6 event type object.
- `email-template-id-mappings.xml` - maps the event id to an assocaited velocity template file.

The `system-event-types.xml` file requires name and description details of the previously added custom event. For example, if the custom event type "Issue Frozen" was added to the system - the following entry should be added to the XML file:

```
<eventtype id="10000">
   <name>Issue Frozen</name>
   <description>This is the 'Issue Frozen' event type.</description>
   <notificationName>ISSUE_FROZEN</notificationName>
   <eventName>issuefrozen</eventName>
</eventtype>
```

The elements provide the following information:

- id - the new id for the event type. **All custom event types should be added from ID 10000 and above**
- notificationName - the original name for the event as found in the Notification table
- eventName - the origianl name for the event as found in workflows

The `email-template-id-mappings.xml` file requires an entry mapping the new custom event to an associated velocity email template. This mapping is used when a notification is sent for this event. Following from the above example, the following entry would be made:

```
<templatemapping id="10000">
   <name>Issue Frozen</name>
   <template>issuefrozen.vm</template>
</templatemapping>
```

The id should match that of the event as specified in the `system-event-types.xml` The `template` entity should reference the Velocity template to be used in email notifications of this event. A HTML and text version should be provided in the appropriate directory (html or text) at:

`<JIRA>/src/etc/java/templates/email/`

All custom event types added to the file *system-event-types.xml* should be added with an ID of 10000 and above

Custom Listeners

Applies to	users who have **added custom listeners** to JIRA.

For all users who have **added custom written** listeners to JIRA, it might be necessary to update the listener to follow the new JIRA 3.6 API.

There are two things to look out for:

1. signature change of the *workflowEvent* method
2. change of return type of *getIssue()* method on the *IssueEvent* object

The signature of the method *workflowEvent* in the *IssueEventListener* has changed from:

```
public void workflowEvent(int type, IssueEvent event);
```

to:

```
public void workflowEvent(IssueEvent event);
```

Note: the type parameter has been removed.

If you have implemented *IssueEventListener* directly or have extended *AbstractIssueEventListener* and have overridden the method *workflowEvent*, you will need to change and recompile your listener before installing JIRA 3.6.

In JIRA 3.6, the event type ID can be retrieved by calling the following method on the *IssueEvent* object:

```
Long eventID = event.getId();
```

However, the returned value of the *getId()* method is different to the values of the type parameter that was passed to the *workflowEvent* method. The following table represents these differences:

Event Name	Old ID	New ID

created	0	1
updated	1	2
assigned	2	3
resolved	3	4
closed	4	5
commented	5	6
reopened	6	7
deleted	7	8
moved	8	9
worklogged	9	10
workstarted	10	11
workstopped	11	12
genericevent	-1	13

Also, the *getIssue()* method of the *IssueEvent* object has changed to return an Issue object instead of a *GenericValue* object representing an issue.

Users who have created and added custom listeners must update the listener to now operate with the *Issue* object. For example:

```
Issue issueObject = event.getIssue();
```

As a quick fix, you can modify your listener to use `event.getIssue().getGenericValue()`.

The event type ID constants are now only available from the class *EventType*. Any use of the original constants must be updated to use the *EventType* constants. For listeners that reference an event ID by its numeric value - it is necessary to ensure that the IDs now match those as defined in *EventType*.

Custom permission types

Applies to	users who have modified JIRA source to add new permission types (ie. in addition to the standard 'user', 'group', 'assignee' types).

The SecurityType interface, used to implement permission types ('single user', 'group' etc) has had a `getUsers()` method added. If you have implemented your own SecurityType you will need to implement this. See the source of current implementations (eg. `GroupCF`) for tips.

Plugin upgrades required

As usual, you should check whether the plugins you use are compatible with the new release. Generally, plugins (like the Subversion plugin or JIRA toolkit) need to be upgraded when JIRA is upgraded. See the list of plugins at:

http://confluence.atlassian.com/display/JIRAEXT/Home

JIRA 3.6.5 Release Notes

JIRA 3.6.5 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.6.5 in Standard, Professional and Enterprise editions. This point release includes 8 bug fixes and improvements, in particular improvements in performance and efficiency. It can be downloaded here.

Not using 3.6? Learn about all the **new features you're missing out on**!

If upgrading from an earlier version please read through the JIRA 3.6.5 Upgrade Guide.

JIRA 3.6.5 includes 8 bug fixes and improvements.

JIRA Issues (10 issues)					
Type	Key	Summary	Priority	Status	Resolution
	JRA-11267	Stacktrace: "java.lang.ClassCastException: java.lang.String" when viewing custom fields		Resolved	Fixed
	JRA-11063	"Add a new Issue Security scheme" link in SelectProjectIssueSecurityScheme is empty		Resolved	Fixed
	JRA-11036	Fix MailHandler attachment code - CPU inefficient		Resolved	Fixed
	JRA-11029	Unable to edit custom fields in workflow transition view in "Bulk change" mode		Resolved	Fixed
	JRA-11014	UnSynced map/cache in AbstractSchemeManager.cacheProjectSchemes		Resolved	Fixed
	JRA-10990	"Edit Configuration" on the "Default Configuration Scheme" for a "Custom Field" I created causes a crash (upgrading from 3.6.3 to 3.6.4) for "All Issue Types"		Resolved	Fixed
	JRA-10951	Mystery stacktraces: IllegalArgumentException in CustomFieldTypeModuleDescriptor.getEditHtml		Resolved	Fixed
	JRA-10907	Caching the user's projects improves the speed of Issue Navigator		Resolved	Fixed
	JRA-10223	Ensure transaction thread local gets cleared for every request and service run (potential deadlock fix)		Resolved	Fixed
	JRA-6744	Issue details pane does not stretch properly when reporter name is a long single word		Resolved	Fixed

JIRA 3.6.5 Upgrade guide

JIRA 3.6.5 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.5 from JIRA 3.6.4. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading to JIRA 3.6.5 from JIRA 3.6.4.

JIRA 3.6.4 Release Notes

JIRA 3.6.4 Release Notes

Atlassian Software Systems is proud to announce the release of JIRA 3.6.4 in Standard, Professional and Enterprise editions. This point release includes over 20 bug fixes and improvements, in particular improvements in performance, efficiency and security. It can be downloaded here.

Not using 3.6? Learn about all the **new features you're missing out on**!

If upgrading from an earlier version please read through the JIRA 3.6.4 Upgrade Guide.

JIRA 3.6.4 includes over 20 bug fixes and improvements.

JIRA Issues (25 issues)					
Type	Key	Summary	Priority	Status	Resolution
	JRA-10199	Need to make move issue in Professional edition show the issue type since we allow issue type schemes in Pro		Resolved	Fixed
	JRA-10700	Link types drop-down on Create Link screen shows "jira_subtask" links		Resolved	Fixed
	JRA-10762	Connection is Closed errors if the ThreadLocal connection gets closed before a commit or rollback.		Resolved	Fixed
	JRA-10833	Ensure the OfBiz Iterator is closed before the transaction is commited (i.e. connection closed)		Resolved	Fixed

	JRA-5038	Porlets - cannot specify Portlet name without i18n string and associated property.		Resolved	Fixed
	JRA-10308	Giving Manage Watchers Permission to Reporter or Current Assignee allows anyone who can view the watchers list to edit it		Resolved	Fixed
	JRA-10535	Hide/show of Free Text Field Custom Field does not work properly in Issue Navigator		Resolved	Fixed
	JRA-3981	Change error message when user does not have global USE permission		Resolved	Fixed
	JRA-10595	SOAPUtils's transformToStringArray() always returns an empty String array		Resolved	Fixed
	JRA-9418	setArchived and setReleased in RemoteVersion does nothing		Resolved	Fixed
	JRA-7831	Changing Workflow should not change Date Updated or provide option not to		Resolved	Fixed
	JRA-10662	Progress Status bar is broken in IE & Opera		Resolved	Fixed
	JRA-10637	No Manage Filters link in saved filter popup		Resolved	Fixed
	JRA-10695	NullPointerException when sharing a filter		Resolved	Fixed
	JRA-11080	Version custom fields not available for use in statistics portlets (not statable)		Resolved	Fixed
	JRA-10601	XML-RPC to fix/improve String casting		Resolved	Fixed
	JRA-10785	Integrity Checker leaks a database connection if an exception occurs during a check		Resolved	Fixed
	JRA-8405	Priority levels popup prompts for login		Resolved	Fixed
	JRA-7213	SOAP Interface - getAttachmentNames(...) always returns null		Resolved	Fixed
	JRA-10336	After changing the icon on a custom Issue Type, related custom fields disappeared from the different screens		Resolved	Fixed
	JRA-10494	Bugzilla import looses descriptions of components		Resolved	Fixed
	JRA-10421	NPE rendering work logged email with '0m'		Resolved	Fixed
	JRA-10468	Add comment of using JVM version 1.5_06-b05 in "Installing Java" section of Documentation.		Resolved	Fixed
	JRA-3363	"Default Locale" is not listed under "Installed Locales"		Resolved	Fixed
	JRA-10789	typo in admin projects page		Resolved	Fixed

JIRA 3.6.4 Upgrade guide

JIRA 3.6.4 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.4 from JIRA 3.6.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading to JIRA 3.6.4 from JIRA 3.6.3.

JIRA 3.6.3 Release Notes

JIRA 3.6.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.6.3 in Standard, Professional and Enterprise editions. This point release includes over 30 bug fixes and improvements, in particular improvements in performance, efficiency and security. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.6.3 Upgrade Guide.

JIRA 3.6.3 includes over 30 bug fixes and improvements.

JIRA Issues (36 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-9090	Wiki Renderer does not HTML encode contents of unknown macros		Resolved	Fixed
	JRA-10276	Custom field is displayed during Bulk Edit and Bulk Transition when it does not have a context for some of the issues.		Resolved	Fixed
	JRA-10317	Bundle JIRA Calendar plugin correctly.		Resolved	Fixed
	JRA-10590	Fix PERF-95 - address Perforce depots with 'gaps' in the commit numbers		Resolved	Fixed
	JRA-10105	jsessionid twice in URL causes Fix and Affects version links to break in issue view screen during session's first page view		Resolved	Fixed
	JRA-10625	Saved Filter is broken in 3.6.3 #159		Resolved	Fixed
	JRA-9296	Indexing intermittently fails with IndexException		Resolved	Fixed
	JRA-10542	Request parameters are not HTML encoded on the 500 page		Resolved	Fixed
	JRA-10504	The CombinedCachingServlet can become a bottleneck under heavy load		Resolved	Fixed
	JRA-10567	Investigate caching 'EditActions' on Bulk Workflow Transition 'edit fields' screen		Resolved	Fixed
	JRA-9933	Digital signature as an attachment		Resolved	Fixed
	JRA-3349	More Information on files for the Version Control Tab		Closed	Fixed
	JRA-10178	Message custom field produces change history entries		Resolved	Fixed
	JRA-10341	Adding New Version does not show up in Open Issues or via SOAP until Restart		Resolved	Fixed
	JRA-10253	Indexing the subtasks is inefficient		Resolved	Fixed
	JRA-10061	Access to schemeItems in FieldScreenSchemeImpl needs to be synchronized		Resolved	Fixed
	JRA-10487	Upgrade to latest Atlassian Renderer		Resolved	Fixed
	JRA-10263	Description is not shown in full content view of Issue Navigator		Resolved	Fixed

	JRA-10609	Ability to sort 2DStats portlet by totals and limit results		Resolved	Fixed
	JRA-10434	ClassCastException in com/atlassian/jira/issue/comparator/IssueKeyComparator		Resolved	Fixed
	JRA-10304	Fix handling of null users present in database		Resolved	Fixed
	JRA-9336	NPE on viewing saved filters if the user is not logged in		Resolved	Fixed
	JRA-10552	Cancel button in Comment does not have tooltip		Resolved	Fixed
	JRA-10505	Second dropdown list of Cascading Select custom field does not get updated during Bulk Workflow Transition		Resolved	Fixed
	JRA-10518	FieldScreenImpl.getTab(...) throws IndexOutOfBoundsException		Resolved	Fixed
	JRA-10488	FilterStatisticsValuesGenerator retains references to managers through static final references		Resolved	Fixed
	JRA-8794	Unable to CreateIssue or Comment as another user via SOAP (eg. set reporter)		Resolved	Fixed
	JRA-10237	Customfields belonging to specific projects are shown when those projects are selected		Resolved	Fixed
	JRA-10394	NullPointerException when searching on Cascading Select		Resolved	Fixed
	JRA-10526	Support Date custom fields in Mantis 1.0.3+		Resolved	Fixed
	JRA-10289	Permlink not translated in e-mails		Resolved	Fixed
	JRA-10272	"Restore Data from XML" doesn't seem to flush SOAP cache		Resolved	Fixed
	JRA-10392	Full Content (HTML) View doesn't show description		Resolved	Fixed
	JRA-10335	Status names are not translated in the tooltip on the Browse Project page		Resolved	Fixed
	JRA-10499	In Firefox, the Home page bar-graph links to Issue Navigator fail		Resolved	Fixed
	JRA-10450	View issue details (Word export) should contain issue status		Resolved	Fixed

JIRA 3.6.3 Upgrade guide

JIRA 3.6.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.3 from JIRA 3.6.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.6.3 from JIRA 3.6.2.

JIRA 3.6.2 Release Notes

JIRA 3.6.2 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.6.2 in Standard, Professional and Enterprise editions. This point release includes over 40 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.6.2 Upgrade Guide.

JIRA 3.6.2 includes over 40 bug fixes and improvements.

JIRA Issues (47 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-9791	Error unarchiving Jira on Mac OS X		Resolved	Fixed
	JRA-10109	Second select box of cascading select custom field not populated correctly during Bulk Edit		Resolved	Fixed
	JRA-10049	Write functional tests for subversion plugin and automate them in the nightly build		Resolved	Fixed
	JRA-10086	create upgrade task to clean out erroneous statuses in workflow transitions with SubTaskBlockingCondition		Resolved	Fixed
	JRA-10167	Bulk Edit allows you to edit issue type even if the workflows are different		Resolved	Fixed
	JRA-9864	OutOfMemoryErrors when running Confluence and Jira on the same server		Resolved	Fixed
	JRA-10065	Cascading select custom field doesn't pop up in the "Move Issue" screen if the field has a context in source and destination projects.		Resolved	Fixed
	JRA-10006	Tomcat Config: Increase *maxActive* database pool connection limit		Resolved	Fixed
	JRA-10007	Docs: Add documentation on DB connection pool size		Resolved	Fixed
	JRA-10083	Exporting an issue to word displays descrption twice		Resolved	Fixed
	JRA-9711	Excel export layout became worse with the new JIRA version		Resolved	Fixed
	JRA-10052	Create a script that calculates the number of classes that use GenericValues		Resolved	Fixed
	JRA-10133	move patched jar to build server and distribute to ImaHima		Resolved	Fixed
	JRA-10101	Review 3.6.1 Bugs		Resolved	Fixed
	JRA-9812	Project cache refreshed incorrectly after creating project by remote API		Resolved	Fixed
	JRA-10113	Upgrade to latest atlassian-extras		Resolved	Fixed
	JRA-9324	Groups in group browser encoded with system encoding		Resolved	Fixed
	JRA-9377	Unable to set default value in Multi User Picker field		Resolved	Fixed
	JRA-10087	Add doc that *no count* setting should not be checked for MS SQL Server		Resolved	Fixed

	JRA-10204	Remove I'm feeling luck from quick search as no-one uses it, and it causes problems on IE 7		Resolved	Fixed
	JRA-10215	CustomFieldManager: getCustomFieldObjects passes issue ID rather than project ID		Resolved	Fixed
	JRA-10148	Subversion Plugin: Repository URLs with no realtive repository path not working		Resolved	Fixed
	JRA-3969	Services should be disabled during reindexes		Resolved	Fixed
	JRA-9724	Session Timeout/Logout with tabbed dashboard broken		Resolved	Fixed
	JRA-9928	Status names with apostrophes generate exceptions on *Browse Projects* page		Resolved	Fixed
	JRA-10084	Update the Issue Security Scheme Documentation		Resolved	Fixed
	JRA-7952	Unable to locate documentation reference		Resolved	Fixed
	JRA-10212	TestUpgradeTask_Build56 should fetch the e-mail address from external-link.properties rather than hard code the value		Resolved	Fixed
	JRA-9934	Sub-Task blocking condition is not displayed correctly in Transition Condition screen after it has been added.		Resolved	Fixed
	JRA-10076	Screenshot attachment applet cause closed session on Websphere 6.0		Resolved	Fixed
	JRA-9961	Session Timeout caused in Bulk Operations		Resolved	Fixed
	JRA-9944	Trackbacks - send outgoing pings to all issues is not reflected in trackback view screen.		Resolved	Fixed
	JRA-9946	The projects won't appear in browse project view after Bugzilla import		Resolved	Fixed
	JRA-10116	JavaScript error on everypage with IE 7		Resolved	Fixed
	JRA-9883	Deleting a User doesn't remove them from the Component Lead		Resolved	Fixed
	JRA-10055	Attachment creation failure causes message handler to loop over message indefinitely.		Resolved	Fixed
	JRA-10213	Version custom fields will munge their data when going through a move or bulk move		Resolved	Fixed
	JRA-10103	Type in Danish Email Translation		Resolved	Fixed
	JRA-10112	Update Screenshot of the Setup Wizard part in the docs		Resolved	Fixed
	JRA-10176	Update docs to mention that the drop down list restricting comment and work log view only contains user's groups		Resolved	Fixed
	JRA-10030	Adjust layout formatting for HTML left column on Issue Details page		Resolved	Fixed
	JRA-9938	JIRA Docs - Trackbacks - Technical Info section displayed twice.		Resolved	Fixed
	JRA-10111	Remove JiraLuceneFieldCache cache		Resolved	Fixed

	JRA-10142	Typo in an error message		Resolved	Fixed
	JRA-10144	Need to synchonize a hashmap in the ofbiz memoryHelper that we use for unit tests		Resolved	Fixed
	JRA-9719	Update notifications doco		Resolved	Fixed
	JRA-10082	Translation error in notifications for German		Resolved	Fixed

JIRA 3.6.2 Upgrade guide

JIRA 3.6.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.2 from JIRA 3.6.1. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Maximum Active Databased Connections

Applies to	JIRA Standalone users

In version of JIRA before 3.6.2, the maximum number of database connections was limited to 8 by default. If JIRA was used by more than 8 concurrent users or under very haeavy usages, the users could experience delays or JIRA could hang.

In JIRA 3.6.2 the default number of maximum active database connections has been increased to 20. When upgrading to JIRA 3.6.2, please ensure that your database will allow JIRA to establish 20 connections, or decrease this number to desired value. To adjust the number of connections change the value of the `maxActive` attribute of the `jdbc/JiraDS` resource in `config/server.xml` file. JIRA has to be restarted to apply the change.

JIRA 3.6.1 Release Notes

JIRA 3.6.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.6.1 in Standard, Professional and Enterprise editions. This point release includes over 30 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.6.1 Upgrade Guide.

JIRA 3.6.1 includes over 30 bug fixes and improvements.

JIRA Issues (32 issues)					
Type	**Key**	**Summary**	**Priority**	**Status**	**Resolution**
	JRA-9903	JIRA doesn't build from source		Resolved	Fixed
	JRA-9967	Invalid workflow action with 'Assignee only' condition and AssignTo field on transition screen		Resolved	Fixed
	JRA-9932	Bug in Find issues - sub tasks field causes error		Resolved	Fixed
	JRA-9820	Danish Property Filtes for 3.6		Resolved	Fixed
	JRA-9889	Version Picker Custom Field can't be CSV imported with value or custom field ID		Resolved	Fixed
	JRA-9811	jira:CreateIssue via Jelly does not allow specifying multiple components		Resolved	Fixed

	JRA-9942	Issue Links in column view not escaped		Resolved	Fixed
	JRA-9887	Merge this fix into the 3.6 branch		Resolved	Fixed
	JRA-9962	Create link to Services page on Email Handler doc		Resolved	Fixed
	JRA-9993	NullPointerException when component has no lead		Resolved	Fixed
	JRA-8954	Ensure JIRA unit (and functional tests?) pass on machines with locate set to other than en_* e.g. en_US or en_AU		Resolved	Fixed
	JRA-9973	Update Traditional Chinese Pack for 3.6.x		Resolved	Fixed
	JRA-10037	Set default email format to text		Resolved	Fixed
	JRA-10022	admin.jsp throws Exception in SunOne Application Server 7		Resolved	Fixed
	JRA-9550	When editing workflow transition with no transition view, transition view is set (seemingly) randomly in editor screen.		Resolved	Fixed
	JRA-9894	IndexOutOfBoundsException in Average Age Report and Pie Chart Report		Resolved	Fixed
	JRA-10027	Synchronize the getTemplateFilename method from TemplateManager		Resolved	Fixed
	JRA-9975	update firebird entitymodel		Resolved	Fixed
	JRA-12899	FixVersions parameter is being ignored in the CreateIssue tag		Resolved	Fixed
	JRA-9806	NullPointerException when commenting on an issue through edit screen		Resolved	Fixed
	JRA-9922	ConcurrentModificationException if you move two portlets on the dasboard at the same time		Resolved	Fixed
	JRA-9971	Update workflow PostFunction does not show all users		Resolved	Fixed
	JRA-9813	Add log location to 500 page		Resolved	Fixed
	JRA-4945	Clicking the Log In link creates ever longer URLs		Resolved	Fixed
	JRA-9969	Minor email-translation issue		Resolved	Fixed
	JRA-9957	Need to make a clean way of deploying the jira dev kit		Resolved	Fixed
	JRA-9974	Please enable localization of Constants Help pop-up page		Resolved	Fixed
	JRA-9905	Contain HSQL warning in the Admin portal		Resolved	Fixed
	JRA-9851	Image thumbnails displayed for (lowercase) png images but not displayed for (uppercase) PNG images		Resolved	Fixed
	JRA-9854	Cannot find a key for a text to translate		Resolved	Fixed

	JRA-9839	Misspelling in SimpleWorkflowManager error - "woth" should be "with"		Closed	Fixed
	JRA-9991	JIRA help link for "version management" is incorrect		Resolved	Fixed

JIRA 3.6.1 Upgrade Guide

JIRA 3.6.1 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.6.1 from JIRA 3.6. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here. When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.6.1 from JIRA 3.6.

JIRA 3.5 Release Notes

JIRA 3.5 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems, recipient of the Deloitte Fast 500 Award, is proud to announce the latest release of *the* issue tracking and project management application - JIRA 3.5 (download it here).

The latest release includes over 50 powerful new features and improvements along with over 50 bug fixes.

Upgrade Information

In order to complete a successful upgrade, please refer to our Upgrade Guides. If you are upgrading from JIRA 3.4.3 please refer to the JIRA 3.5 Upgrade Guide.

If you are upgrading from a pre-3.4.3 release, please refer to all JIRA 3.x Upgrade Guides.

Contents

- New Features
- Improvements
- Bug Fixes

New Features

JIRA 3.5 includes an impressive array of new features and improvements - some of which are noted below. To view the entire new feature and improvement list - ask JIRA!

- Bulk Workflow Transition
- FogBugz Importer
- Charting Plugin
- MS Word Export
- JIRA Page Linker Plugin
- Component Lead Notification Type

Bulk Workflow Transition

JIRA 3.5 extends the bulk operation capabilities with the addition of **Bulk Workflow Transition** - allowing a selected collection of issues to be advanced through the associated workflow.

The **Bulk Workflow Transition** process is as simple as if transitioning one issue. Once the collection of issues is retrieved through the issue navigator, it is possible to progress the issues through the associated workflow. The associated screen for the transition is displayed to the user - allowing all related fields to be edited as normal. All edits (including comments) are applied to each issue and each issue is advanced to the associated workflow status. All field configurations, workflow conditions, validators and post functions are respected throughout the process.

!bulkworkflow.png|align=center!

FogBugz Importer

Users of the FogBugz issue tracking system can now easily import their FogBugz data into JIRA through the new configurable import wizard. The import wizard allows the user specify which projects, custom fields and issue links are to be imported.

!fogbugzimporter.png|align=center!

Charting Plugin

The ability to visually represent JIRA data in a graphical form is a key requirement for many JIRA users. With the **Charting Plugin**, users can translate their data into graphs and charts.

Allowing for more meaningful presentations with greater impact, graphs and charts are more easily absorbed than plain numerical data. Over time, users can also plot trend graphs - with the prospect of extrapolating and interpreting the graphical data to predict future trends and workloads.

The plugin can be downloaded here.

!chart-plugin.png|align=center!

MS Word Export

It is now easier to create Word document presentations, with the ability to export JIRA data to Microsoft Office Word format. It is possible to export an individual issue or the 'Full Content View' of any search to a Word document.

!word-export.png|align=center!

JIRA Page Linker Plugin

Providing further integration between JIRA and Confluence, the **JIRA Page Linker Plugin** provides a custom field plugin for linking a JIRA issue with a Confluence URL.

While you are creating or editing a JIRA issue, you can bring up a popup window that will allow you to quickly search a Confluence site for pages you would like to link to this issue. Clicking a link in the popup window will add the page to your new issue.

The JIRA Page Linker plugin can be downloaded here.

!linker.png|align=center!

Component Lead Notification Type

JIRA 3.5 Enterprise introduces the notification type of **Component Lead**. With this option, notification schemes can be simplified through the specification of the **Component Lead** as a recipient of issue update emails. For each event specified in the scheme, the **Component Lead** will receive an email update - ensuring that the correct users are alerted.

!componentlead.png|align=center!

Improvements

- Bulk Assignment of Users to Groups
- Jelly Improvements
- SOAP Improvements
- Configure CSV Delimiter
- Mantis Custom Field Import
- Plugin System Improvements
- Configure Email Address Format
- Internationalisation
- Multi-Select Version Picker Custom Fields in Filters
- Delete Trackbacks
- Configured Navigator Columns in Email Filter Subscription
- Convert Bugzilla Link to JIRA Link

Bulk Assignment of Users to Groups

Group management has been greatly simplified with the addition of **Bulk Assignment of Users to Groups** - allowing multiple users to be added or removed to a group at once.

!bulkgroups.png|align=center!

Jelly Improvements

The Jelly tag **RunSearchRequest** now accepts a filter ID - the ID of the filter to be executed. This tag allows one to execute any saved Search Request and then use its results in any JIRA Jelly script.

For instance, one could define a filter identifying all old/inactive issues, and write a Jelly script to move them to an 'Inactive' state (see the Jelly docs for examples). This Jelly script can be scheduled to run periodically with the Jelly service.

SOAP improvements

This release includes various SOAP improvements - including the ability to retrieve via SOAP:

- a list of general configuration properties - i.e. determine if attachments, time tracking, voting, unassigned issues, sub-tasks, issue linking or watching is turned on/off
- an issue by its ID
- a match count for a search filter - useful to determine how results should be displayed - e.g. full or partial list of results

Configure CSV Delimiter

Some versions of MS Office Excel (e.g. German) save CSV files with separating values using '**;**' instead of '**,**'. The CSV Import Wizard has been improved to allow the delimiter to be specified for a particular import file.

Mantis Custom Field Import

Mantis 0.18+ has a basic custom field implementation. The JIRA Mantis Importer can now import any defined custom fields.

Plugin System Improvements

The ability to create more powerful plugins is now possible through the following improvements:

- Reference downloadable external resources. Additional static files such as images, Javascript or CSS can be served through **Downloadable Plugin Resources**. Further details available here.
- The plugin configuration screen has been improved with the ability to include a checkbox parameter. Further details available here.
- Servlet Plugin module enables users to deploy Java servlets as part of a plugin. Further details available here.

Maybe these improvements will be of use to the entrants of the Codegeist Competition?

Configure Email Address Format

Previously, it was necessary to stop JIRA and edit a properties file to modify the email address format. JIRA 3.5 now allows this modification to be made through the web interface - without the need to stop the server.

Internationalisation

The **Administration** section of the JIRA User Interface is now fully internationalized. This makes it possible to translate and present the entire JIRA web interface in a particular language, once the appropriate set of language files are included.

Further details on the translation process and how to get involved are available here. The continued support of those users who volunteer their translation skills is greatly appreciated.

Multi-Select Version Picker Custom Field

A new multi-select **Version Picker Custom Field** is now available - allowing this field to store multiple versions related to the associated project. It is also possible to use this field within a search through the Issue Navigator.

Delete Trackbacks

Trackback management has been improved to allow users delete specific trackbacks.

Configured Navigator Columns in Email Filter Subscription

In JIRA Enterprise edition, it is possible to configure which issue fields are displayed to the user for each saved filter. Previously, this configuration was only used throughout the web interface. JIRA 3.5 Enterprise now respects the chosen fields when sending e-mail notifications to subscribers

of a filter.

Convert Bugzilla text links to JIRA Issue links

On import, original Bugzilla text links will be linked to the new JIRA issues. This improvement helps ease the migration to JIRA for users who still use the Bugzilla bug IDs.

Bug Fixes

This release includes over 50 bug fixes. To view the entire bug fix list - ask JIRA!

JIRA 3.5 Upgrade Guide

JIRA 3.5 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5 (release notes) from JIRA 3.4.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

JIRA 3.5 Jira Service extension

- If you have implemented a custom JIRA service you need to be aware of the following API change.

In JIRA 3.5 the `getName()` and `setName(String name)` methods was added to the `com.atlassian.jira.service.JiraService` interface. This method should return and set the name of the service respectively. The name of the service can be used to identify a service uniquely. (Fixed made due to JRA-8352 bug)

Therefore, if you have implemented this interface, you will need to implement these methods and recompile your service(s) before deploying it into JIRA 3.5. If you have extended a JIRA class instead, e.g. `com.atlassian.jira.service.AbstractService` or `com.atlassian.jira.service.JiraServiceContainer` you do not need to modify your custom services.

Introduction of global **Bulk Change** permission

JIRA 3.5 introduces the global **Bulk Change** permission. This permission governs the ability to execute the bulk change operations:

- Workflow Transition
- Edit
- Move
- Delete

An upgrade task has been added to grant the new **Bulk Change** permission to all groups with the global **JIRA Users** permission.

The JIRA documentation includes further details on this new permission.

The decision to grant the **Bulk Change** permission should be considered carefully - the permission permits a user to modify a collection of accessible issues at once. For example, in JIRA installations configured to run in 'Public' mode (anybody can sign up and create issues), a user could comment on all accessible issues with the **Bulk Change** and **Add Comments** permission. Undoing such modifications may not be possible through the JIRA UI and may require changes made directly against the database.

CustomFieldPersister changes

CustomFieldPersister is used to store custom field values to database. The methods of this class has been refactored to remove the redundant parameter, `defaultValueMarker`. For example, the create values method went from:

```
void createValues(CustomField field, Long issueId, String defaultValueMarker, PersistenceFieldType
persistenceFieldType, Collection values, String parentKey);
```

to:

```
void createValues(CustomField field, Long issueId, PersistenceFieldType persistenceFieldType,
Collection values, String parentKey);
```

You will need to update and recompile any `CustomFieldType` that you wrote to use this new interface.

VersionCFType Changes

This affects plugin writers who uses the version custom field `VersionCFType`. The change is that previously the Transport Object type was a single `Version` object, but it is now a collection that contains a single `Version` object.

This was done to handle an improved version custom field which can be a mutli-select version custom field as well

JIRA 3.5.3 Release Notes

JIRA 3.5.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.5.3 in Standard, Professional and Enterprise editions. This point release includes over 20 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.5.3 Upgrade Guide.

JIRA 3.5.3 includes over 20 bug fixes and improvements.

JIRA Issues (23 issues)

Type	Key	Summary	Priority	Status	Resolution
	JRA-9691	jira.projectkey.warning and jira.projectkey.description properties not used		Resolved	Fixed
	JRA-9689	Jira says JAVA_HOME contains spaces even if it doesn't		Resolved	Fixed
	JRA-9670	Integrity Checker fails when correcting the default dashboard		Resolved	Fixed
	JRA-9666	When an issue type is not specified the CSV importer grabs the first it sees, but this may not be valid because of issue types per project		Resolved	Fixed
	JRA-9643	Update docs to reflect the renaming of 'Developer Workload'		Resolved	Fixed
	JRA-9640	Add to OS section to Requirments page		Resolved	Fixed
	JRA-9613	Found typo in a file		Resolved	Fixed
	JRA-9586	Bugzilla doesn't import priorities		Resolved	Fixed
	JRA-9582	UpgradeTask 132 fails if "Assign Issue Screen" does not have a description		Resolved	Fixed
	JRA-9578	initiating user detail not properly shown in usercreated email		Resolved	Fixed
	JRA-9575	Permission-Bug in Manage Watcher List		Resolved	Fixed
	JRA-9562	@123,456@ in Wiki renderer results in a text field		Resolved	Fixed
	JRA-9548	Custom field values for multi selects not returned in SOAP		Resolved	Fixed

	JRA-9544	file "\includes\snippets\listbyname.jsp" is missing taglib identifier		Resolved	Fixed
	JRA-9505	Delete component displays erratic message		Resolved	Fixed
	JRA-9476	Workflow cache not updated on project creation.		Resolved	Fixed
	JRA-9465	Update global permissions of the documentation to include teh Bulk Change permission		Resolved	Fixed
	JRA-9452	Last two builds JIRA won't allow setting an issue to unassigned as a Post function in workflow transitions		Resolved	Fixed
	JRA-9435	Add to Upgrade Guide advice that IssueEvent changelog can now be null		Resolved	Fixed
	JRA-9430	Add documentation that specifys that text emails for subscriptions do not behave the same way as html emails		Resolved	Fixed
	JRA-9399	Bulk Move - New security level field values are shown from originating project, not the target project		Resolved	Fixed
	JRA-9394	Security Bug: Deletion of a comment made to jira-developers is visible to all jira-users when viewing change history		Resolved	Fixed
	JRA-9285	Project portlet doesnt remember the [hide] [show>>] settings upon refresh		Resolved	Fixed

JIRA 3.5.3 Upgrade Guide

JIRA 3.5.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.3 from JIRA 3.5.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.5.3 from JIRA 3.5.2.

JIRA 3.5.2 Release Notes

JIRA 3.5.2 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.5.2 in Standard, Professional and Enterprise editions. This point release includes over 20 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.5.2 Upgrade Guide.

JIRA 3.5.2 includes over 20 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11790' does not exist for the field 'fixVersion'.

JIRA 3.5.2 Upgrade Guide

JIRA 3.5.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.2 from JIRA 3.5.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Issue Event Changelog Can Now Be Null

- If you have implemented a custom JIRA Issue Event Listener you need to be aware of the following API change.

In JIRA 3.5.2, the IssueEvent object thrown as a result of an edit operation, **may now return null** from a `getChangeLog()` call. The case where this happens is when a user chooses to edit an issue but only leaves a comment and makes no other changes to the issue. Prior to 3.5.2 no event was fired in this case and this was identified as a bug (JRA-9415) and has since been fixed. Check any calls to getChangeLog() for null.

JIRA 3.5.1 Release Notes

JIRA 3.5.1 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Atlassian Software Systems is proud to announce the release of JIRA 3.5.1 in Standard, Professional and Enterprise editions. This point release includes over 20 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version please read through the JIRA 3.5.1 Upgrade Guide.

JIRA 3.5.1 includes over 20 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11720' does not exist for the field 'fixVersion'.

JIRA 3.5.1 Upgrade Guide

JIRA 3.5.1 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.5.1 from JIRA 3.5. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here. When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.5 to JIRA 3.5.1.

JIRA 3.4 and 3.4.1 Release Notes

JIRA 3.4 Release Notes

Atlassian Software Systems, Australia's fastest growing software company, is proud to announce the latest release of *the* issue tracking and project management application - **JIRA 3.4** (download it here). Just over 3 weeks since the final 3.3.x release, **JIRA 3.4** includes some **50** powerful new features and improvements along with over **70** bug fixes.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Upgrade Information

In order to complete a successful upgrade, please refer to our Upgrade Guides. If you are upgrading from JIRA 3.3.3 please refer to the following document:
http://confluence.atlassian.com/display/JIRA/JIRA+3.4+and+3.4.1+Upgrade+Guide

If you are upgrading from a pre-3.3.3 release please read the following guides also:
http://confluence.atlassian.com/display/JIRA/All+JIRA+3.x+Upgrade+Guides

Features and Improvements:

- Issue Types Per Project
- Renderers
- Clone Portlets

- Issue Operation Plugin
- RSS Improvements
- RPC Searching
- Dynamic Header Banner
- Internationalisation
- Bulk Migration by Issue Type
- Change Parent Issue of Sub-Task
- Multi-User Custom Field
- Trackback Configuration

Contents

1. New Features
2. Improvements
3. Bug Fixes
4. Notes

New Features

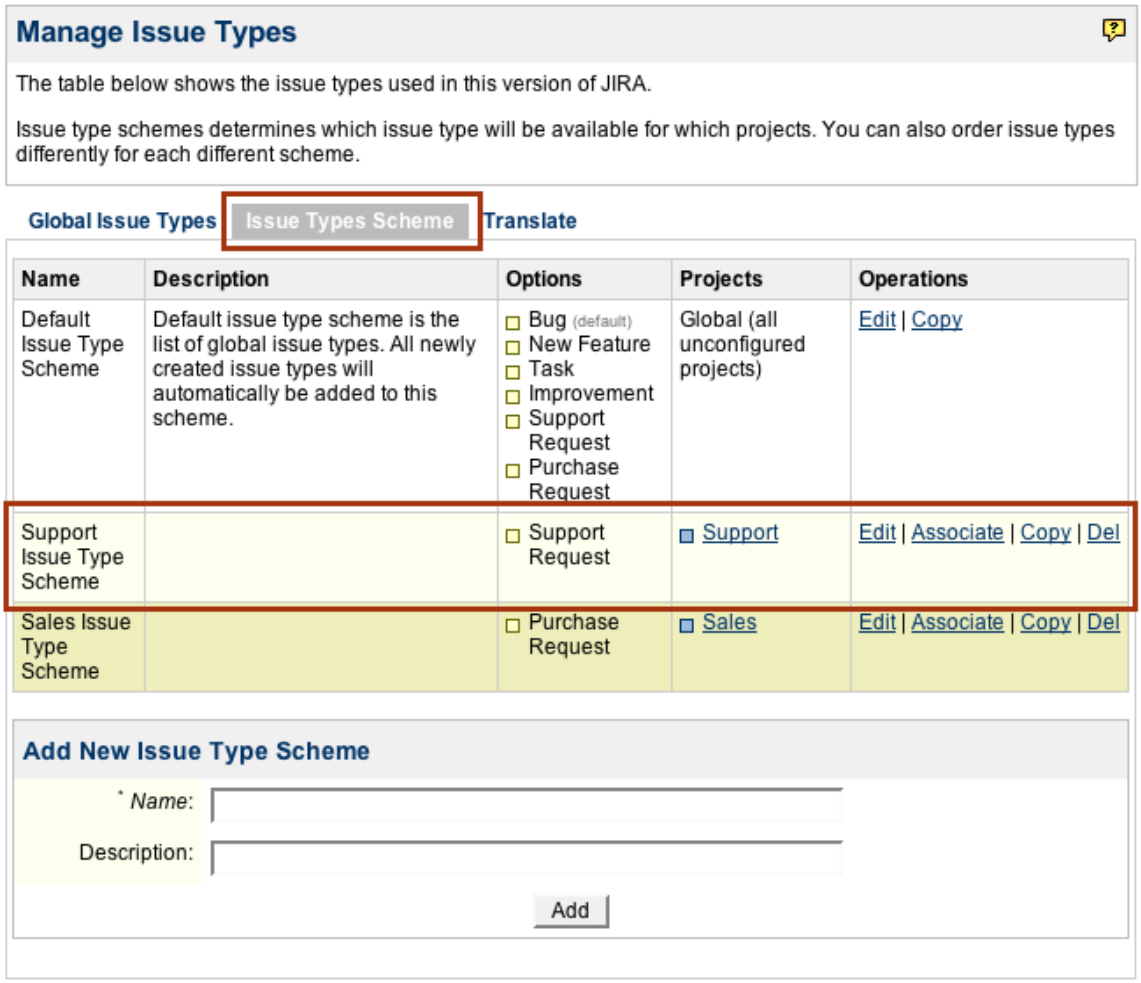
This release includes over 50 new features and improvements - see the full list of features & improvements online

Issue Types Per Project

The **most highly requested** feature in JIRA - with over **200** votes - you can now configure **Issue Types Per Project**. Issue types can be defined for a specific project - adding contextual relevance to issue type selection when creating/editing/moving an issue.

This feature provides the ability to better define the context of a project with only relevant issue types available - resulting in a more intuitive user interface, and enables greater control over the users' issue selection process.

For example, the **Sales** project may require a **Purchase Request** issue type, while this issue type would not apply to the context of the **Support** project. Further, the issue type **Support Request** is relevant within the **Support** project but would not apply to the context of the **Sales** project.



Renderers

This release also introduces **Renderers** within JIRA text-based fields such as description and comments - allowing a greater range of expression within these fields.

This release ships with two renderers, the default text renderer, which preserves the functionality available in previous JIRA releases, and the Atlassian wiki renderer, which brings the power of the Confluence wiki engine to JIRA. In the Enterprise edition of JIRA, renderers can be configured on a per field, per project/issue type level, allowing a flexible combination of text and wiki markup. In the Standard and Professional editions renderers can be configured on a per field basis. Renderers are implemented as pure JIRA plugins, meaning that any renderer can be easily added to or removed from use within JIRA, including any custom renderers that may be developed.

Further details on the renderer functionality is available in the documentation:

http://www.atlassian.com/software/jira/docs/v3.4/renderers_overview.html

Description

Header 1

Header 4

```
public String exampleCode(String test)
{
   return "this is an example";
}
```

inline attachments

green blue red

TST-31

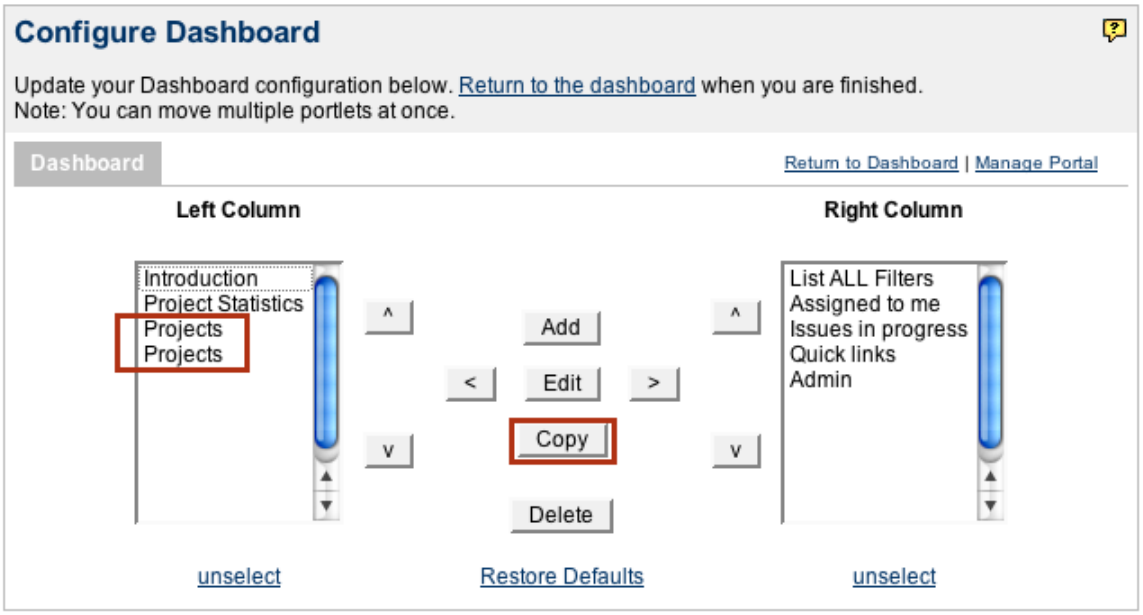
link to google

1. item 1
2. item 2

Day	Working Hours
Mon (Day 1)	8 - 5
Tue (Day 2)	8 - 5
Wed (Day 3)	8 - 5
Thu (Day 4)	8 - 5
Fri (Day 5)	8 - 5

Clone Portlets

Dashboard administration is simplified with the ability to clone a portlet.

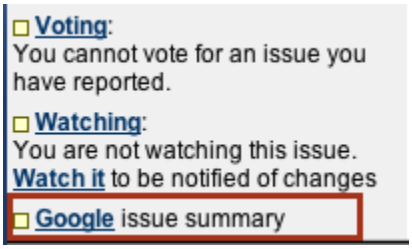


Issue Operation Plugin

This new system plugin allows new issue operations (links and HTML) to be added to the operations panel of the view issue screen. With a simple configuration process, it is possible to add quick access to commonly used links to every issue.

For example, the following operation will execute a **Google** search on the issue summary.

Information about how to create your own Issue Operations can be found in the docs online.

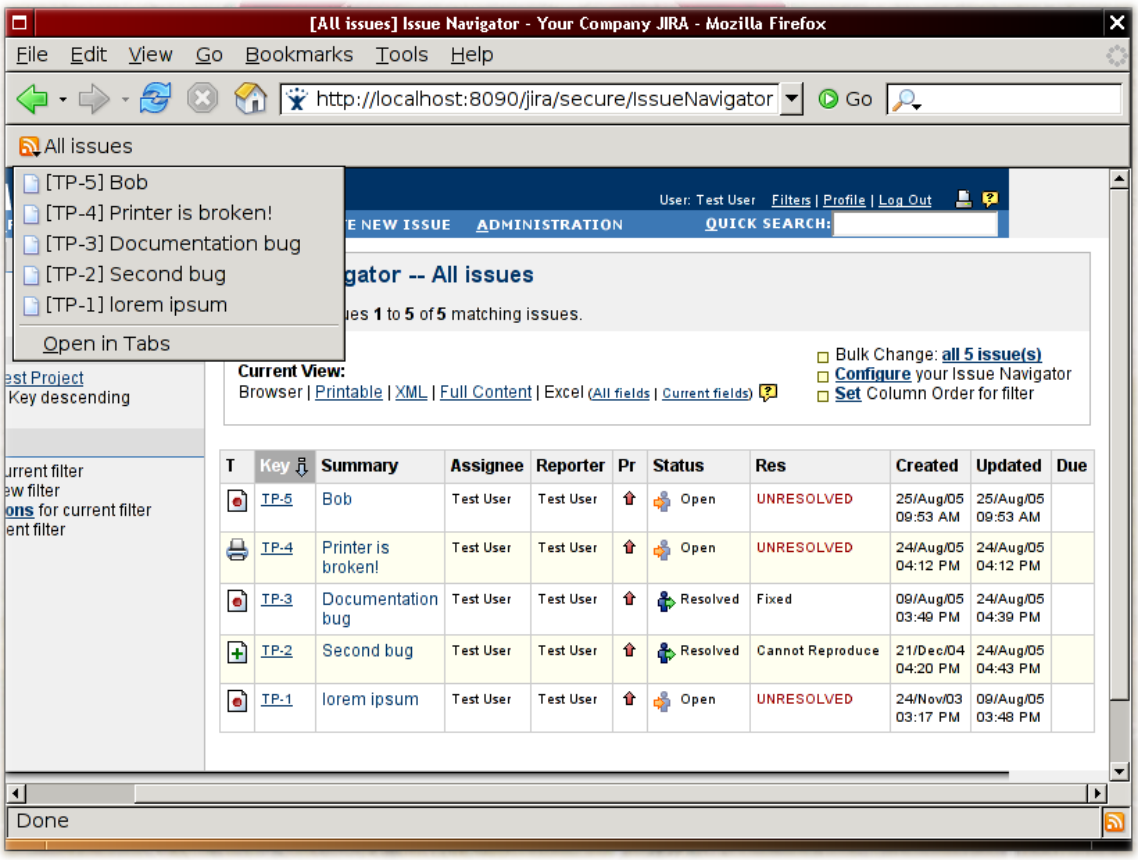


Improvements

RSS Support Improvements

JIRA has been improved to take advantage of browser support for RSS feeds. A result of any search can be accessed quickly and directly by adding 'live bookmarks'.

Basic authentication support for RSS feeds from JIRA has also been added in this release - allowing the option of avoiding the transmission of usernames and passwords across the wire in clear text format.



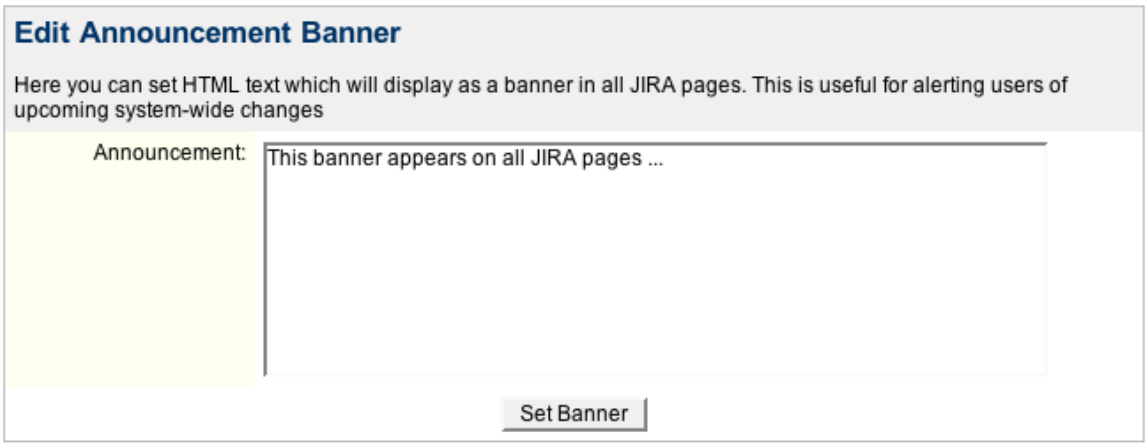
RPC Searching

The RPC searching functionality has been improved - results are returned quicker and more efficiently with fewer database calls.

Dynamic Announcement Banner

With thanks to Nick Minutello, the announcement banner can now be dynamically configured through the JIRA UI.

The announcement banner is displayed on all JIRA pages and is useful for alerting users to important information - for example, scheduled server maintenance, approaching project deadline, etc. This information can be updated immediately without having to restart JIRA.

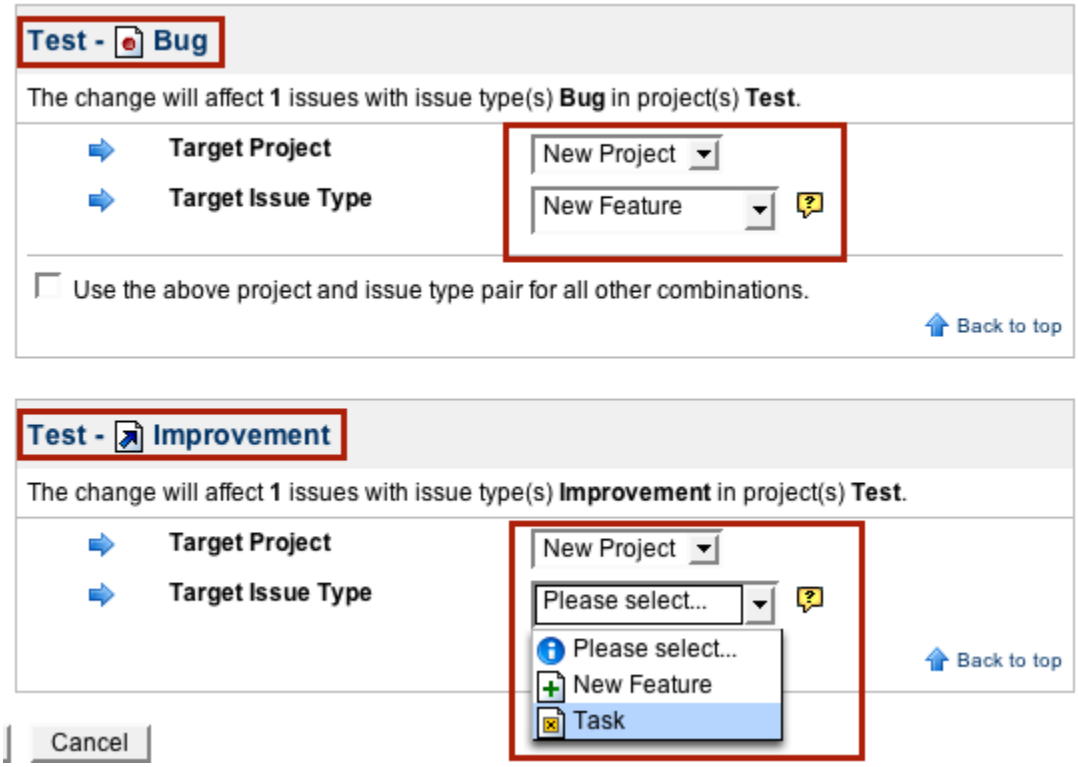


Internationalisation

With the introduction of the **Czech** language pack, JIRA is now available in **16** different languages. This release also includes an updated **Traditional Chinese** language pack. Once again, the Atlassian team would like to thank our users who have contributed to the translation process - this work is much appreciated.

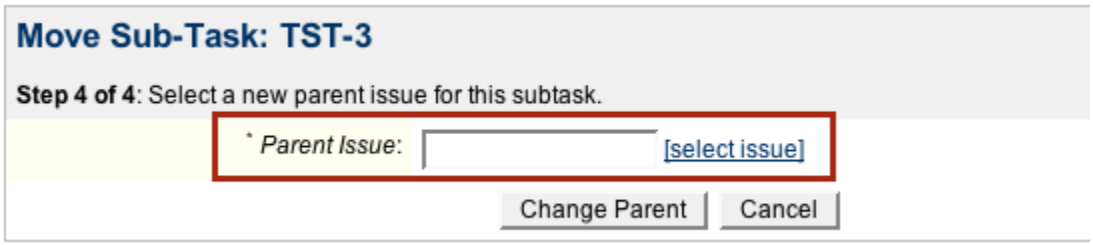
Bulk Migration by Issue Type

Previously, it was only possible to bulk migrate a collection of issues to a single project and issue type. Now, different issue types can be migrated to different projects and issue types - allowing you to specify exactly how each issue should be moved. With this fine-grained configuration capability, bulk migration operations are now easier and quicker.



Change Parent of Sub-Task

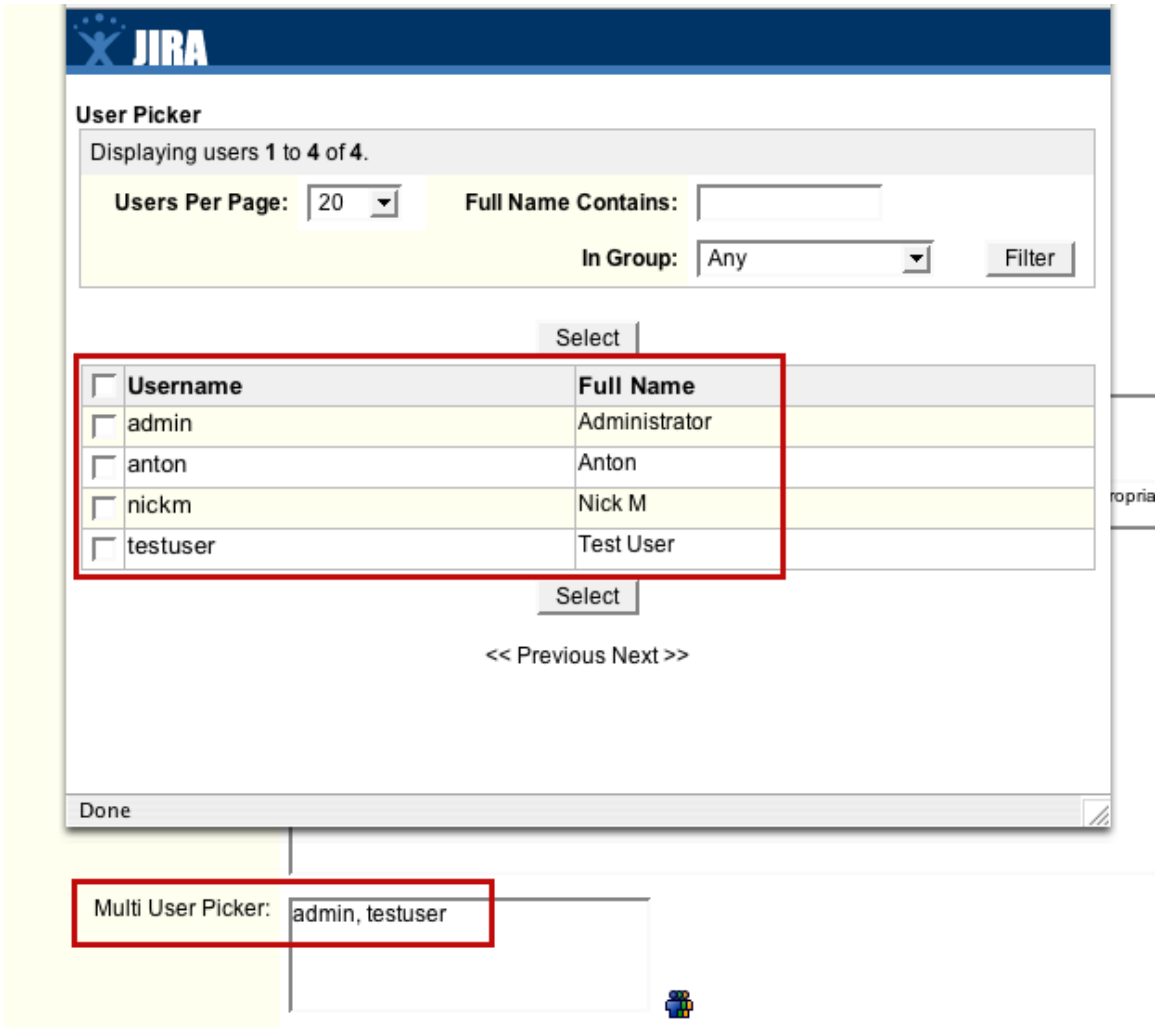
The parent issue of a sub-task can now be changed - allowing a sub-task to be 'moved' from one parent issue to another.



Multi-user Custom Field

It is now possible to select multiple users through the multi-user custom field.

This improvement has also been extended to the configuration of security level, permission and notification schemes. This means that by selecting users for the Multi-user field it is possible to control who will have access to the an issue and who will be notified when the issue is updated. Please note that using Multi-user custom field for permissions and security levels is only supported by the Enterprise Edition.



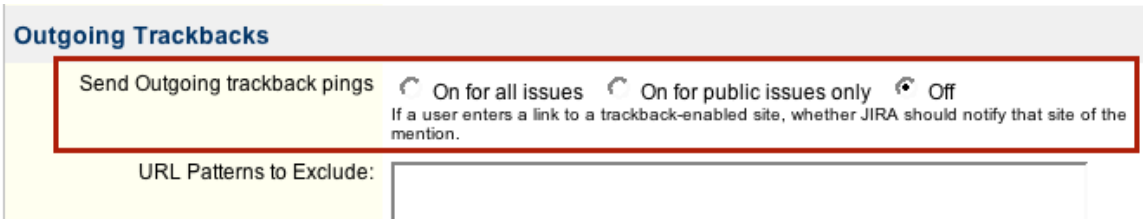
Trackback Configuration

Trackback configuration has been extended with three possible modes for **Outgoing Trackbacks**:

- **Off for All Issues**
- **On for Public Issues Only**
- **On for All Issues**

By using the **On for Public Issues Only** setting, trackbacks from issues protected by a security or permission scheme (i.e. issues not visible by non-logged in users) will not be sent - the external sites referenced in these secure issues will not be notified of the referral.

The operation of **Incoming Trackbacks** has also been modified such that 'secure' issues can receive trackback notifications. The issues will remain secure based on their security settings, but users who have access to the issues will be able to see the trackback references that issues have received.



Notable Features & Improvements

Some other notable improvements included in this release:

- New Jelly tag allowing a field (including custom field) to be placed onto a Screen
- Bugzilla importer preserves inter-issue dependencies

- Bugzilla importer converts Bugzilla text links to JIRA links
- Search user custom fields using group membership

Bug Fixes

This release includes over 70 bug fixes - the full list of bug fixes online

Notes

Javascript Caching

In some instances, the browser may read from a cached version of the Javascript files used throughout JIRA. This will result in certain elements appearing incorrectly - e.g. the issue type selection list may appear with repeated icons. This problem can be corrected by forcing a reload on the page in question - i.e. pressing 'CTRL-R' or 'F5'.

JIRA 3.4.1 Release Notes

JIRA 3.4.1 is released today in Standard, Professional and Enterprise editions. This point release fixes JIRA 3.4's incompatibility with MySQL and MS SQL Server and includes bug fixes and improvements which appear below.

If upgrading from JIRA 3.3.3 or 3.4 please read through JIRA 3.4 and 3.4.1 Upgrade Guide. If upgrading from an earlier version of JIRA please read through all the Upgrade Guides for all versions that you are skipping during the upgrade.

JIRA 3.4.1 bug fixes and improvements.

jiraissues: Unable to determine if sort should be enabled

JIRA 3.4 and 3.4.1 Upgrade Guide

JIRA 3.4 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4 from JIRA 3.3.3. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Two major new features of JIRA 3.4, wiki renderer previews, and issue types per project require that javascript be enabled to make use of their full functionality. You will still be able to use all the core features of JIRA with javascript disabled.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

1. Please do **not** copy `jira-application.properties` file from your old JIRA installation. Edit the file that is shipped with JIRA 3.4 and make needed changes. New properties have been added to this file so if you simply copy the old file across the following error would occur JRA-8645.
2. If you have written any CustomFieldType or CustomFieldSearcher plugins please refer to Upgrading Custom Field Types in JIRA 3.4
3. The default user preferences are now configured in the jira-application.properties file and are configurable through the admin section of JIRA. Any properties in the old file preferences-default.xml will no longer effect JIRA configuration.
4. Please note that to configure issue types per project you must have JavaScript turned on in your web browser.
5. If you are using MySQL please do not use Connector/J 3.1.11 JDBC Driver as it has the following bug. Connector/J 3.1.10 and earlier work fine.

JIRA 3.4.1 Upgrade Guide

This section contains specific information you need to know when upgrading to JIRA 3.4.1 from JIRA 3.4. If upgrading from JIRA 3.3.3 please read the previous section as well. If upgrading from an older version than JIRA 3.3.3, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

1. Please do **not** copy `jira-application.properties` file from your old JIRA installation. Edit the file that is shipped with JIRA 3.4 and make needed changes. New properties have been added to this file so if you simply copy the old file across the following error would occur JRA-8645.
2. If you have written a CustomFieldType that implements the `com.atlassian.jira.issue.customfields.CustomFieldType` interface directly rather than extending one of the Abstract classes that ship with JIRA please read Upgrading Custom Field Types in JIRA 3.4.1.
3. If you have written an Custom Field Searcher please have a look at Upgrading Custom Field Types in JIRA 3.4.1.
4. JIRA 3.4 and 3.4.1 do not generate an Issue Assigned event. The Issue Updated event is generated instead. In previous versions of JIRA the Issue Assigned event was generated when issues are assigned using the "Assign" operation on the View Issue page. This means

that even when the "Assign" operation is used JIRA will send notifications to parties listed under the Issue Updated event. The patch to correct this behaviour is available at JRA-8533.

Upgrading Custom Field Types in JIRA 3.4

Unknown macro: {version-warn}

JIRA 3.4 and higher

Changes to the custom field types code

For JIRA 3.4, there has been further upgrades to the custom fields code. If you have written your own Custom Field Types of Custom Field Searchers, you'll want to read this document. We recommend that you ensure that your custom field types compile against the latest 3.4 build and verify this under a testing environment before putting it into production.

You can download the latest JIRA Development Kit from its confluence space.

CustomFieldType Interface changes CustomFieldConfig and CustomFieldConfigItemType changed to FieldConfig and FieldConfigItemType

Note that the references to `CustomFieldConfig` and `CustomFieldConfigItemType` has become `FieldConfig` and `FieldConfigItemType` respectively. This change is to bring CustomFields and SystemFields closer so that, in the future, fields can be configured in uniform way.

```
com.atlassian.jira.issue.customfields.config.CustomFieldConfig;
com.atlassian.jira.issue.customfields.config.CustomFieldConfigItemType;
```

has become

```
com.atlassian.jira.issue.fields.config.FieldConfig;
com.atlassian.jira.issue.fields.config.FieldConfigItemType;
```

A straight replacement should do the trick.

isRenderable()

With the addition of renderers in JIRA 3.4 there has been a minor addition to the CustomFieldType interface. There is a now a method of the signature:

```
public boolean isRenderable();
```

There is a default implementation of this method in the AbstractCustomFieldType class that returns false. If you are extending any of the existing base classes, AbstractCustomFieldType, TextCFType, SelectCFType, then there are no changes needed to upgrade your CustomField Types.

If you are implementing the CustomFieldType interface then you will need to implement the isRenderable method in your class. To retain the functionallity that you always had the method should return false. With the introduction of renderable fields in JIRA 3.4 if you return true in the isRenderable method then you will be able to assign the custom field a renderer. This only really makes sense for text fields where you can use the power of the wiki syntax.

getChangelogString()

To better support logging of change history from custom fields there is now a method of the signature:

```
public String getChangelogString(CustomField field, Object value);
```

This method allows a custom field type to provide a string that can give a custom fields value meaning. The two examples that ship with JIRA are the ProjectPicker and VersionPicker custom fields. Both of these fields store the database id of the value (e.g. My Project is id 10000, so it stores 10000). The ProjectCFType object will return the value 'My Project' from this new method so that we can register in the change item bean that the string is 'My Project' and that the value is '10000'. The AbstractCustomFieldType object returns null for this method by default. If you are extending any of the existing base classes, AbstractCustomFieldType, TextCFType, SelectCFType, then there are no changes needed to upgrade your CustomField Types.

Upgrading Custom Field Types in JIRA 3.4.1

Upgrading Custom Field Types in JIRA 3.4.1 valuesEquals method addition

If you have implemented a CustomFieldType by directly implementing the `com.atlassian.jira.issue.customfields.CustomFieldType` interface instead of extending one of the Abstract classes that ship with

JIRA, you will need to add the following method to your implementation:

```
/**
 * Used to compare old field value to the new field value when the issue is being updated
 * and work out whether a change item should be generated.
 * @param v1 current value
 * @param v2 new value
 * @return true if the change item should be generated, false otherwise
 */
boolean valuesEqual(Object v1, Object v2);
```

This means that you will need to implement this method and recompile your class before deploying it into JIRA 3.4.1.

The method has been introduced to resolve bug JRA-8480 and allows the CustomFieldType to have control over change items that JIRA generates when an issue is updated.

The `com.atlassian.jira.issue.customfields.impl.AbstractCustomFieldType` implements the method as follows:

```
if (v1 == v2)
{
    return true;
}

if (v1 == null || v2 == null)
{
    return false;
}

return v1.equals(v2);
```

which is compatible with the behaviour of previous JIRA releases. Therefore if your CustomFieldType extends AbstractCustomFieldType, or any of its subclasses, you do not need to add this method, unless you would like to take advantage of it.

CustomFieldSearchers - AbstractCustomFieldSearcher constructor change

If you have implemented any CustomFieldSearchers that extend the com.atlassian.jira.issue.customfields.searchers.AbstractCustomFieldSearcher class then you will have to modify your CustomFieldSearcher. The constructor of the AbstractCustomFieldSearcher class has changed from being a no-arg constructor to taking an instance of com.atlassian.jira.web.bean.FieldVisibilityBean. You must pass the FieldVisibilityBean to the super classes constructor. You will need to include code that will look something like this:

```
public MultiProjectSearcher(FieldVisibilityBean fieldVisibilityBean)
{
   super(fieldVisibilityBean);
}
```

JIRA 3.4.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.4.3 Release Notes

In the tradition of worthwhile updates, JIRA 3.4.3 is released today in Standard, Professional and Enterprise editions. This point release includes over 40 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version than JIRA 3.4.2 please read through the Upgrade Guides for all versions that your are skipping during the upgrade. If upgrading from JIRA 3.4.2 please read the 3.4.3 Upgrade Guide before continuing.

JIRA 3.4.3 includes over 40 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11581' does not exist for the field 'fixVersion'.

JIRA 3.4.3 Upgrade Guide

JIRA 3.4.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4.3 from JIRA 3.4.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.4.2 to JIRA 3.4.3.

JIRA 3.4.2 Release Notes

JIRA 3.4.2 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

In the tradition of worthwhile updates, JIRA 3.4.2 is released today in Standard, Professional and Enterprise editions. This point release includes over 35 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version than JIRA 3.4.1 please read through the Upgrade Guides for all versions that your are skipping during the upgrade. If upgrading from JIRA 3.4.1 please read the 3.4.2 Upgrade Guide before continuing.

JIRA 3.4.2 includes over 35 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11542' does not exist for the field 'fixVersion'.

JIRA 3.4.2 Upgrade Guide

JIRA 3.4.2 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.4.2 from JIRA 3.4.1. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.
When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

- There are no specific instructions you need to be aware of related to upgrading from JIRA 3.4.1 to JIRA 3.4.2.

JIRA 3.4.1 Release Notes

JIRA 3.4.1 Release Notes

JIRA 3.4.1 is released today in Standard, Professional and Enterprise editions. This point release fixes JIRA 3.4's incompatibility with MySQL and MS SQL Server and includes bug fixes and improvements which appear below.

If upgrading from JIRA 3.3.3 or 3.4 please read through JIRA 3.4 and 3.4.1 Upgrade Guide. If upgrading from an earlier version of JIRA please read through all the Upgrade Guides for all versions that you are skipping during the upgrade.

JIRA 3.4.1 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11540' does not exist for the field 'fixVersion'.
jiraissues: Unable to retrieve issue data

JIRA 3.3 Release Notes

JIRA 3.3 Release Notes

Atlassian Software Systems is proud to announce the latest release of *the* issue tracking and project management application - **JIRA 3.3** - download it here. Only 10 weeks since the last release, **JIRA 3.3** includes some of the most requested features along with a host of bug fixes and improvements.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Upgrade Information

In order to complete a successful upgrade, please refer to the following guides:

- JIRA 3.2 Upgrade Guide
- JIRA 3.3 Upgrade Guide

It is necessary to follow both guides if upgrading from a **pre 3.2.x** version. It is only necessary to follow the JIRA 3.3 Upgrade Guide when upgrading from a **3.2.x** version.

Features and Improvements:

- Multiple Project Filters
- Bulk Move
- User Custom Field Notification Target
- Extended Search Capabilities
- SOAP Enhancements
- Performance Improvements
- New Translations

Contents

1. New Features
2. Improvements
3. Bug Fixes

New Features

This release of JIRA includes some of the most requested features as logged at http://jira.atlassian.com - the full list can be viewed at JIRA 3.3 New Features & Improvements.

Multiple Project Filters

The Issue Navigator has been completely overhauled to pave the way for future extended search functionality and enhancements. Addressing one of the most popular feature requests (with over 50 votes), this release includes the ability to execute a **search across multiple projects**.

This feature increases the searching capabilities available to the user with the ability to aggregate a collection of issues from across multiple projects. Users can now design a search query that directly addresses their requirements for searching throughout JIRA.

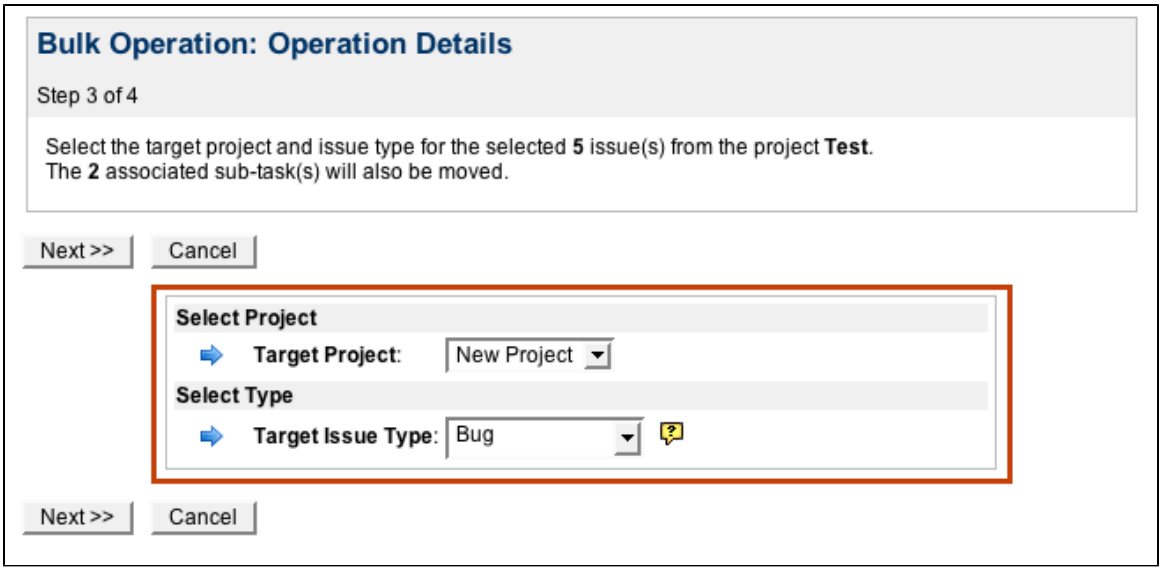
The applications of this improved search capability are countless ... for example:

- Retrieve all issues opened in the last week from selected projects - add these results to an RSS feed or view them on the JIRA dashboard
- Add a statistics portlet to the JIRA dashboard with results from a multi-project filter
- Subscribe to a multi-project filter containing issues assigned to your team members - with issues retrieved only from the projects you manage!multiplesearch.png|align=center,border=1!

Bulk Move

Extending the range of bulk operations available in JIRA, it is now possible to move multiple issues at once.

The Bulk Move operation allows a collection of issues (from multiple projects and consisting of multiple issue types) to be moved to another project and/or issue type. The ability to update multiple issues in this manner gives the user even more power - for instance, it is now possible to merge issues from multiple projects into one project.



User Custom Field as Notification Target

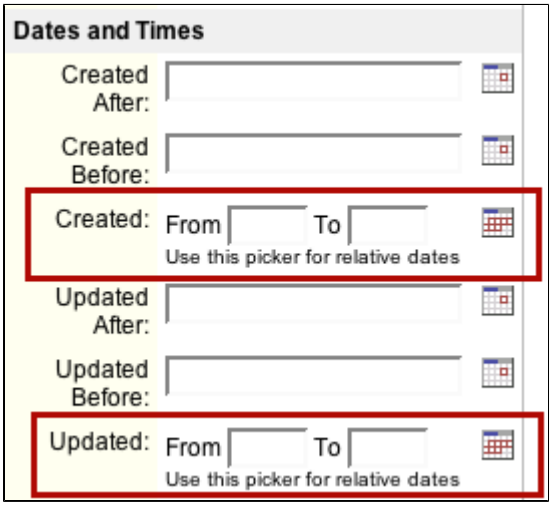
It is now possible to specify an additional notification target by selecting a user in a '**CC**' custom field. This provides the issue creator/editor the ability to add another user to the notification recipient list for a specific issue. Many thanks to Chris Wood at MetOcean Engineers for his work on this feature!

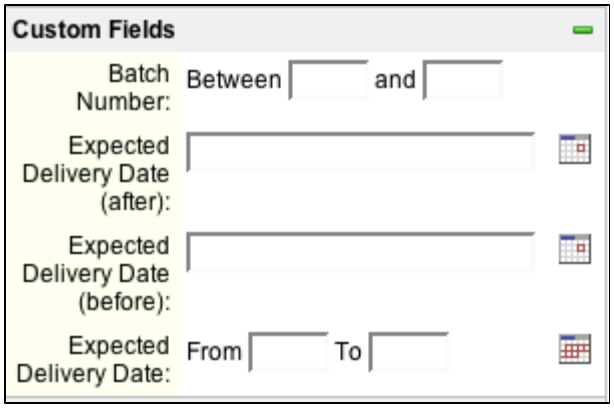
Improvements

This release of JIRA includes some significant improvements in available functionality and overall system performance.

Extended Search Capabilities

More precise search filters can be created by specifying a date range in relation to the system fields '**Created**' and '**Updated**' and the custom field '**Date Time**'. A number range can also be specified for the '**Number**' custom field. The ability to specify a range in this manner provides the user with a very fine-grained filter over a specific set of issues.





SOAP Enhancements

JIRA's SOAP capabilities are growing more complete with each release. For JIRA 3.3, you can now progress an issue through a workflow remotely; find out what actions are available for an issue, update the fields and progress through the workflow. It's now easier than ever to seamlessly integrate JIRA with external tools.

Extended XML-RPC Functionality

Bringing the XML-RPC functionality inline with the current SOAP plugin, it is now possible to update an issue & run a search request.

Performance Improvements

Continually striving to advance system efficiency, this release includes a number of notable performance improvements:

- **Issue Navigator** performance has been improved with fewer database access calls - displaying filter results quicker
- **Workflow Activation** is now less memory intensive - the operation completes without pulling all issues into memory to make the new workflow association
- **Version Management** operations have been streamlined to complete quicker
- The **Link Issue** pop-up displays quicker
- The **Permission Schemes** and **Notification Schemes** management pages have been refined to return scheme information quicker
- **Date Range** searches are executed more efficiently

New Translations

Along with updates to the **Traditional Chinese** and **German** translations, JIRA is now available to use in **Italian** and **Slovakian**. Once again, we would like to thank all those who have contributed to the translation process - JIRA is now available in **14** different languages.

JIRA Standalone

JIRA Standalone now ships with Tomcat 5.5 - allowing users to use JIRA out of the box with the latest version of the popular application server.

> **Previous Standalone Installations**
>
> Due to incompatibilities in the format of the **server.xml** file between versions of Tomcat - please **do not copy** the **server.xml** file from previous standalone installations to the new standalone installation.

Bug Fixes

This release includes over 90 bug fixes - the best way to see them is them is to ask JIRA - Issues Resolved for JIRA 3.3.

JIRA 3.3 Upgrade Guide

JIRA 3.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.3 from JIRA 3.2.x. If upgrading from an older version of JIRA, please go to the complete list of Upgrade Guides, and read the notes for each version you are skipping during the upgrade.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

Known incompatibilities

3.3.x is not a good release for IBM shops:

1. JIRA 3.3.x may not work on **Websphere 5.0.x and 5.1.x** due to JRA-7699
2. When using DB2, JIRA may hang when deleting projects or performing workflow operations. See the full problem description (and possible workaround) in the documentation

Websphere or DB2 users, please stick with 3.2.x or move on to 3.4.x or higher, where these problems have been resolved.

Notes on upgrading

1. Due to web browser caches, changes to JIRA's Issue Navigator might appear corrupted or unstyled. Please refresh your browser's cache (press *Shift+Reload* on the Find Issue's page) for the changes to appear correctly.
2. JIRA's **issue cache** size will be automatically set to 0 during the upgrade, as it is no longer needed due to performance improvements in JIRA (JRA-7166)
3. If you have written any CustomFieldType or CustomFieldSearcher plugins please refer to this document
4. Users with **outgoing** trackback pings enabled (not the default) may wish to disable this until JRA-7589 is fixed, to avoid the risk of the mail queue hanging.
5. If you have bookmarks or deal with hard coded links to the issue navigator, you should read about the changed issue navigator parameters
6. If you are **using JIRA Standalone**, please do *not* simply copy your old `conf/server.xml` file to the new installation of JIRA. Please read this document.
7. If upgrading JIRA in an external Tomcat installation, be sure to delete the `work/` temporary directory before restarting JIRA, to clear cached JSPs from the old JIRA.

Parameter changes in Issue Navigator

We've made significant backend changes to the issue navigator in 3.3. This resulted in some parameters being changed and are deprecated.

What does this affect?

This affects only direct links to the issue navigator that's been saved outside JIRA. e.g. a bookmark to an RSS feed, a Confluence page with the search parameters hard coded. This will not affect saved filters in JIRA, or portlets shipped with JIRA.

What has changed?

For 3.3 parameters that have changed are:

1. resolutionIds -> resolution
2. priorityIds -> priority
3. statusIds -> status
4. createBefore -> create:before
5. createAfter -> create:after
6. createPrevious -> create:previous
7. updateBefore -> update:before
8. updateAfter -> update:after
9. updatePrevious -> update:previous
10. duedateBefore -> duedate:before
11. duedateAfter -> duedate:after
12. duedatePrevious -> duedate:previous
13. duedateNext -> duedate:next

Also the values for the createNext and updateNext parameters has been modified.

What this means for me? What do I need to do?

We've put in place mechanisms that makes the issue navigator backwards compatible, so you won't actually notice any difference using links with deprecated parameters. However, it's **strongly recommended** that you re-bookmark any affected links to JIRA. We can't guarantee that this will be in place forever and it's better if you update it as a soon as possible.

For system administrators, all searches using the deprecated parameters will be logged to the server with the client's URL and IP address. You should try to chase up the user so that there's no major problems down the track.

If you find these warnings impossibly annoying, you can update your `log4j.properties` to disable them by adding the line below to your log4j.properties. However, we **strongly discourage** you from doing this. The warnings are there so that they can be identified and stop any problems further down the track.

```
log4j.category.com.atlassian.jira.util.retro = ERROR, console
```

Upgrading custom CustomFieldTypes in JIRA 3.3

Applies to JIRA 3.3 and higher

Changes to the custom field types code

For JIRA 3.3, there has been further upgrades to the custom fields code. If you have written your own Custom Field Types of Custom Field Searchers, you'll want to read this document. We recommend that you ensure that your custom field types compile against the latest 3.3 build and verify this under a testing environment before putting it into production.

You can download the latest JIRA Development Kit from its confluence space.

CustomFieldType Interface changes

We removed all references to GenericValue objects in the CustomFieldType and replaced it with Issue. The issue object provides powerful accessors to the issue's parameters such as affects versions and components. If you still need the old GenericValue object, you can call `getGenericValue` on the issue object.

We've also added a new method to the interface.

```
    /**
     * Returns a list of indexers that will be used for the field. This will over-ride the anonymous
searcher specified
     * by {@link AbstractCustomFieldSearcher#getRelatedIndexers()} and {@link
AbstractCustomFieldSearcher#index(Document,
     * CustomField, Object)}
     *
     * @return List of instantiated and initialised {@link FieldIndexer} objects. Null if no related
indexers.
     */
    List getRelatedIndexers(CustomField customField);
```

Through this method you can over-ride how this custom field will get indexed.

CustomFieldSearcher Interface over-haul

The CustomFieldSearcher interface now extends the new IssueSearcher interface. This allows you to write more flexible searchers as well as easily extend and reuse code from the default system searchers.

If you haven't developed your own custom searchers, then there's nothing you need to do. If you do have custom searchers but they extend AbstractCustomFieldSearcher then you shouldn't have to make any changes to the Java code as we have tried to keep this class backwards compatible. However, you should still recompile to make sure.

You will need to update all velocity pages used by the searcher. For the edit pages you'll need to use the new header (rather than the controlHeaders)

```
#searcherEditHeader ($customField.id $customField.name)

...

#searcherEditFooter ($customField.id $customField.description)
```

and for the view templates:

```
#searcherHeader ($customField)

...

#searcherFooter ($customField)
```

This change is in order to give the search templates greater flexibility (i.e. allow different rendering behaviour of the searcher from the standard

edit screens.

Upgrading to JIRA 3.3 Standalone

Upgrading to JIRA 3.3 Standalone

JIRA Standalone now ships with Tomcat 5.5 - allowing users to use JIRA out of the box with the latest version of the popular application server.

Previous version of JIRA shipped with an older version of Tomcat, for example, JIRA 3.2.x shipped with Tomcat 4.1.29. The configuration files for Tomcat 5.5 are **not compatible** with files from older Tomcat releases.

This means that you cannot simply copy the `conf/server.xml` from your old JIRA Standalone installation. You will need to use the `conf/server.xml` shipped with JIRA 3.3 Standalone as a starting point, and then customise it for your database as described in our documentation.

JIRA 3.3.3 Release Notes

JIRA 3.3.3 Release Notes

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

In the tradition of worthwhile updates, JIRA 3.3.3 is released today in Standard, Professional and Enterprise editions. This point release includes over 70 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version than JIRA 3.3.2 please read through the Upgrade Guides for all versions that your are skipping during the upgrade. If upgrading from JIRA 3.3.2 please read the 3.3.3 Upgrade Guide before continuing.

JIRA 3.3.3 includes over 70 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11441' does not exist for the field 'fixVersion'.

JIRA 3.3.3 Upgrade Guide

JIRA 3.3.3 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.3.3 from JIRA 3.3.2. If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below.

1. the release notes templates, releasenotes-html.vm and releasenotes-text.vm are no longer provided with an Issue GenericValue, they are now provided an Issue object. If you have customized these templates you will need to modify your velocity syntax to access the Object methods instead of accessing the values via
   the GenericValue.

JIRA 3.3.2 Release Notes

JIRA 3.3.2 Release Notes

In the tradition of worthwhile updates, JIRA 3.3.2 is released today in Standard, Professional and Enterprise editions. This point release includes over 40 bug fixes and improvements. It can be downloaded here.

If upgrading from an earlier version than JIRA 3.3.1 please read through the Upgrade Guides for all versions that your are skipping during the upgrade. If upgrading from JIRA 3.3.1 there is no specific Upgrade Guide for JIRA 3.3.2.

JIRA 3.3.2 includes over 40 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11394' does not exist for the field 'fixVersion'.

JIRA 3.3.1 Release Notes

JIRA 3.3.1 Release Notes

In the tradition of worthwhile updates, JIRA 3.3.1 is released today in Standard, Professional and Enterprise editions. This point release includes over 30 bug fixes and improvements. It can be downloaded here. See the JIRA 3.3 Upgrade Guide and JIRA 3.3.1 Upgrade Guide before upgrading.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.3.1 includes over 30 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11360' does not exist for the field 'fixVersion'.

JIRA 3.3.1 Upgrade Guide

JIRA 3.3.1 Upgrade Guide

This page contains specific information you need to know when upgrading to JIRA 3.3.1 from JIRA 3.3.

If upgrading from an older version of JIRA, please read the *Upgrade Guide* for each version your are skipping during the upgrade. The complete list of Upgrade Guides is available here.

When upgrading JIRA please follow the general upgrade instructions keeping in mind the information below:

1. If you have implemented a cutom Issue Tab Panel plugin you need to be aware of this API change.

If you are upgrading to JIRA 3.3.1 from a previous version, due to web browser caches, changes to JIRA's Issue Navigator might appear corrupted or unstyled. Please refresh your browser's cache (press Shift+Reload on the Find Issue's page) for the changes to appear correctly.

JIRA 3.3.1 Issue Tab Panel extension

Unknown macro: {version-warn}

JIRA 3.3.1

JIRA 3.3.1 Issue Tab Panel extension

You only need to worry about this page if you have a implemented a custom Issue Tab Panel JIRA plugin

Before JIRA 3.3.1 every `IssueAction` object that a `IssueTabPanel` returned would appear on the "All" tab on the View Issue page. Therefore an Issue Tab Panel could not return an IssueAction with a simple informational message, without poluting the "All" tab with this message. (This caused the JRA-7822 bug).

In JIRA 3.3.1 the `isDisplayActionAllTab()` method was added to the `com.atlassian.jira.issue.action.IssueAction` interface. This method should return "true" if the `IssueAction` should appear on the "All" tab, and "false" otherwise.

Therefore, if you have implemented this interface directly in your plugin, you will need to implement this method and recompile your plugin before deploying it into JIRA 3.3.1. If you have extended a JIRA class instead, e.g. `com.atlassian.jira.issue.action.AbstractIssueAction` or `com.atlassian.jira.issue.action.AbstractGVIssueAction` you do not need to recompile your plugin.

JIRA 3.2 Release Notes

JIRA 3.2 Release Notes

Atlassian Software Systems is proud to announce the latest release of *the* issue tracking and project management application - **JIRA 3.2** (download). Following the tradition of frequent and worthwhile upgrades, **JIRA 3.2** once again raises the bar in the professional issue tracking arena with a host of new features, improvements and bug fixes. Included in this release:

- Customisable fields for transitions
- Tabbed field screens
- Contextual custom fields
- Extended Bulk Edit Capabilities
- Improved internationalisation
- Improved performance
- ...and much more!

JIRA just got a whole lot better!

Upgrading to JIRA 3.2?

Please refer to the following documentation regarding the upgrade process to JIRA 3.2 from previous versions - JIRA 3.2 Upgrade Guide

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Contents

1. New Features
2. Improvements
3. Bug Fixes

New Features

JIRA 3.2 Documentation

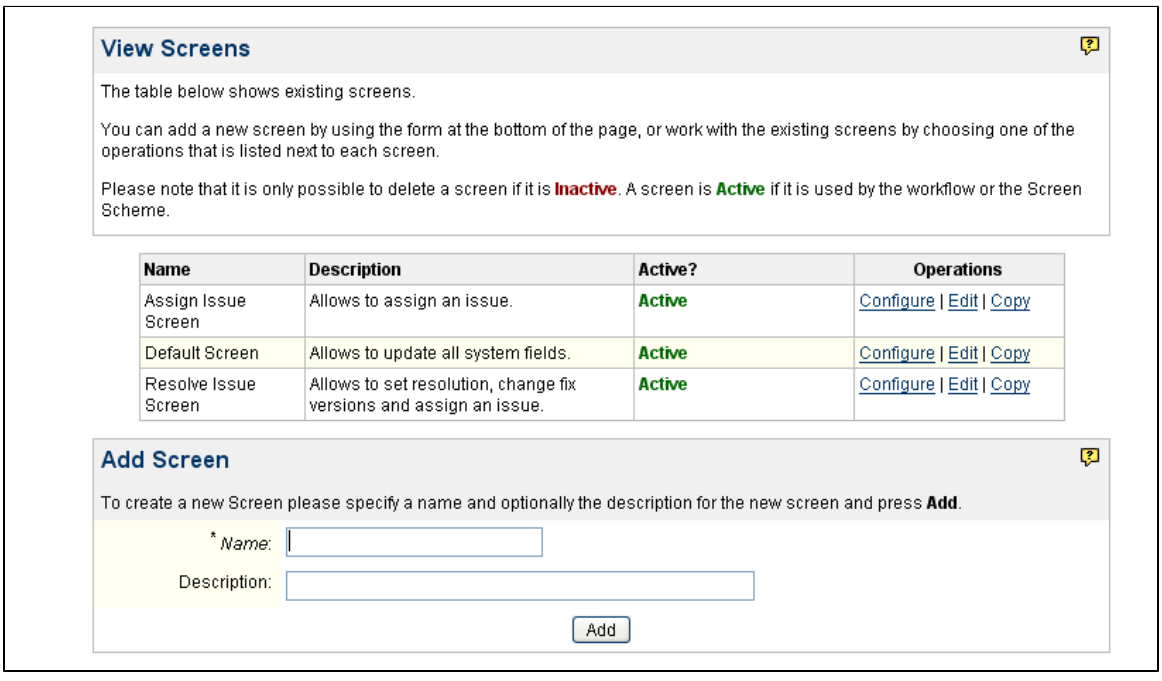
Further information on all these new features can be found in the JIRA 3.2 documentation. A full list of the new features and improvements in JIRA 3.2 can be found here.

Field Screens

JIRA 3.2 allows configuration of field position and visibility for each issue operation and in Professional and Enterprise editions for each workflow transition screen.

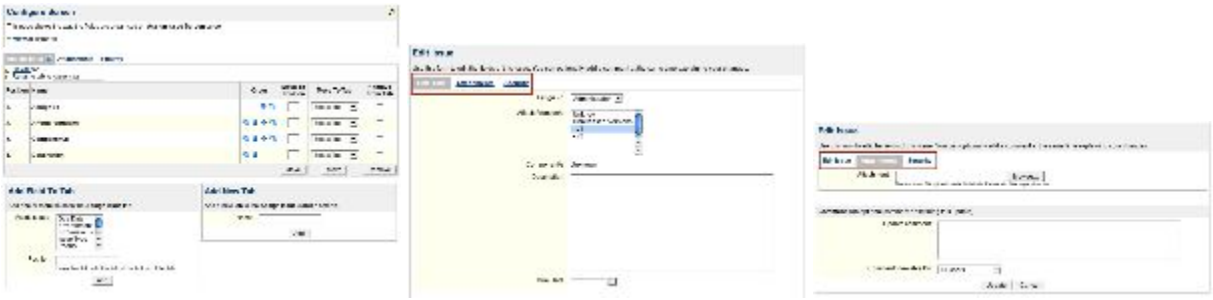
Each page on which you view and enter data for an issue is now a customisable "field screen". This provides for a flexible, more intuitive interface, with the ability to configure exactly which fields are presented for each operation. Each screen contains only those fields that directly relate to the operation being performed - while fields that do not relate to the operation can be hidden.

For example, it is possible to configure the *'Create Issue'* screen to display the *'Assignee'* field, while hiding this field in the *'Edit Issue'* screen. It is also possible to configure JIRA such that certain fields (including custom fields) appear only on specific workflow transitions: for example, if you have a *'QA Contact'* custom field, you can now configure JIRA to ensure that it is shown only on the *'Close Issue'* transition.



Field Screen Tabs

With JIRA Enterprise edition, you aren't even limited to a single field screen. Each page can have its fields divided amongst a group of tabs, creating a less cluttered environment as less used fields (e.g. attachments) can be placed in separate tabs.

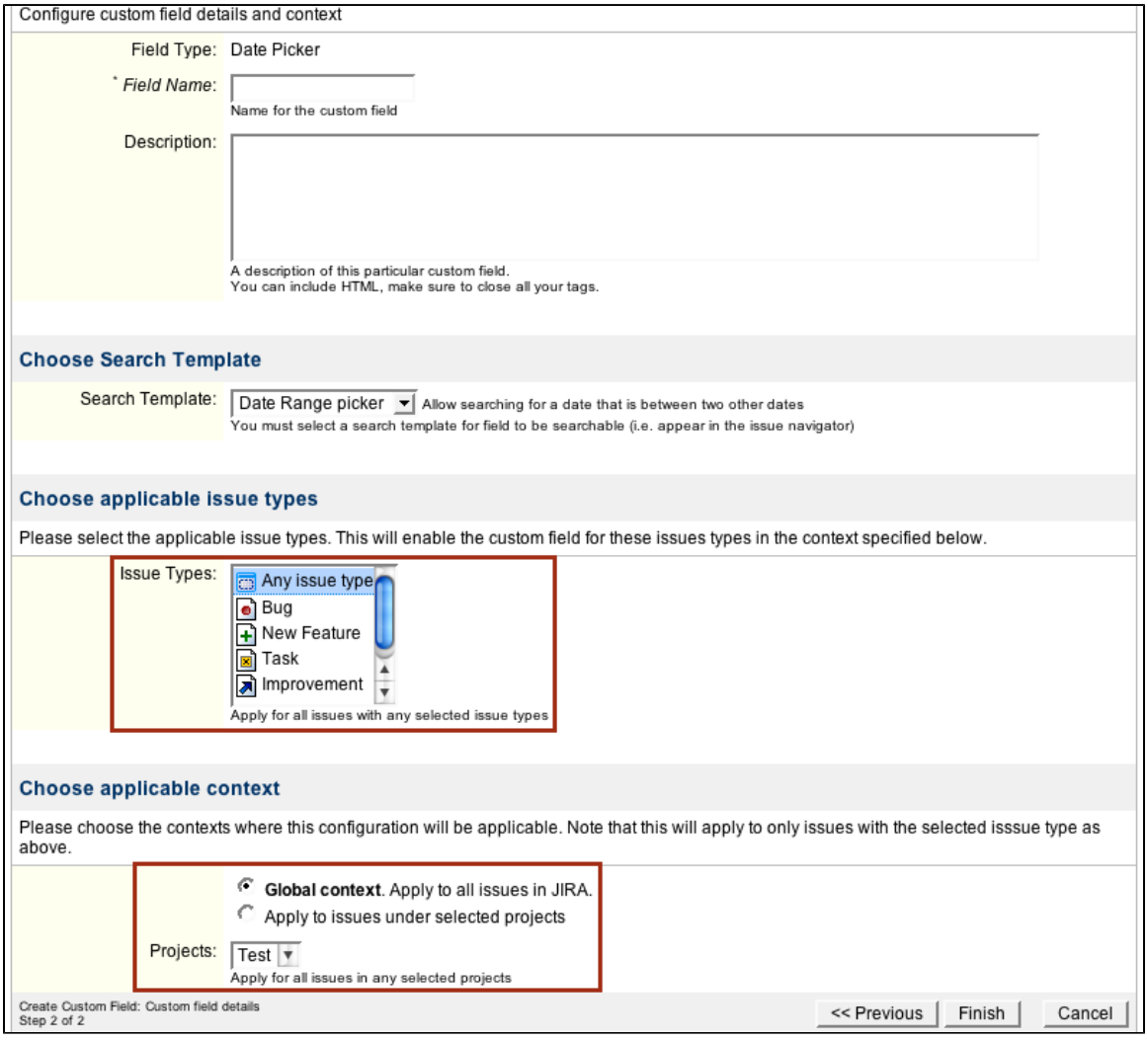


Contextual Custom Fields

Custom fields have been enhanced even further to allow greater flexibility and potential for customisation. Custom Fields are no longer limited to a single project or issue type, but can be shared between multiple issue types and multiple projects. Gone is the need to create the same custom field over and over again. Custom fields can be defined within a specified context or set of issue types, allowing them to be defined exactly where you need them.

Custom fields related to several projects and/or issue types can be configured to appear for those projects and/or issue types. For example, a custom field *'Customer ID'* could be associated with the *'Sales'* and *'Customer Info'* projects, while a custom field *'Operating System'* could be associated with the *'Bug'* and *'Improvement'* issue types in the *'Support'* project.

With configuration contexts, the same custom fields can still be configured differently for each project: for example having different default values.



Extended Bulk Edit

Further extending the bulk edit functionality, it is now possible to bulk edit more fields - a much requested feature. The list of fields that can be modified has been extended to include:

- Due Date
- Reporter
- Issue Security Level
- Issue Type

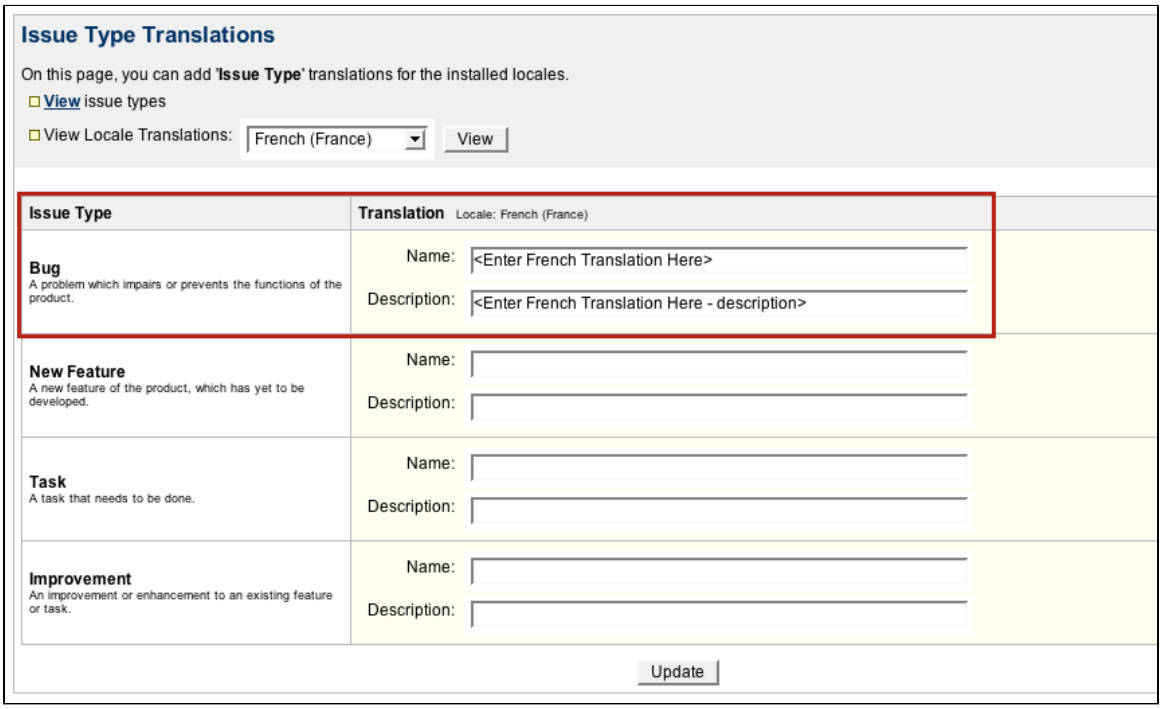
The bulk edit of assignee also works across multiple projects - allowing the selection of only valid assignees across all projects.

Internationalisation - *Viva la JIRA!*

Issue Constant Translations

Continuing our commitment to a fully internationalisable issue manager, JIRA 3.2 provides the ability to translate all issue settings - i.e. Priorities, Statuses, Issue Types and Resolutions. This allows for even more complete translations to present JIRA to each user in their own chosen language.

Hence, users who have selected the French locale will be presented with the French translations for the issue settings, while the Spanish users will be presented with the Spanish translations. The translations are presented throughout the JIRA interface - in issue creation/editing/viewing, reports, portlets, etc.



New Language Pack

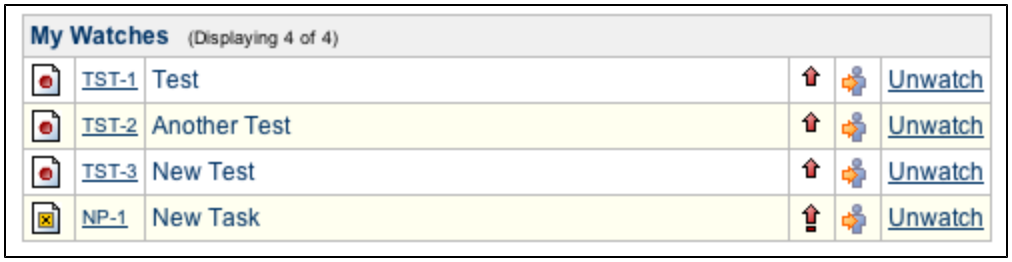
With the addition of Traditional Chinese, JIRA's user interface can be displayed in 12 languages.

Subversion - Multiple Repositories Support

The most popular JIRA Subversion plugin feature request has been addressed with support for multiple repositories now included in the latest plugin release. This allows users to map multiple JIRA projects to multiple Subversion repositories.

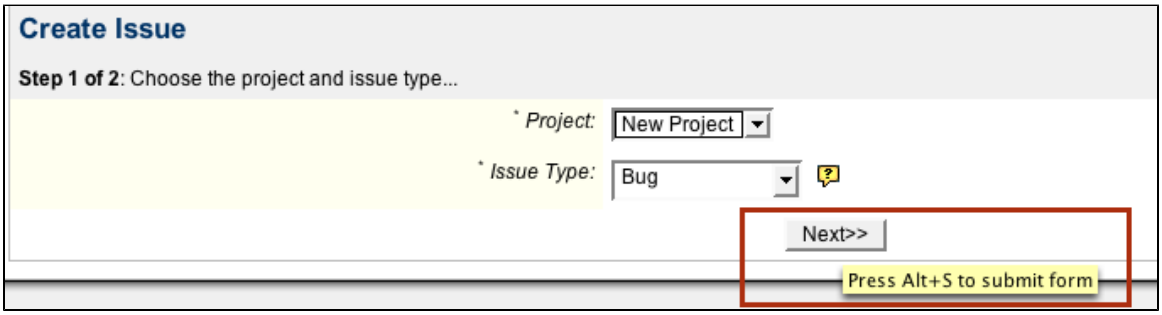
New Portlets

JIRA 3.2 introduces the **Voted Issues** and **Watched Issues** portlets - respectively showing the issues voted for and watched by the current user. These portlets can be added to the dashboard in the normal manner - providing quick access to the data you need.



Shortcuts

Navigating through JIRA is now quicker than ever with the addition of a number of shortcuts. Every form - from issue creation to editing a version - in JIRA can be submitted with the shortcut **ALT+S**. Form submission can be canceled using the shortcut **ALT+`** (ALT + backquote).



For more information on keyboard shortcuts please see our documentation.

Profile Email

In order to protect against spammers harvesting email addresses from JIRA, this release includes the functionality to hide the email addresses located on the user profile page. It is possible to configure the display of this information as follows:

- Show
- Hide
- Mask (e.g. *'user@example.com'* becomes *'user at example dot com'*)
- Show to logged in users only

Improvements

Performance - *JIRA goes to boot camp!*

One of the main goals of this release was to improve on overall performance of JIRA and there are major improvements in memory usage and speed. Utilising the Lucene search engine and some clever coding, some of the notable enhancements include:

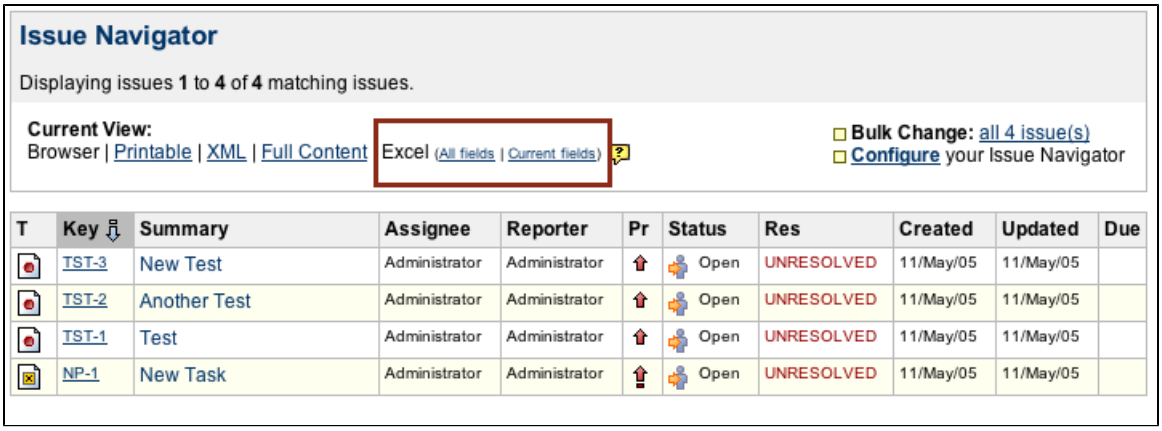
- Issue Navigator
  - **Much** quicker searching and Sorting (usually under 1s)
- Browse Project
  - Quicker **Project**, **Popular Issues** and **Open Issues** reports benefit with *up to **10x** improvement*
- Portlets
  - **Project** portlet is now ***2x** faster*
  - **Assigned to Me** portlet is *up to **10x** faster* and more efficient
- Importing and Indexing
  - Restoring JIRA backups or upgrading to a new version of JIRA is now ***2x** faster*.

Smart Query - *let JIRA find it for you!*

The 'Smart Query' functionality within JIRA has been further extended to provide easy access to commonly requested searches. For example, entering the query *'unresolved'* will direct the user to a list of unresolved issues. Further details on this feature can be found here.

EXCEL View

Many users requested that the EXCEL view display the same columns as displayed in the Issue Navigator view ... now it is possible to display *all* columns or just those visible in the the issue navigator.



Email Notification

Many users requested the ability to change the format of the ***From*** email address within JIRA notifications. The format is now fully configurable - with the ability to include or exclude the user name, email address and email hostname: for example, the format can be set to display as follows 'John Doe (ATLASSIAN) <jira@atlassian.com>'.

JDK 1.5 Compatibility

JIRA 3.2 is now fully compatible with JDK 1.5!

Bug Fixes

This release includes nearly 200 bug fixes - the best way to see them is them is to ask JIRA - JIRA 3.2 Bug Fixes.

> **Upgrading to JIRA 3.2**
>
> Please refer to the following documentation regarding the upgrade process to JIRA 3.2 from previous versions - JIRA 3.2 Upgrade Guide

JIRA 3.2 Upgrade Guide

This page contains information you need to know when upgrading to JIRA 3.2. The general upgrade instructions can be found here.

1. If you have written any Custom Field Type plugins please refer to this document
2. If you have created any Workflow plugins (custom Validators or Post Functions) please read this document.
3. If you have any custom file based workflows (workflows *not* created through JIRA's Workflow Editor) please read this document.
4. If you wish issues that are associated with the default system workflow and are closed to be bulk editable - please read this.

Notifications now respect permissions

In 3.2, JIRA respects the permission scheme and security levels when sending notifications (see JRA-5743. People who won't be able to see an update online won't get a notification email.

This has one important effect: if you have a project where:

- the notification scheme specifies that a raw email address (eg. developers@mycompany.com) should be notified, and
- 'Browse' permission has not been granted to 'Anyone' (eg. it is granted to 'jira-users'

  then that email address ('developers@mycompany.com' in our example) won't be mailed. As JIRA cannot verify that the recipient(s) of the email address have the 'browse' permission, it makes the conservative assumption that they are not.

This can be fixed by creating a user (eg. 'developers') for the email address, making it a member of a group that has 'Browse' permission, and adding it as a recipient of notifications. The raw email address should then be removed from the notification scheme, as it serves no purpose.

Notifications no longer sent to raw email addresses if anonymous browsing disabled

In 3.2, JIRA respects the permission scheme and security levels when sending notifications (see JRA-5743. People who won't be able to see an update online won't get a notification email.

This has one important effect: if you have a project where:

- the notification scheme specifies that a raw email address (eg. developers@mycompany.com) should be notified, and

- 'Browse' permission has not been granted to 'Anyone' (eg. it is granted to 'jira-users'
  then that email address ('developers@mycompany.com' in our example) won't be mailed. As JIRA cannot verify that the recipient(s) of the email address have the 'browse' permission, it makes the conservative assumption that they are not.

This can be fixed by creating a user (eg. 'developers') for the email address, making it a member of a group that has 'Browse' permission, and adding it as a recipient of notifications. The raw email address should then be removed from the notification scheme, as it serves no purpose.

Restricting Edit based on Issue Status

Restricting Edit based on Issue Status

JIRA 3.2 ensures that issues cannot be bulk edited if they cannot be edited normally due to their workflow status. The default workflow restricts issues in the *Closed* status from being edited.

Restricting Edits

A new property - **jira.issue.editable** - has been added that allows the administrator to specify which statuses/steps within a workflow are editable. The administrator can set the flag to be false for a given status - any issue assocaited with the workflow in this status will not be editable or bulk editable.

An example can be found in the default workflow - the **Closed** status has the property key set to false - restricting the ability to bulk edit any issue in this status.

In order to change this behaviour, it is necessary to copy and edit the default workflow and associate your issues with the new workflow.

Upgrading custom CustomFieldTypes in JIRA 3.2

Changes to the custom field types code

For JIRA 3.2, there has been incremental upgrades to the custom fields code. If you have written your own Custom Field Types, you'll want to read this document. We recommend that you ensure that your custom field types still compile against the latest 3.2 build and verify this under a testing environment before putting it into production. It is vital that at the time of the data upgrade, all of your custom field types are functioning correctly, or else they may not be upgraded correctly.

- CustomFieldType Interface changes
- Velocity template changes
- Abstract class changes
- CustomFieldType deal with CustomFieldConfig and not CustomField objects

CustomFieldType Interface changes

There has been some new methods added to the CustomFieldType interface. You may need to implement them if you didn't extend one of the existing concrete classes or AbstractCustomFieldType. The main change here is that FieldValidationException are now thrown instead of CustomFieldValidationException and the addition of two methods.

```
    /**
     * Return the String value object from the CustomFieldParams. The object may be a single String
(e.g. TextCFType,
     * List of Strings (e.g. MultiSelectCFType) or CustomFieldParams of Strings (e.g.
CascadingSelectCFType)
     *
     * @param parameters - CustomFieldParams containing String values
     * @return String value object from the CustomFieldParams
     */
    public Object getStringValueFromCustomFieldParams(CustomFieldParams parameters);

    /**
     * Returns a List of {@link CustomFieldConfigItemType} objects.
     * This opens up possibilities for configurable custom fields
     *
     * @return List of {@link CustomFieldConfigItemType}
     */
    public List getConfigurationItemTypes();
```

Velocity Template Changes

The parameters passed to velocity templates for view and editing custom fields have also significantly changed. To update your velcoity templates, you should change your headers to be:

```
#controlHeader ($action $customField.id $customField.name $fieldLayoutItem.required
$displayParameters.noHeader)
...
#controlFooter ($action $fieldLayoutItem.fieldDescription $displayParameters.noHeader)
```

Abstract class changes

There has also been changes to the abstract CustomfFieldTypes with an addition of a GenericConfigManager dependency. You'll need to add this to all constructors of classes which extends them. You can use GenericConfigManager as an arbitrary data store.

CustomFieldType deal with CustomFieldConfig and not CustomField objects

Custom fields can now be configured differently per context. Thus the CustomFieldType now gets passed the CustomFieldConfig object (which is an instance of the configuration set) rather than just the CustomField themselves.

Upgrading Workflow Plugins for JIRA 3.2

If you have written a workflow plugin (Validator or Post Function) for JIRA 3.0 or 3.1 you might have to modify it to make it work for JIRA 3.2. If you are getting ClassCastExceptions while transitioning issues through workflow after upgrading to JIRA 3.2 there is a high chance that this document is what you are looking for.

Workflow Conditions still use GenericValues for issues, so if you have written a custom Workflow Condition it should work with JIRA 3.2

Regular Workflow Transition

For a regular workflow transition in JIRA 3.1 the `transientVars` map contained a GenericValue object that represented an issue. The `transientVars` Map also contained the fields that were changed during the transition (if any), namely Fix Versions, Assignee and Resolution.

In JIRA 3.2 an Issue object has been created, and all the changes that have been made to the issue can be obtained from the Issue object:

```
Issue issue = (Issue) transientVars.get("issue");
Map modifiedFields = issue.getModifiedFields();
```

Please note that in version of JIRA 3.1 and earlier the `transientVars` map contained a `GenericValue` object with the key "issue". If your code does something like:

```
GenericValue issue = (GenericValue) transientVars.get("issue");
```

This will cause a ClassCastException in JIRA 3.2. You need to cast the object to `Issue` instead of `GenericValue`. If you need to get the `GenericValue` of the issue, you can do that by calling:

```
GenericValue issueGV = issue.getGenericValue();
```

As mentioned earlier, the `modifiedFields` map contains all the fields that have been updated during the workflow transition. The keys of the `modifiedFields` map are ids of fields (please see `com.atlassian.jira.issue.IssueFieldConstants`) that have been modified, and the values of the `modifiedFields` map are `ModifiedValue` objects. A ModifiedValue object represents an updated field. The object stores the old and the new value of the field for the issue. You can use this object like so

```
Map modifiedFields = issue.getModifiedFields();
for (Iterator iterator = modifiedFields.keySet().iterator(); iterator.hasNext();)
{
    String fieldId = (String) iterator.next();
    ModifiedValue modifiedValue = (ModifiedValue) modifiedFields.get(fieldId);
    // Old value of the field
Object oldValue = modifiedValue.getOldValue();
    // New Value of the field
Object newValue = modifiedValue.getNewValue();

}
```

Please note, that the `comment` and `commentLevel` are still recorded in the `modifiedFields` map.

Initial Workflow Transition

An initial workflow transition is the transition that creates an issue.

In JIRA 3.1 and earlier the `transientVars` map contained the field values that should be used to create an issue. In JIRA 3.2 the value of the fields are recorded in the Issue object rather than in the `transientVars` map. You can use various getter methods to retrieve values of these fields. For example, if you need to get issue's description, instead of doing:

```
String description = (String) transientVars.get(IssueFieldConstants.DESCRIPTION);
```

do this:

```
Issue issue = (Issue) transientVars.get("issue");
String description = issue.getDescription()
```

Using Oracle 10g drivers to solve the 4000 character limitation

As you might be aware Oracle has a 4000 character limitation on VARCHAR2 fields. Which causes quite a few headaches when dealing with custom workflows or working with issues that have long descriptions, comments or custom field values.

Fortunately Oracle have worked around the VARCHAR2 limitation in their latest Oracle 10g JDBC driver. This fix (described online here) works with Oracle 9 and 10. We strongly recommend using Oracle 10g drivers and the setup described below if you are using Oracle 9i or 10g.

Even though Oracle suggests that Oracle 10g drivers work with Oracle 8i, users have reported problems with this configuration.

Please follow the Upgrading JIRA Safely instructions, keeping the following in mind:

Use Oracle 10g driver

This configuration will work **only** with Oracle 10g drivers. Therefore, from Oracle's site download the ojdbc14.jar (or applicable) JDBC driver, and copy it to your app server's lib/ directory (eg. common/lib/ for Tomcat). Remove the old JDBC jar used previously.

Create a new database for JIRA 3.2

Please create a **new** database for JIRA 3.2 and configure JIRA's data source to use it. Please do *not* point JIRA 3.2 at your old database.

Set the SetBigStringTryClob flag

When configuring the data source for the new database set the SetBigStringTryClob flag to true. The way this must be done depends on your application server

JIRA Standalone, Tomcat 4 and 5.0:

Edit conf/server.xml (Tomcat 4) or conf/Catalina/localhost/jira.xml (Tomcat 5.0), locate the section where the 'jdbc/JiraDS' DataSource is set up, and add:

```
<parameter>
  <name>connectionProperties</name>
  <value>SetBigStringTryClob=true</value>
</parameter>
```

For instance, in JIRA Standalone one would then have:

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"/>
    <ResourceParams name="jdbc/JiraDS">
      <parameter>
        <name>driverClassName</name>
        <value>oracle.jdbc.driver.OracleDriver</value>
      </parameter>
      <parameter>
        <name>url</name>
        <value>jdbc:oracle:thin:@<database host machine>:<port>:<SID></value>
      </parameter>
      <parameter>
        <name>username</name>
        <value>...</value>
      </parameter>
      <parameter>
        <name>password</name>
        <value>...</value>
      </parameter>

<parameter>
  <name>connectionProperties</name>
  <value>SetBigStringTryClob=true</value>
</parameter>

      <parameter>
        <name>factory</name>
        <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
    </ResourceParams>
```

Tomcat 5.5

In Tomcat 5.5, the format for the added section would be:

```
<Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
    username="jirauser"
    password="jirauser"
    driverClassName="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@localhost:1521:jiradb"

connectionProperties="SetBigStringTryClob=true">

</Resource>
```

Orion / OC4J

For Orion/OC4J, edit config/data-sources.xml, and add the property as a nested tag:

```
<data-source
    class="<datasource driver class>"
    name="<name>"
    location="<location>"
    xa-location="<xa-location>"
    ejb-location="<ejb-location>"
    url="<url>"
    connection-driver="<driver>"
    username="<login>"
    password="<password>"
    inactivity-timeout="30"
>
  <property name="SetBigStringTryClob" value="true" />
</data-source>
```

Other app servers

Consult the relevant JIRA app server guide and the app server documentation to find how to add the property.

Use oracle10g field-type-name

Please specify **oracle10g** (*not* oracle) as the `field-type-name` when editing `WEB-INF/classes/entityengine.xml`

After the data is re-imported and JIRA upgrades the data the 4000 character limitation should disappear.

Workflows using default "Closed" status

Prior to JIRA 3.2, workflow steps using the default "Closed" JIRA status had the special behaviour that disabled certain operations such as "Edit". Since 3.2, this behaviour has been made available to **all** workflow steps, regardless to status. You can disable issue editing on a workflow step by adding a meta attribute "jira.issue.editable=false" to the relevant steps.

For backwards compatibility, all workflow steps with the "Closed" step that is stored in the database have been upgraded to have this new meta attribute. So for the majority of users, JIRA will have already done the upgrading for you. For those with workflows that are stored in the file system, you will have upgrade manually (if you want the closed status to disallow edits). Instructions are below:

Upgrading file based workflows

You only need to do this upgrade if you have created workflows as XML documents on disk and if you want the step to disallow edits. For most JIRA installations workflows are created using the workflow editor. These workflows will be upgraded automatically.

First, find the workflow step with the status id of 6 (closed)

```
<meta name="jira.status.id">6</meta>
```

Simply add the following to the step:

```
<meta name="jira.issue.editable">false</meta>
```

Restart JIRA and all the new changes should take effect.

3.2 performance benchmarks

Here are some quick benchmarks to illustrate the performance improvements in JIRA 3.2 vs. 3.1.1.

Return to the 3.2 release notes

Sample JIRA data details

Issues	14862
Comments/changes	38294

Users	9163 (most inactive)

Performance comparison

Operation	3.1.1	3.2
Full reindex	429s	287s
Search	37s	1.1s

Search returning first 25 of 14,862 issues, average over multiple runs.

System

Software: JIRA 3.1.1 Enterprise; all issues cached.
System: 2.6Ghz Pentium 4
Database: MySQL 4.1.9

JIRA 3.2.3 Release Notes

JIRA 3.2.3 Release Notes

In the tradition of frequent and worthwhile updates, JIRA 3.2.3 is released today in Standard, Professional and Enterprise editions. This point release includes 16 bug fixes and improvements. It can be downloaded here. See the JIRA 3.2 Upgrade Guide before upgrading.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.2.3 includes 16 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11326' does not exist for the field 'fixVersion'.

JIRA 3.2.2 Release Notes

JIRA 3.2.2 Release Notes

In the tradition of frequent and worthwhile updates, JIRA 3.2.2 is released today in Standard, Professional and Enterprise editions. This point release includes over 30 bug fixes and improvements. It can be downloaded here. See the JIRA 3.2 Upgrade Guide before upgrading.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

JIRA 3.2.2 includes over 30 bug fixes and improvements.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11272' does not exist for the field 'fixVersion'.

JIRA 3.2.1 Release Notes

JIRA 3.2.1 Release Notes

In the tradition of frequent and worthwhile updates, JIRA 3.2.1 is released today in Standard, Professional and Enterprise editions. This point release includes over 50 bug fixes and improvements. It can be downloaded here. See the JIRA 3.2 Upgrade Guide before upgrading.

JIRA 3.2.1 includes over 50 bug fixes and improvements.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11251' does not exist for the field 'fixVersion'.

JIRA 3.1 Release Notes

JIRA 3.1 Release Notes

Following JIRA 3.0.3, Atlassian is proud to release the latest version of JIRA in Standard, Professional and Enterprise editions - **JIRA 3.1** (download).

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

In the tradition of frequent, worthwhile upgrades, JIRA 3.1 includes over 80 bug fixes, improvements and new features. See the JIRA 3.0 Upgrade Notes before upgrading.

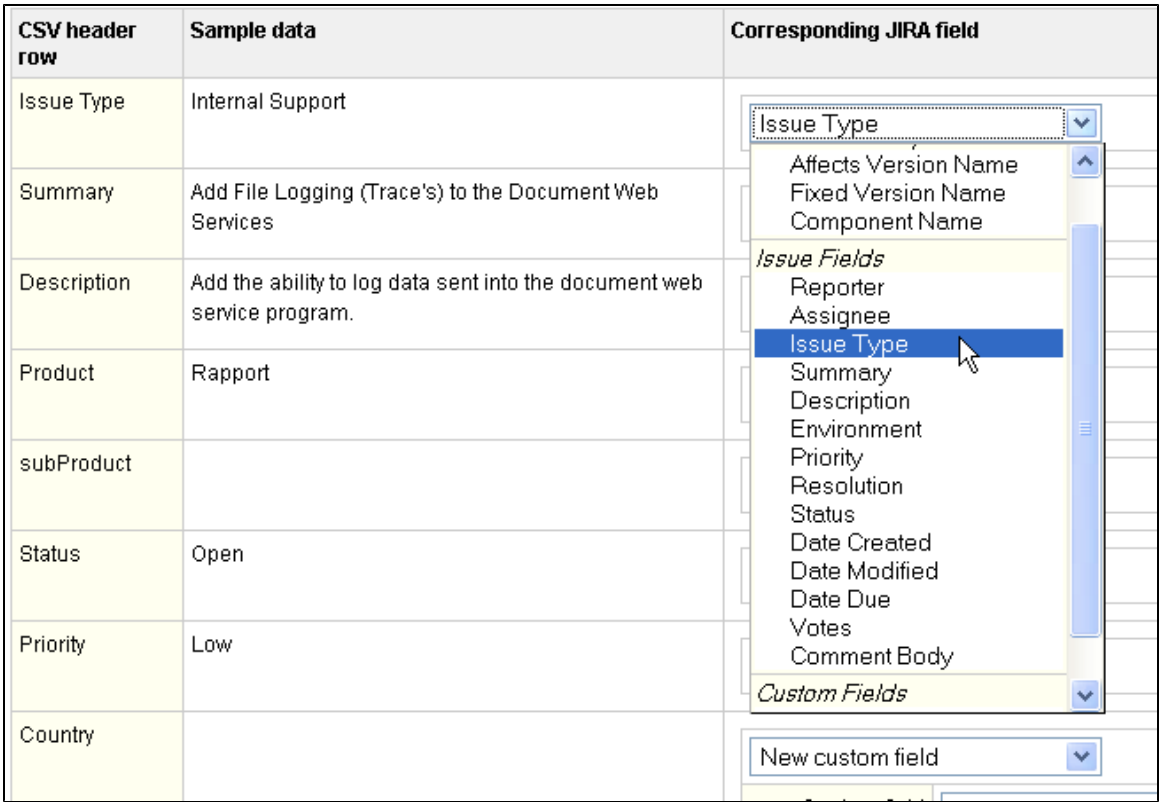
Contents

1. New Features
2. Improvements
3. Bug Fixes
4. Editions

New Features

CSV Importer Wizard

The CSV importer, new in JIRA 3.1, allows you to import issues from any comma-separated file. JIRA's fully configurable wizard will step you through the process of converting your import file into JIRA issues.



The wizard provides the following features:

- Automatic creation of missing users, versions and components
- Easy creation of resolutions, priorities and issue types.
- Creation of custom fields on the fly
- Simple mapping of issue values to values in JIRA

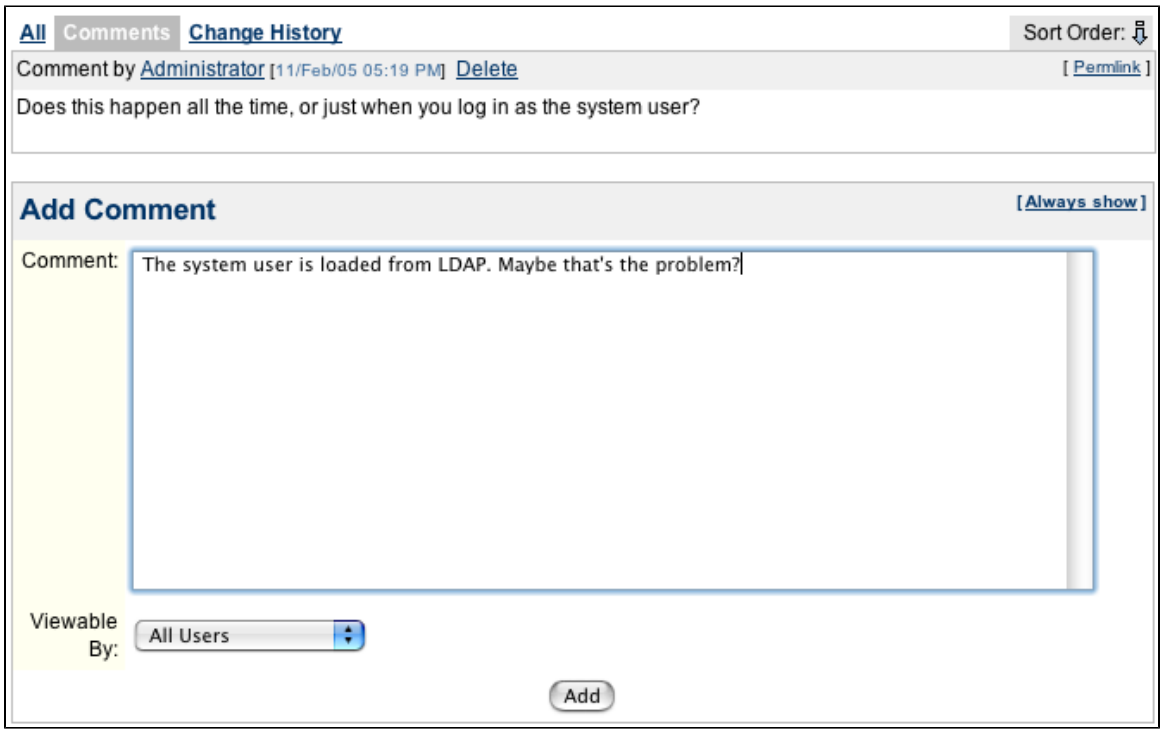
Webwork Plugin Type

JIRA 3 plugins allow developers to extend JIRA in many different ways - with new reports, custom fields, workflow conditions and more. JIRA 3.1

gives developers even more power with the webwork plugin type: enabling plugin creators to integrate their own actions right into JIRA's web framework.

Add Comment on 'View Issue' field

A much-requested feature, you can now see what you are commenting on! The comment panel is now displayed in the same page as the issue details, so you don't have to switch between comment and issue to remember what you're talking about.



Workflow Action Keyboard Shortcuts

For the power-user or just the impatient, alt-1 to alt-9 (ctrl-1 to ctrl-9 on OS X) will execute the corresponding workflow action for an issue.

Improvements

RPC / SOAP Improvements
The RPC/SOAP integration with JIRA has been improved to include new types (e.g. RemoteGroup, RemoteUser, RemotePermission, etc.) and new services (e.g. IssueService. ProjectService, UserService, etc.). A full list of the new features is available here.

Improved Performance

JIRA 3.1 includes a number of performance improvements. Most notably, the data import process has been revised - now requiring less memory with the process completing in less time. The import process should also handle invalid characters encountered in the XML. A number of reports regarding slow responses from JIRA while editing workflows have also been addressed.

Assign Issues by Mail

The 'Create Issue Handler' can be configured so that issues created via email are automatically assigned to the user listed in the email's 'CC:' field.

Internet Explorer UI Compatibility

This release contains many fixes addressing issues with Internet Explorer UI incompatibilities. All pages should appear identically - whether viewed in IE or Firefox or Safari or ...

Notable Bug Fixes

JIRA 3.1 includes over 70 Bug Fixes.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10792' does not exist for the field 'fixVersion'.

Notable Bug Additions

The following bugs have been reported so far, and will be fixed in 3.1.1:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10792' does not exist for the field 'affectedVersion'.

Edtions

Standard? Professional? Enterprise? Which edition is for you?

In order to provide you with greater choice, JIRA is offered in a number of editions - with different feature sets and capaibilities available in each edition.

Check out the feature comparison at:

http://www.atlassian.com/software/jira/comparison.jsp

or download an evaluation to determine which edition best addresses your requirements.

JIRA 3.1 Upgrade Notes

This page lists a few things to be aware of when upgrading from JIRA 3.0.x to JIRA 3.1. To perform the actual upgrade, see the upgrade documentation. For upgrading from JIRA 2.x to JIRA 3.x see JIRA 3.0 Upgrade Notes

MySQL Users dB upgrade (JRA-5635)

The size of the descriptor field in the jiraworkflow table has been increased. MySQL users will see warnings when they start their app server. This can be fixed by running the SQL below. This will also allow for Workflows of up to 4GB as opposed to just 64k

```
alter table jiraworkflows change DESCRIPTOR DESCRIPTOR LONGTEXT;
```

JIRA 3.1.1 Release Notes

JIRA 3.1.1 Release Notes

In the tradition of frequent and worthwhile updates, JIRA 3.1.1 is released today in Standard, Professional and Enterprise editions. This point release includes over 40 bug fixes along with a number of improvements and new features. It can be downloaded here. See the JIRA 3.1 Upgrade Notes before upgrading.

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

New Features, Improvements and Bug Fixes

This release includes a number of new features and improvements - including:

- Ability to specify custom field values while creating an issue via XML-RPC/SOAP
- Improved translations for various locales.
- New custom field - DateTime
- Fixed portlet hyperlinks

JIRA 3.1.1 also includes over 40 Bug Fixes.

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '11130' does not exist for the field 'fixVersion'.

JIRA 3.0 Release Notes

Atlassian is proud to present the next major evolution in JIRA - JIRA 3.0! These are the release notes for the final release of JIRA 3.0. There are addendum pages for 3.0.1, 3.0.2 and 3.0.3.

> **JIRA 3.13.5** has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
> *Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!*

Existing customers who wish to upgrade, or new users who wish to try out JIRA 3.0 for 30 days can download either the standalone or WAR distributions from the Atlassian website.

What's new in JIRA 3.0?

JIRA 3.0 is the second major JIRA overhaul (after 2.x), and the largest release in Atlassian's short history. As always, all existing customers who

are still within the 12 months maintenance period can upgrade for free, thanks to the provision for one year of free upgrades in your license.

You will find JIRA 3.0 contains many significant new features as well as the many, many enhancements, bug fixes and things we just couldn't resist throwing in at the last minute. By our calculations, over 400 issues were collectively resolved for JIRA 3.0!

New JIRA editions

Previously, JIRA came in 2 flavours - Professional and Enterprise. JIRA 3.0 marks the introduction of a new family member, JIRA Standard edition, which fits below the Professional edition. For more details on what features are in which editions - see JIRA 3.0 Feature Comparison.

Note: all existing JIRA Professional licenses who are still within the 12 months maintenance period are able to upgrade to JIRA 3.0 Professional at no cost - so for example if you are a JIRA 2.x Professional customer, you can upgrade to JIRA 3.0 Professional and will get all the Professional features.

Upgrading from previous versions

Upgrading JIRA should be pretty easy, see the JIRA 3.0 Upgrade Notes, and the guide to upgrading JIRA.

Documentation

The JIRA 3.0 documentation is online here.

Contents

1. New Features
2. Improvements

New Features

JIRA 3.0 contains a lot of major new features - more than in any other single release of JIRA. Let's look at a few of the most important ones:

Workflow Editor & Configurable Workflow

The most anticipated new features in JIRA 3.0 are the workflow editing features. JIRA now contains a workflow editor (see screenshot) that allows you to view and edit workflows through the web interface, in the Professional and Enterprise editions. The workflow controls the set of steps which an issue moves through in order to move to a resolution.

Editor features:

- Workflows can be created with arbitrary steps (linked to statuses) and transitions between them.
- Statuses can be added and removed through the web interface.
- Conditions, functions, validators are now JIRA plugins (see below for details of the new plugin system):
    - Conditions allow you to govern when transitions can be executed and by whom (for example the transition from "Open" to "Closed" can only be executed by a user with the "Close Issue" permission).
    - Functions control exactly what happens after a workflow transition is executed (for example an event is fired, an issue field is changed or an email is sent).
    - Validators ensure that the data provided for a transition is correct.

Multiple Workflows - Enterprise users can configure multiple workflows for their JIRA installation, assigning a default workflow to each project, and then overriding that workflow for particular issue types if required (via a new workflow scheme). This powerful feature allows an issue to progress through a uniquely tailored workflow - one specifically designed for the life-cycle of that issue.

Sub-tasks

Issues are like people, they come in all shapes and sizes - large and small. Sometimes a given issue will be solved in multiple 'stages', often by different people. This is the genesis behind the sub-tasks feature.

Sub-tasks allow you to break an issue up into multiple tasks, each of which is a full issue in itself (with its own individual workflows). As you can see from the screenshot here, sub-tasks are very rapid to create and manipulate almost never requiring you to leave the View Issue screen.

Unlike other issues, sub-tasks also have a defined order and the issue contains an indicator of overall sub-task progress (see arrows and red/green indicators on screenshot).

As with all issue types, Enterprise users can override the default project workflow association and specify a custom workflow for each sub-task

type.

Plugin System

JIRA now has a plugin system. This means that developers can build plugins, which extend the functionality of JIRA in different ways, plugged in at multiple points within the interface.

Detailed information for developers interested in building their own plugins will be coming soon, but as a summary: each plugin is one or more classes and a simple XML plugin descriptor. Often plugins will also contain Velocity templates to render portions of the UI as necessary. A brief guide to JIRA plugins is available here.

Each plugin consists of one or more plugin modules, each of a specific type (see below). Both plugins and individual modules can be disabled or enabled through the administration interface (see screenshot).

The 12 current module types shipping with JIRA 3.0 are:

- Reports - define a report with the information collected to run it and the resulting HTML.
- Portlets - define portlets and the parameters they accept when configured.
- Custom Field Types - define new types of custom fields (for example a "User Picker") including their rendering templates.
- Custom Field Searchers - provide index and search capabilities to custom field types.
- Issue Tab Panels - add new panels to the View Issue screen (like existing "Comments" and "Version Control" panels).
- Project Tab Panels - add new panels to the Browse Project screen (like existing "Roadmap" and "Popular Issues" panels).
- SOAP RPC Endpoints - define new SOAP web services for JIRA.
- XML-RPC Endpoints - define new XML-RPC web services for JIRA.
- Components - implement new components within JIRA (or override existing components) that are provided to other plugin modules (via PicoContainer).
- Workflow Conditions / Functions / Validators - define extra conditions, functions and validators to be used in your JIRA workflow.

Upcoming Plugins

The popular requests for Subversion and Perforce support within JIRA will be addressed with the forthcoming release of two new plugins. While providing support for these extensions to JIRA, they are also great examples of creating plugins within the new plugin system. These plugins are due to be released shortly.

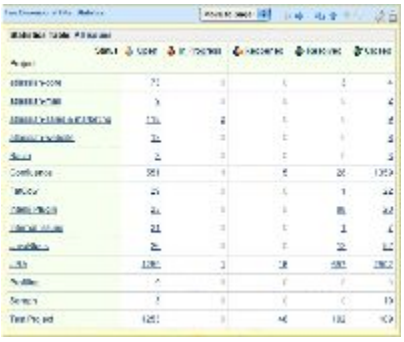
Dashboard Overhaul

The Dashboard has long been one of the best, most loved features in JIRA. With JIRA 3, it gets a whole lot better!

Inline Editing - the usability of the Dashboard has been greatly improved with the new inline editing mode. This allows you to move, add, delete and edit your portlets from the Dashboard itself, instead of having to switch back and forth to the Dashboard configuration screen (see screenshot).

Pluggable Portlets - portlets within JIRA have now moved within the plugin system as well, enabling easy creation and sharing of portlets with other users.

Filter Statistics Portlet - a new, very useful portlet has been added which allows you to calculate statistics for any saved filter. For example, you can create a filter showing the currently open issues assigned to yourself for a particular project - and then put a chart of those issues onto your Dashboard, broken down by component.



Two Dimensional Filter Statistics Portlet - building on the Filter Statistics portlet, this portlet allows you to display statistics from a filter in a configurable table. The X and Y axis of the table are configurable - for example, issue type against priority - and provides a powerful tool for manipulating the display of the statistics.

Multiple Dashboard Pages - And, to save the best for last, you can now have multiple dashboard pages in the Professional and Enterprise editions. This allows you to create multiple, named Dashboard pages which show up as different tabs on screen (see top right hand side of screenshot).

This is extremely useful in a number of scenarios. Often we've noticed that users' Dashboard pages become extremely long (up to 10 screens!) due to adding more statistic portlets - multiple dashboards allows you to split up these long pages. If you work with multiple projects it is fantastic to be able to create a Dashboard per project, each with relevant statistics providing deeper insight into the project.

Custom Fields

Custom Field Types - The custom field system has been completely overhauled for JIRA 3.0 (thanks to Joseph Dane of Hawaii University for the initial impetus!). Custom field types are now JIRA plugins, so new custom field types can be easily created, installed and shared by users. Custom fields themselves can now support multiple values, multiple search methods and can have configurable view / edit / search templates.

For example, imagine you wished to connect certain internal users within your organisation with a particular field in your issue. You could create

an 'internal user' custom field type edited with a simple drop down select box of user's full names pulled from a remote web service. To display this custom field, you could have a completely different template which actually included a picture of that user from your intranet!

New System Types - Using this custom field system, we've created a number of new custom field types for JIRA 3.0. The existing custom fields - text field, free text, select list, date picker and number field - all still exist, but now they're joined by the following new types out of the box:

- User Picker - choose a user from the user base via a popup picker window.
- Project Picker - choose from projects that the user can view in the system.
- Cascading Select - choose multiple values using two select lists.
- Multi Checkboxes - choose multiple values using checkboxes.
- Multi Select - choose multiple values in a select list.
- Radio Buttons - a list of radio buttons.
- URL Field - allow the user to input a single URL.
- Read-only Text Field - a read-only text label (only possible to create values programatically).
- Version Picker - choose from available versions in the project.

Voting and Watching

The most voted for JIRA feature has arrived! Users with the "Edit Watcher List" permission can now update and remove watchers. In this way, other users can be added as watchers of a particular issue (for example, I may want to make sure that Bob is notified of changes to a given issue).

For visibility, there is also a new governing "View Watchers and Voters" permission. Users with this permission can see the list of people who have voted on or are watching an issue.

The number of votes an issue has can now also be added to the issue navigator.

Other New Features

- **Change Reporter** - the second most requested JIRA feature has also arrived! Users with the "Modify Reporter" permission can now update the reporter of an issue.
- **Clone Issue** - you can now duplicate existing issues, optionally linked to the original issue. The sub-tasks of an issue are also cloned if any exist.
- **Multiple Attachments** - you can now upload multiple attachments at the same time if you want
- **Attachments while Creating** - attachments can also be added while creating the issue in a single step, rather than the existing two step 'create issue', 'attach file' process.
- **XML issue view** - each issue can now be viewed or downloaded as XML, including its comments and custom fields.

Improvements

There were many, many improvements made in JIRA 3.0, here are the major items of note:

- **Version management** - version management has been overhauled and versions now have a due date.
- **Issue Navigator** - sorting of columns has been extended with the ability to sort by Versions, Fix for Versions and Components (sorting on the earliest of each found).
- **CVS integration** - improved hugely in JIRA 3.0. JIRA can now access CVS repositories via SSH and the local file system (in addition to the existing pserver support), as well as display branch information for commits. The performance has also been worked on a lot, such that CVS access now requires a near constant amount of memory regardless of repository size.
- **More languages** - JIRA has now been translated to German, Spanish, Danish, Russian, French and Brazilian Portuguese.
- **Issue linking** - you can now link multiple issues at a time, as well as use search filters to find issues to link.
- **Project keys** - there is a configurable regular expression to govern the project key structure within JIRA.
- **SSO support** - JIRA integrates with a number of existing single sign-on frameworks, and can easily be customised to work with custom SSO systems.
- **Remote API** - the SOAP and XML-RPC remote APIs are distributed as a plugin, which can be updated independently of the main JIRA installation.
- **Remote Issue Creation** - issues can now be created and retrieved via the remote APIs.
- **Field Layouts** - Enterprise users can now define field layout schemes for each issue type within a project.
- **Reports** - there is now a version workload report (thanks to the JetBrains developers for this contribution) which shows the estimated times against developers for a given version.
- **Email notifications** - Improvements have been made to the format of notifications sent out by JIRA, especially text emails.
- **Startup time** - JIRA should now start up *much* faster than it did previously!
- **User interface** - As always, we've worked on improving the user interface in various areas (most notably the browse projects screen and the version administration screen).

and much, much more...

JIRA 3.0 Upgrade Notes

This page lists a few things to be aware of when upgrading from previous releases of JIRA to JIRA 3. To perform the actual upgrade, see the upgrade documentation.

Existing SMTP Mail Server 'From' address may break notifications (JRA-5089)

In JIRA 3, email notification 'From' addresses now contain the reporter name, eg. "Joe Bloggs (JIRA) <jira@company.com>", where "jira@company.com" is set by the admin as the SMTP mail server From address. If you have this address to already include a name (eg "Tech Support <jira@company.com>", then email notifications will fail with errors like:

```
2005-01-06 11:30:53,856 ERROR [atlassian.mail.queue.MailQueueImpl] com.atlassian.mail.MailException:
Sending failed;
  nested exception is: javax.mail.internet.AddressException:
Missing '<' in string ``"Joe Bloggs (JIRA)" <Tech Support <jira@company.com>>'' at position 62
```

Fix

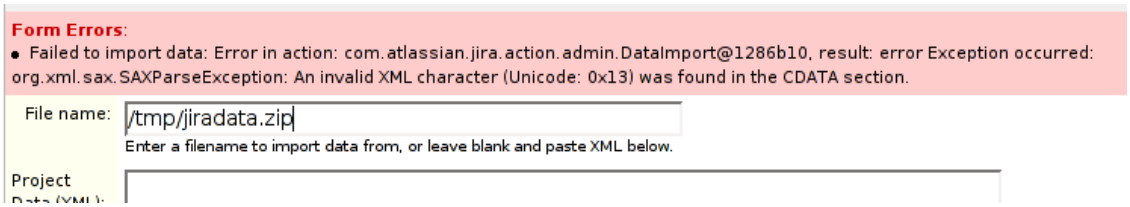
The fix is to edit `WEB-INF/classes/jira-application.properties`, and change the following property value to `false`:

```
jira.option.include.user.in.mail.from.address = true
```

- If using JIRA Standalone, the file is *atlassian-jira/WEB-INF/classes/jira-application.properties*, after which you should run bin/shutdown and bin/startup to restart.
- If using JIRA deployed as a webapp, copy *webapp/WEB-INF/classes/jira-application.properties* to *edit-webapp/WEB-IFN/classes*, make the change to the *edit-webapp* copy, run *build* to rebuild the webapp, and redeploy it on your app server.

Invalid characters break XML import

JIRA's recommended upgrade process involves deploying an XML backup of your data. Some users will find that the import fails with this error:



This is usually because the database contains control characters that cannot be represented in Unicode, and hence XML.

Fix

The fix is to follow these instructions to remove the invalid characters from the XML before import.

JIRA 3.0.3 Release Notes

JIRA 3.0.3 is a bugfix release. For the full list of changes from 2.x releases, see the JIRA 3.0 Release Notes. The JIRA 3.0 Upgrade Notes apply.

Changes since 3.0.2:

Bugs fixed:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10931' does not exist for the field 'fixVersion'.

Improvements:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10931' does not exist for the field 'fixVersion'.

JIRA 3.0.2 Release Notes

JIRA 3.0.2 is a bugfix release, mainly to address a performance problem with the 'find issues' page when there are large numbers of projects. For the full list of changes from 2.x releases, see the JIRA 3.0 Release Notes. The JIRA 3.0 Upgrade Notes apply.

Changes since 3.0.1:

Bugs fixed:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10924' does not exist for the field 'fixVersion'.

Improvements:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10924' does not exist for the field 'fixVersion'.

JIRA 3.0.1 Release Notes

JIRA 3.0.1 is a minor bugfix release. For the full list of changes from 2.x releases, see the JIRA 3.0 Release Notes. The JIRA 3.0 Upgrade Notes apply.

Issues addressed since 3.0

Bugs fixed:

JIRA 3.0.1 fixes one bug, which prevented users importing backup data into an empty JIRA instance:

Error formatting macro: jiraissues: java.lang.RuntimeException: A value with ID '10921' does not exist for the field 'fixVersion'.

See also JIRA 3.0.2 Release Notes

JIRA 2.6 Release Notes

JIRA 2.6

Only three months after the release of JIRA 2.5 (and two after our last point-release), Atlassian are continuing our tradition of frequent, worthwhile upgrades with JIRA 2.6. In the process, we've fixed more than 110 known bugs, and added over 70 individual improvements or new features. All a free upgrade if you've purchased your JIRA license in the last year.

JIRA 3.13.5 has been released. Read the full JIRA 3.13.5 Release Notes and Upgrade Guide.
Don't have JIRA 3.13? Take a look at the features of JIRA's latest released version and try it out!

As always, we strongly encourage you to back up your data before upgrading.

Contents

1. New Features
2. Improvements
3. Notable Bug-fixes
4. Outstanding Issues
5. Enterprise Features

New Features

CVS Integration

Prior to 2.6, the only way to integrate JIRA with CVS was via syncmail, a clumsy and error-prone solution. For 2.6, we have overhauled the CVS support so that now JIRA can get all the information it needs directly from the CVS repository itself. Wherever a JIRA issue key is mentioned in a CVS commit message, JIRA will link the commit to the issue, as shown below:

Enterprise customers also have the ability to associate multiple CVS repositories with a single project, just in case your project spans multiple repositories, or moves from one to another.

Quick Search

The quick search bar on the top right of the screen is a very useful feature in JIRA, allowing you to quickly jump to an issue by its key, or run a full text search. If you use the keyboard shortcut (*ALT+Q*), then it is even quicker.

In JIRA 2.6, we've made the quick search even more powerful: it now tries much harder to guess what you are searching for. It will recognise the following in your search terms, and narrow your search accordingly.

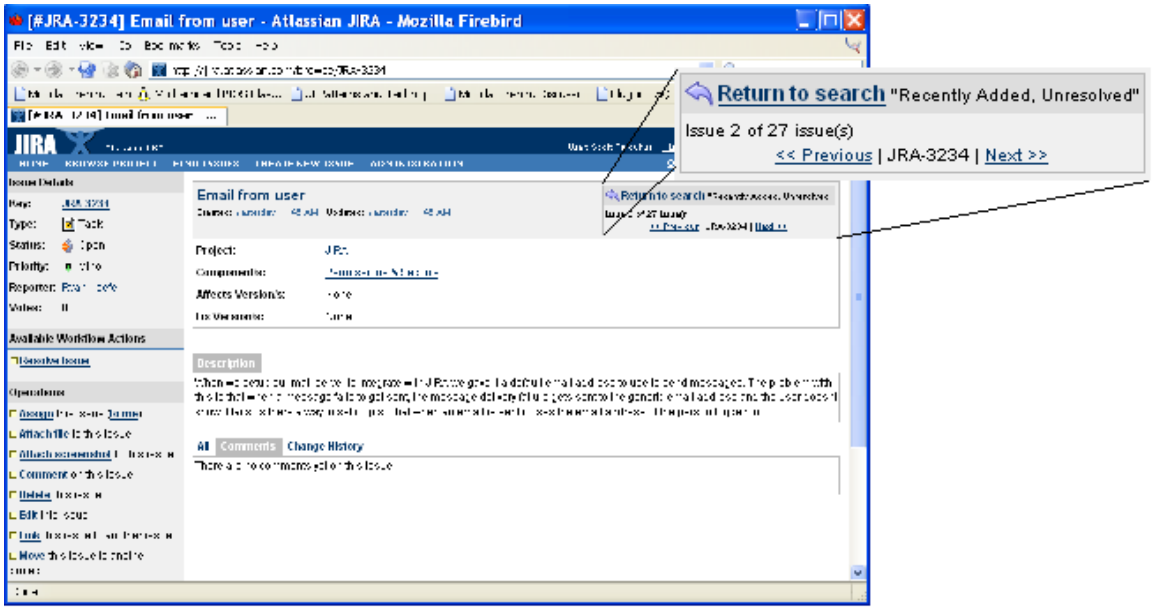
- Project keys
- Project names (single word names only)
- Issue statuses
- Issue types
- "my"

So, for example:

- "JRA open bugs" will search for all open bugs in the JIRA project.
- "my JRA open improvements" will search for all open improvements in the JIRA project that are assigned to you.

Streamlined Search Results

On the subject of searching, we've improved the site's navigation by giving you the ability to move through your list of search results without returning to the search results page. When you visit an issue from a search, the navigation box shown below will allow you to skip quickly to the next and previous results, conveniently bound to the hot-keys *ALT+N* and *ALT+P* respectively. *ALT+F* will take you back to the search results page.



Per-Search Column Ordering (Enterprise)

Users of the Enterprise version of JIRA can now have custom column-ordering for each of their saved searches (previously, there was just one column-ordering preference that applied to every search). This gives you much more power to build custom issue reports containing just the information you want to know, in the right order.

XML-RPC/SOAP interface to JIRA

JIRA now has an external programmatic API, a much-demanded feature from users who wanted a more direct way to interact with their issue-tracker, or to integrate JIRA more seamlessly with their business. It also provides a raft of new features for Mark Derricutt's Intellij IDEA JIRA Plugin. The API is available in SOAP and XML-RPC flavours, and is documented here.

You can also get an XML view of any issue by adding '?view=rss' to the end of any issue page's URL.

If you come up with some interesting use for the API, or application that takes advantage of it, let us know. Similarly, if you have any suggestions for features that would make the API more useful to you, don't hesitate to tell us.

Screenshots and Thumbnails

Our new screenshot feature makes it easy to attach screenshots to your issue: it's as simple as pasting your screenshot directly into the applet provided, and hitting enter. (Currently, this feature is only available for Windows clients)

Any image that is attached to an issue is automatically displayed as a thumbnail, giving you a much better idea of what each attachment might

contain. You can see an example of this online: JRA-2789. These thumbnails can also be configured to display in search results, for organisations that make heavy use of screenshots in issue reports.

Trackback and Confluence Integration

Confluence is the latest Atlassian product, and you can now link Confluence docs to JIRA issues and vice versa. When a JIRA issue contains a link to a Confluence page the server will automatically inform Confluence that it has been linked to, so Confluence can in turn refer back to the issue. This works both ways, of course, so when a Confluence page refers to a JIRA issue, JIRA also links back to the page. (For example, see http://jira.atlassian.com/browse/JRA-2789)

This is all done with the standard trackback API. Trackback is also widely supported by weblogging software, so you can see when people are talking about a particular issue on their blogs.

If you don't want trackback, you can easily turn it off in the administrative configuration. (By default, JIRA will receive trackback pings, but not send them)

Email Integration

JIRA's email integration is greatly improved. For people sending email to JIRA:

- JIRA can be configured to create new user accounts for anyone who sends an email to JIRA who does not already have an account (great for support!)
- HTML emails that do not contain a plaintext alternative will be converted to text
- Email attachments are automatically added as attachments to the issue

And for people receiving email from JIRA:

- Emails sent by JIRA are properly threaded, so if your email client supports threading, all notifications on a particular issue will be grouped together.
- JIRA's emails will no longer prompt mail clients to send 'vacation' messages (which previously would cause erroneous issues to be created)
- JIRA can be configured to use a different "from" address per project.
- Administrators can now send emails to a group of JIRA users from within JIRA.

Release Notes

JIRA can now produce release notes: a cleanly formatted changelog of issues that have been resolved since the previous version. You can see the 2.6 Professional release notes online, or read the documentation.

The release-notes page contains a convenient text-box that allows you to cut and paste the HTML source directly from JIRA to your website.

Page Compression

JIRA now ships with a gzip filter, which will compress pages before sending them to the web browser. In some pages, this results in a size-reduction of 90%, massively decreasing download times and bandwidth usage for JIRA installations. This is most useful for installations of JIRA on the Internet: if you enable it on your LAN, the time the server takes to compress the pages will likely be greater than the time saved by them being compressed.

It is not enabled by default, but can be enabled from the Administration pages.

Improvements

Internationalisation

- Searching in all UTF-8 languages is now supported.
- The process of translating JIRA has been improved: adding support for a new language is now as simple as dropping in the localised jar file and restarting.
- The Calendar popup window now works in your selected language.
- JIRA i18n now works on Resin 2.1.12 on linux.

Bulk Edit

- Bulk deleteof issues is now supported
- Bulk edit can set issues' fix-for version

Import
There is now a Mantis import available, and the Bugzilla import has been greatly improved.

Other Features

There are 70 new features or improvements. Take a look.

Notable Bug-fixes

There are over 110 Bug Fixes in this release. If you raised a bug, chances are that it is fixed.

Outstanding Issues

- Currently we only have internationalised files for JIRA 2.5.3, so much of the interface is only half translated. We are in the process of co-ordinating the translation of 2.6, and new translations will be made available in future 2.6 point releases.
- If you are upgrading from 2.6 RC1, mail threading may not work correctly for issues that were created before the upgrade unless you upgrade via export / import. Issues created after upgrading to the 2.6 release will thread correctly.
- The printable view for a single issue is broken: it just redirects you to the normal, less printer-friendly page.

JIRA Release Summary

This page shows the highlights of the major JIRA releases.

Current Release

For information about the latest release, please go to the Release Notes.

JIRA 3.13 — 9 September 2008

- Shareable dashboards
- Improved filter sharing
- Favourite filters and dashboards
- Restoring projects
- Editable active workflows
- Enhanced sub-task quick creation
- Personal licenses
- New plugins
- Progress bar for long-running operations
- Application improvements
- More in release notes

JIRA 3.12 — 7 December 2007

- 'Trusted' Confluence
- 'JIRA System Administrators' permission
- FishEye plugin now bundled with JIRA
- Improvements to the Subversion plugin
- Improvements to the 'Project Statistics' and 'Filter Statistic' portlets
- New post function for workflows: 'Assign to Current User'
- Enhanced language support for searching
- Visual SourceSafe plugin
- More in release notes

JIRA 3.11 — 25 September 07

- Sub-task progress shown within issues
- Issue Navigator offers sub-task aggregates
- Time Tracking reports now include sub-tasks
- Multi-project 'Road Map' portlet
- Performance improvements
- Indexing improvements
- JIRA Labels Plugin
- More in release notes

JIRA 3.10 — 9 July 2007

- Editable Worklogs
- Start Date for Worklogs
- New way to browse Components
- New way to browse Versions
- Auto-complete 'User-picker' and 'Issue-picker'
- Auto-complete 'Issue-picker'
- More in release notes

JIRA 3.9 — 8 May 2007

- Ability to convert sub-tasks to issues (and vice versa)
- Convenient new scheduler for filter subscriptions
- Separate permissions for 'Delete Comment', 'Delete Attachment' and 'Delete Issue'

- Performance Improvements for Project Roles
- More in release notes

JIRA 3.8 — 13 March 2007

- Editable comments
- Self-installer for JIRA
- CAPTCHA for new account signup
- Integration with Crowd
- Improvements to the Bugzilla importer
- DHTML-loading of Issue screens
- More in release notes

JIRA 3.7 — 18 December 2006

- Project Roles - assign users and groups to roles on a per project basis
- Chart View - view charts in Issue Navigator using the JIRA Charting plugin
- RSS Improvements
- User Properties - record arbitrary information to the user profile (admin only)
- SVN project panel plugin - provides a summary of all commits made against a particular project or a project version
- More in release notes

JIRA 3.6 — 18 April 2006

- Custom Events - extension point for notification and workflow schemes
- Group Picker Custom Field - searchable in the issue navigator
- Per-issue Group Notifications and Permissions - based on the group picker custom field
- "I'm Feeling Lucky" Quick Search
- Collapsible Fields - control the level of detail of environment, description, individual comment fields and any textarea custom field
- Nestable Conditions - construct complex workflow conditions using nested conditions with AND or OR statements
- More in release notes

JIRA 3.5 — 01 February 2006

- Bulk Workflow Transition
- FogBugz Importer
- Charting Plugin
- MS Word Export
- JIRA Page Linker Plugin - linking a JIRA issue with a Confluence URL
- Component Lead Notification Type
- Bulk Assignment of Users to Groups
- More in release notes

JIRA 3.4 — 15 November 2005

- Issue Types Per Project
- Renderers - Confluence markup in JIRA text-based fields such as description and comments
- Issue Operation Plugin
- Announcement Banner
- RSS Support Improvements - live bookmarking with supported browsers
- Change Parent of Sub-Task
- Multi-user Custom Field
- More in release notes

JIRA 3.3 — 05 August 2005

- Multiple Project Filters - execute a search across multiple projects
- Bulk Move
- User Custom Field as Notification Target
- Extended Search Capabilities - search by date range for 'Created' and 'Updated' system fields and the custom field 'Date Time'
- JIRA standalone ships with Tomcat 5.5
- More in release notes

JIRA 3.2 — 27 May 2005

- Field screens - configuration of field position and visibility for each issue operation and in Professional and Enterprise editions for each workflow transition screen
- Contextual custom fields - shared between projects and issue types
- Extended Bulk Edit Capabilities - Due Date, Reporter, Issue Security Level, Issue Type
- Improved internationalisation - Issue Constant Translations (Priorities, Statuses, Issue Types and Resolutions)
- Improved performace - quicker searching in the issue navigator and reports generation

- Smart Query
- Excel View
- More in release notes

JIRA 3.1 — 14 February 2005

- CSV Importer Wizard
- Add Comment on 'View Issue' field
- Webwork Plugin Type
- Assign Issues by Mail (via the CC field) using the Create Issue Handler
- More in release notes

JIRA 3.0 — 12 October 2004

- Workflow editor and configurable workflows
- Sub-tasks
- Plugin System
- Pluggable Custom Fields
- Dashboard Overhaul
- Issue cloning
- More in release notes

Beta Releases

A **Beta release** is a preliminary release leading up to the official release of a JIRA version. Beta releases are a snapshot of our work in progress and provide an advance preview of new features to the general public. JIRA plugin developers can also use Beta releases to test and fix their plugins in advance of an official release.

Do not use in production
Beta releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **Beta releases are not safe** — Beta releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No upgrade path** — Because Beta releases represent work in progress, we **can not** provide a supported upgrade path between Beta releases, or from any Beta to the eventual final release. Thus, any data you store in a JIRA Beta release may not be able to be migrated to a future JIRA release.

The following Beta releases are currently available for download. Your help with testing them is very appreciated! Please log the bugs you find on http://jira.atlassian.com in the "JIRA" project.

Beta Releases

- JIRA 4.0 Beta 2 Release Notes
- JIRA 4.0 Beta 1 Release Notes

JIRA 4.0 Beta 2 Release Notes

July 23, 2009

JIRA 4.0 Beta 2 is a public development release leading up to **JIRA 4.0**. For all production use and testing of JIRA, please use the latest official release.

A **Beta release** is a preliminary release leading up to the official release of a JIRA version. Beta releases are a snapshot of our work in progress and provide an advance preview of new features to the general public. JIRA plugin developers can also use Beta releases to test and fix their plugins in advance of an official release.

Do not use in production
Beta releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **Beta releases are not safe** — Beta releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No upgrade path** — Because Beta releases represent work in progress, we **can not** provide a supported upgrade path between Beta releases, or from any Beta to the eventual final release. Thus, any data you store in a JIRA Beta release may not be able to be migrated to a future JIRA release.

It is not possible to upgrade JIRA 4.0 Beta 1 data to Beta 2. The data needs to come from an already released JIRA version (for example, JIRA 3.13.5).

The **only** plugin that is compatible with JIRA 4.0 Beta is the latest JIRA Toolkit. Do not install any other plugins.

The Atlassian JIRA team is delighted to present a brand new version of one of the world's favourite issue-trackers.

Highlights of JIRA 4.0 Beta 2:

- Advanced Searching
- Dashboard Gadgets
- Activity Stream
- Issue Actions in the Issue Navigator
- Charting Now Comes Standard
- New-look "Browse Project"
- Project Icons
- New-look Header
- Default Unit for Time Tracking
- "History" is now permanent
- Engine Room
- Plus over 250 other fixes and improvements

Thank you for your interest in JIRA 4.0 Beta 2

Download Beta

Known Issues

Upgrading to JIRA 4.0
JIRA 4.0 Beta can be downloaded here. Before upgrading, please refer to the JIRA 4.0 Upgrade Guide.

Known Issues

- **Supported browsers:** For optimal experience with JIRA 4.0 Beta 2, please use Firefox 3.x. Support for other browsers will be added prior to the launch of JIRA 4.0.
- **Portlet-to-gadget upgrade task:** This has not yet been implemented. Your existing portlets will be displayed in legacy mode.

^Top

Highlights of JIRA 4.0 Beta 2

Advanced Searching

The new advanced search (JQL) provides support for logical operations (including AND/OR/NOT/NULL, even on custom fields) when filtering issues:

For more on the new JQL search syntax, please see the documentation.

^Top

Dashboard Gadgets

The new-look JIRA dashboard now uses industry-standard 'gadgets'. So you can add external gadgets to your JIRA dashboard, as well as displaying JIRA gadgets in other places (such as iGoogle).

What's happened to your favourite JIRA portlets? Don't worry: every portlet that previously shipped with JIRA has been converted to a 'legacy gadget'. And if you are a plugin developer and have created your own portlets, see the instructions for converting your portlets to gadgets.

The following gadgets are available in Beta 2:

- 'Activity Stream' gadget (see below)
- 'Admin' gadget
- 'Create Issue' gadget
- 'Favourite Filters' gadget
- 'Filter Results' gadget
- 'Introduction' gadget
- 'Issue Completed This Iteration' gadget
- 'Login' gadget
- 'Quicklinks' gadget
- 'Voted Issues' gadget

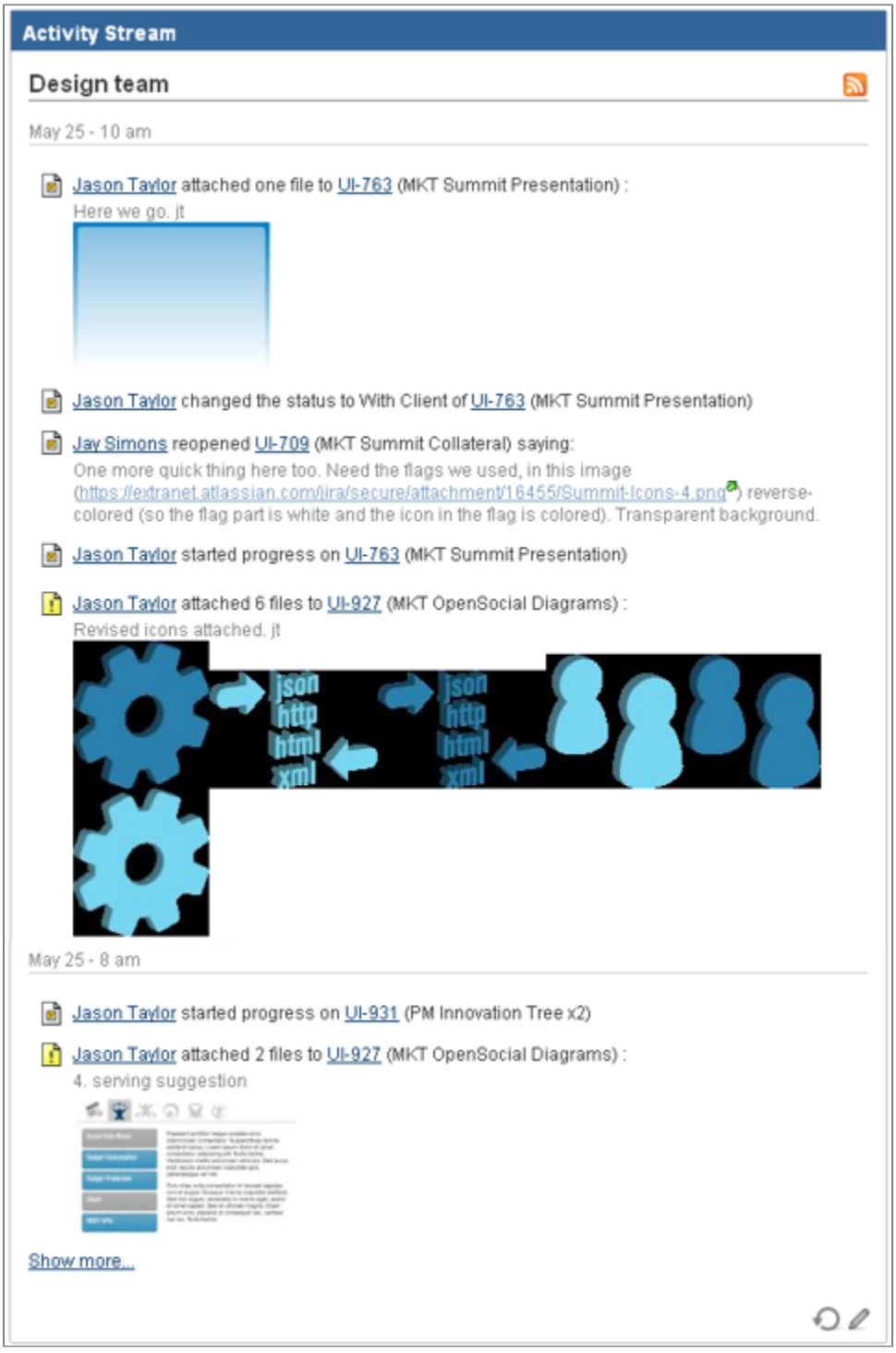
^Top

Activity Stream

The new '**Activity Stream**' gadget displays a summary of the latest activity in JIRA projects (and/or by particular people) in which you are interested.

The '**Activity Stream**' gadget also provides an RSS feed, allowing you to create very specific RSS feeds of only the information that is most relevant to you.

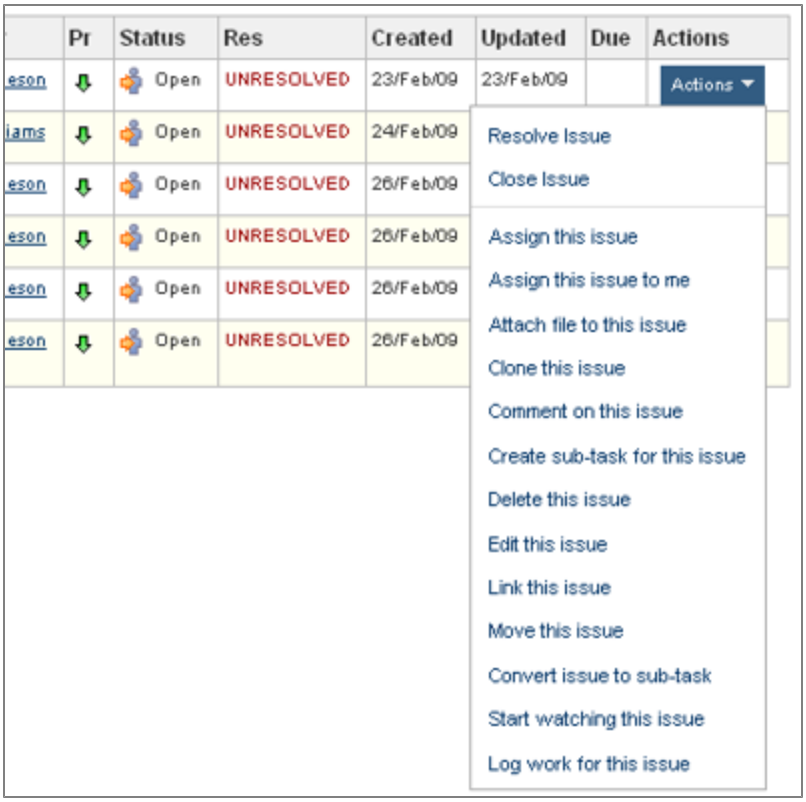
See the documentation for more details.



^Top

Issue Actions in the Issue Navigator

By popular request, issues can now be actioned directly from the Issue Navigator:

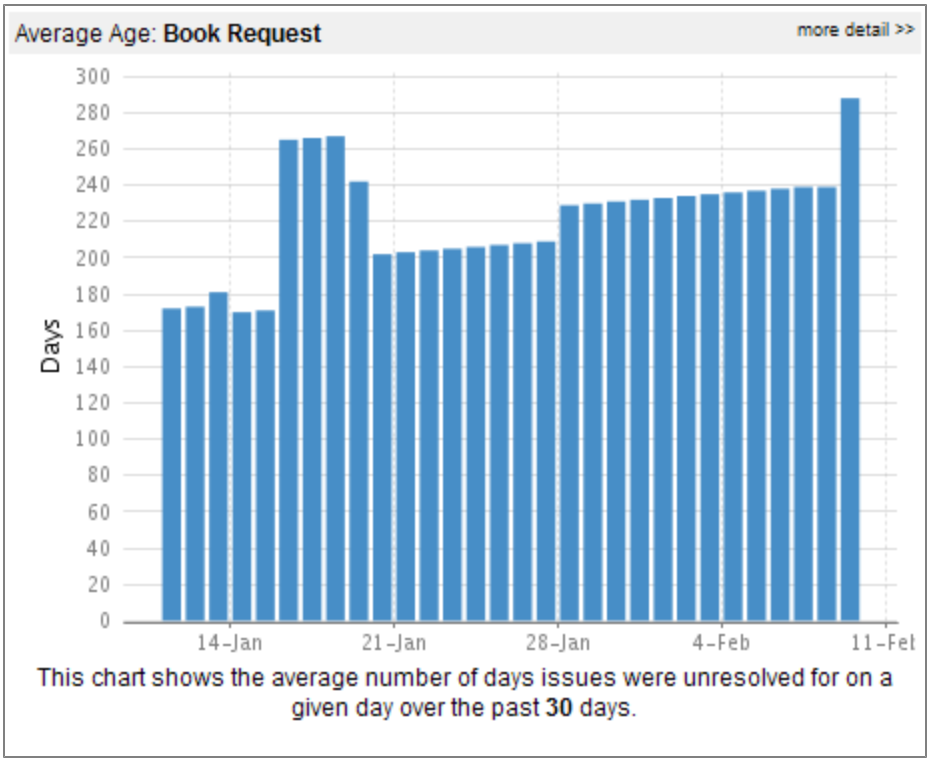


The "Actions" menu is also available for the list of sub-tasks within an issue.
^Top

Charting Now Comes Standard

The following reports and gadgets from the Charting plugin have now been integrated into JIRA:

- "Average Age" report and gadget — Shows the average age (in days) of unresolved issues, e.g.:



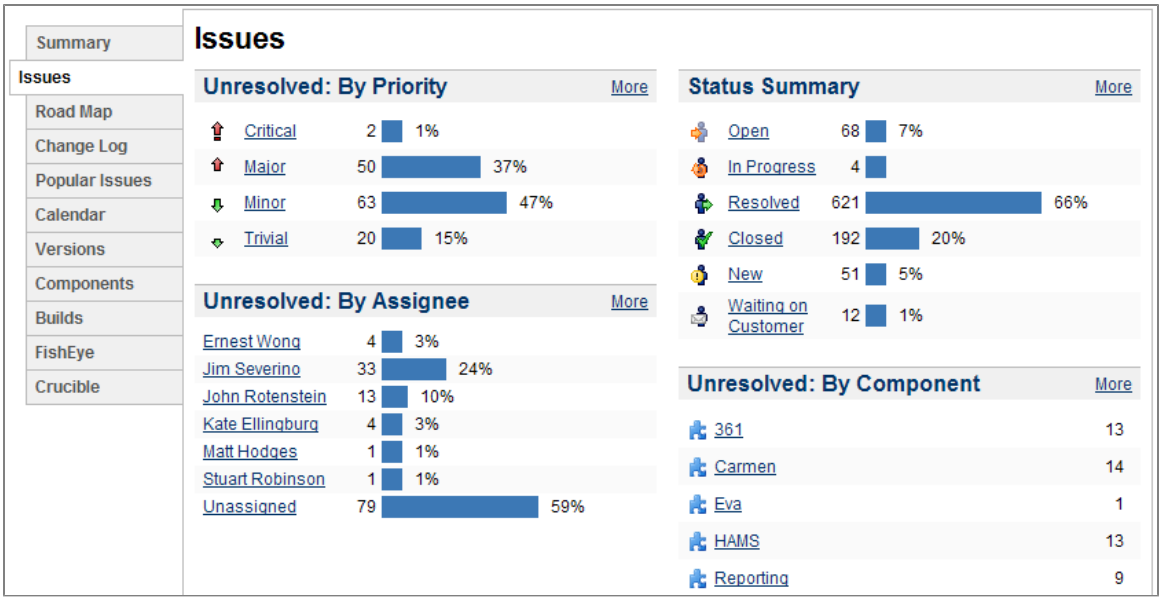
- "Created vs Resolved Issues" report and gadget — Shows the number of issues created vs number of issues resolved over a given period of time.
- "Pie Chart" report and gadget — Shows the search results from a specified issue filter (or project) in a pie-chart, based on a statistic of your choice.
- "Recently Created Issues" report and gadget — Shows the rate at which issues are being created.
- "Resolution Time" report and gadget — Shows the average time taken to resolve issues.
- "Time Since Issues" report and gadget — Shows the number of issues for which your chosen date field (e.g. 'Created') was set on a given date.

Also, the "Resolved" field from the Charting plugin is now part of JIRA, so every issue now automatically has its resolution date recorded.

^Top

New-look "Browse Project"

JIRA 4.0 provides a cleaner, more interactive view into a project:

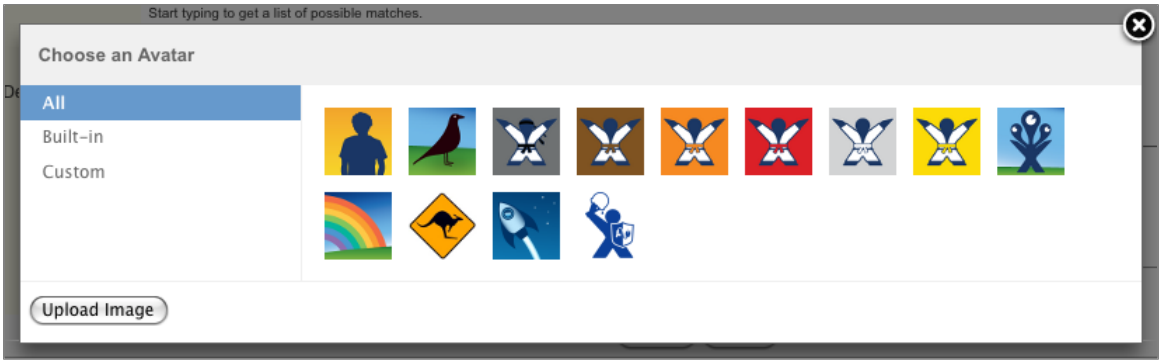


See the documentation for more about browsing projects, versions and components.
^Top

7

Project Icons

You can now give your project a visual identity, thanks to the introduction of project icons ('avatars'):

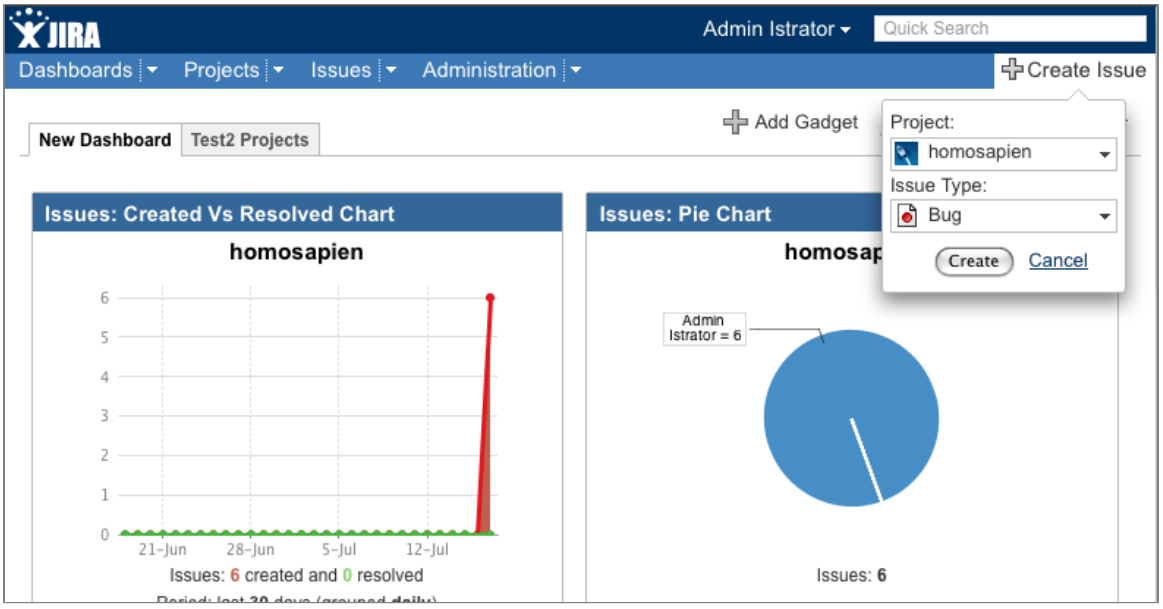


^Top

8

New-look Header

The new-look JIRA header gives you quick access to all the most commonly-used functions. Creating an issue just got super-fast!



Click to zoom in:

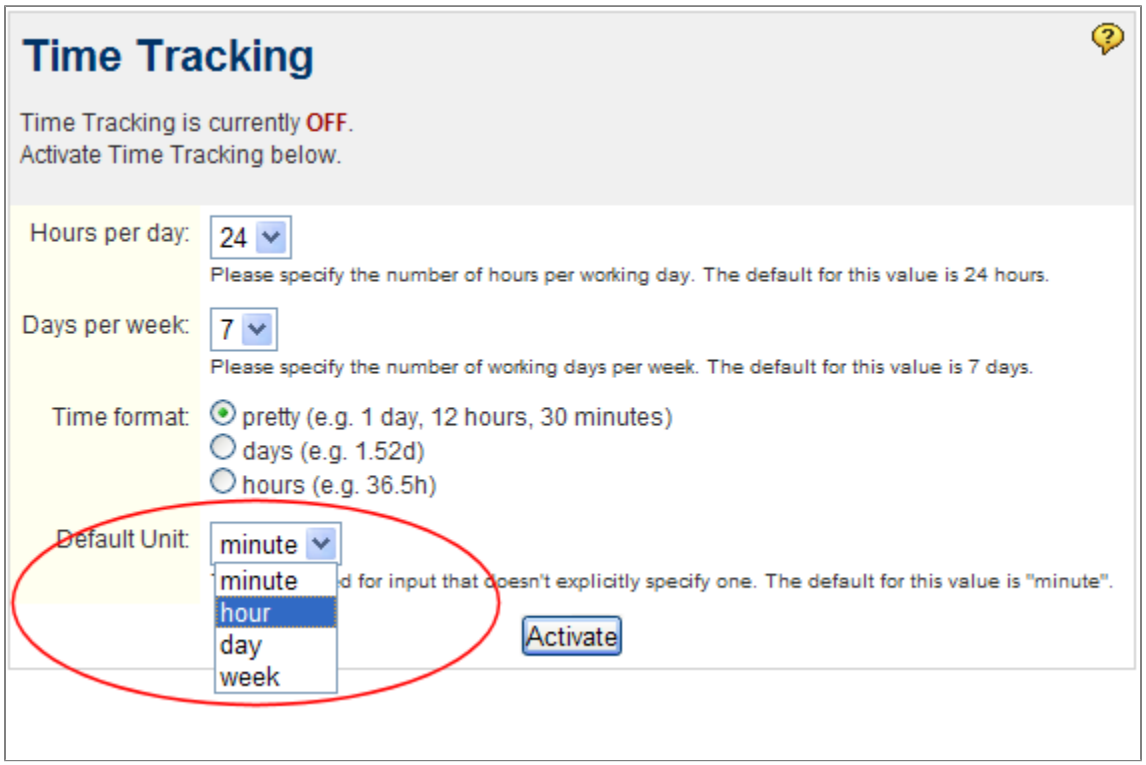
The new 'Dashboards' menu
The new 'Projects' menu
The new 'Issues' menu

The new 'Admin' menu
The new 'User' menu
The new 'Create Issue'

^Top

Default Unit for Time Tracking

You can now specify your preferred **Default Unit** (minutes/hours/days/weeks) for your JIRA system. This will be applied whenever users log work on an issue without specifying a unit.



^Top

"History" is now permanent

Your list of recently-viewed issues is now stored in JIRA's database — so it's available after you log out and back in, even if you use a different machine.

^Top

Engine Room

Beyond the 'Back' Button

When navigating away from a page where you have modified data, you will be prompted to see if you would like to save the data or discard your changes (see JRA-14911).

Index Queue

Index updates are now put in a queue. So even if the update takes longer than 30 seconds, the operation remains on the queue and is not lost. (See JRA-14220.)

^Top

Plus over 250 other fixes and improvements

Click here for full list.

JIRA Issues (200 issues)				
Key	**Type**	**Summary**	**Priority**	**Status**

JRA-923	Allow filter by "No Fix For" across projects		Resolved
JRA-1538	Filter on Versions and Components across Projects		Resolved
JRA-1560	Better support for logical operation (and/or/not) type of filters.		Resolved
JRA-1635	"not" qualifier on fields for searching		Resolved
JRA-1642	Create home directory instead of index & attachment directory		Resolved
JRA-1983	Enable filtering on "older than 1 month"		Resolved
JRA-1994	Ability to filter on time tracking related fields		Resolved
JRA-2033	Add an RSS feed query for comments to individual issues		Resolved
JRA-2469	It would be really nice to specficy several Asignee options in filters		Resolved
JRA-2607	Would like to create a filter also with OR conditions		Resolved
JRA-2681	Extend filter capablities by adding negative clauses		Resolved
JRA-2852	search for issues on version lower or equal to a given version		Resolved
JRA-2916	Allow Previous version searching		Resolved
JRA-2925	Can't filter by Security Level		Resolved
JRA-3000	Add key NUMBER (only number) searching to default search filter.		Resolved
JRA-3101	Jira - query / search / filter by issue links		Resolved
JRA-3206	View issues without an estimate		Resolved
JRA-3451	Enable filtering by Date Resolved		Resolved
JRA-3464	allow filtering by project category		Resolved
JRA-3624	released/unreleased version filter		Resolved
JRA-4227	Recent History Popup - persistance across sessions & more data		Resolved
JRA-4605	new filter criteria: add NOT to all existing criteria		Resolved
JRA-4688	Browse Project: Within the tab panel, if components are hidden - the version info appears to be right aligned.		Resolved
JRA-5121	Filter Portlet with configurable columns		Resolved
JRA-5152	Show issue linked to another issue.		Resolved

JRA-5201	Enable filter to specify more than 1 user		Resolved
JRA-5310	Watchlist should be exportable		Resolved
JRA-5383	My Votes and My Watches as filters		Resolved
JRA-5435	Issue actions and operations on Issue Navigator		Resolved
JRA-5560	Improved query functionality		Resolved
JRA-5844	allow multiple users when creating filters		Resolved
JRA-5965	Allow configure units of time tracking		Resolved
JRA-6010	Thought processing		Resolved
JRA-6180	Search for a custom field that is empty		Resolved
JRA-6344	Send to both previous and current assignees for all notifications		Resolved
JRA-6527	Allow filters to be built upon other shared filters (combined filters)		Resolved
JRA-6550	if text contains certain characters, cdata in xml based on this will be badly formed		Resolved
JRA-7068	Allow for list of issues to be saved as a filter		Resolved
JRA-7551	Provide capability to find issues by resolution date		Resolved
JRA-7626	Build search queries remotely		Resolved
JRA-7772	Ability to create advanced queries to search across all data		Resolved
JRA-7909	Search/ filter for "empty" fields		Resolved
JRA-8009	java.lang.IllegalStateException: getAttribute: Session already invalidated when trying to call getAvailableActions or getFieldsForAction		Resolved
JRA-8159	Add ability to issue navigator to find all issues linked to x issue - with option to constrain by link type		Resolved
JRA-8293	Import fails if searchrequest:request data too large		Resolved
JRA-8487	Bad logging from uk.ltd.getahead.dwr.util.CommonsLoggingOutput on startup		Resolved
JRA-8527	Put task actions directly in filter output		Resolved
JRA-8686	Allow searching of issues by Full Name for all user fields		Resolved
JRA-8758	Cannot create filter for multiple projects all issues in version "Released Versions"		Resolved
JRA-8852	Sort filter results by non-visible field		Resolved

JRA-8973		RSS of Project Changes	Resolved
JRA-9048		Calendar week begins with sunday independently from locale	Resolved
JRA-9115		Ability to search for issues with no due date associated	Resolved
JRA-9278		New Field "Resolution Date" automatically filled with date of setting resolution	Resolved
JRA-9651		User Activity Log	Resolved
JRA-9658		Minor css bug (cursor)	Resolved
JRA-9823		Allow to optionally clone an issue's attachments when cloning an issue.	Resolved
JRA-10245		Ability to filter/view Issues upon "Versions" across multiple "Projects"	Resolved
JRA-10422		Error in logs when nonexistent key used in wiki-rendered text	Resolved
JRA-10427		Changing field descriptions in "Field Configurations" for custom fields does not work	Resolved
JRA-10443		"Not Assigned to User" criteria in filters	Resolved
JRA-10492		Search for several users as Assignee or Reporter	Resolved
JRA-10603		MultipleSelect searcher for cascading selection field	Resolved
JRA-10644		Make filters more accessible	Resolved
JRA-11134		Allow setting of column order/sort with no issues in result set	Resolved
JRA-11933		AutoTransitionListener - Reopen transition deletes issue summary	Resolved
JRA-12165		Unclear error message when bulk moving issues whose reporter cannot create issues	Resolved
JRA-12200		Reporter system field throws ClassCastException after populateFromIssue() and validateFromParams()	Resolved
JRA-12525		Emails containing attachments with non-ASCII names lost	Resolved
JRA-12596		Enable cross-project filtering on special versions	Resolved
JRA-12656		Add paging/optimization for Change Log scope	Resolved
JRA-12816		OutOfMemoryError PermGen Space on Windows Func Test (under VMWare)	Resolved
JRA-12921		Ability to export Watched Issues to excel	Resolved
JRA-12976		AbstractMessageHandler might not be removing spaces from email addresses before using them to determine if a user exists when creating an issue from an email	Resolved
JRA-13011		Component of a subtask is still component of original project after moving an issue	Resolved

JRA-13035	CSV import can not import resolution date.		Resolved
JRA-13426	Next/previous version links for 'Browse Version' screen		Resolved
JRA-13625	Implicit profiling functionality broken		Resolved
JRA-13689	Saved filters reverted to "All projects" when we deleted a project contained		Resolved
JRA-13793	Confusing "The 'Project Information' panel is not available" message when fields are disabled		Resolved
JRA-13801	Call method addWorklogAndAutoAdjustRemainingEstimate, the soap server reponse with this type IssueServiceImpl$RemoteWorklogImpl		Resolved
JRA-13850	Servlet Content Listeners should implement the catch / log / rethrow pattern		Resolved
JRA-14031	Form data lost when using back and forward web browser buttons		Resolved
JRA-14220	Ensure the index optimize operation does not cause index lock timeouts		Resolved
JRA-14416	Move Issue with SubTask between differents project, IssueType and SubIssueType		Resolved
JRA-14490	Deleting project can cause filter to select all projects		Resolved
JRA-14513	JIRA Soap Service log and Access filter log footprint needs improving - Invoked Method would be handy		Resolved
JRA-14516	JIRA upgrade page should warn about possible character encoding issue if JIRA is moved between two servers.		Resolved
JRA-14598	Add access key for administer project from browse project		Resolved
JRA-14616	Ability to query for issues that you are not watching		Resolved
JRA-14654	ColorPicker for LookAndFeel page does not work on all browsers		Resolved
JRA-14701	OSPropertyText table should have the value column set to extremely-long datatype		Resolved
JRA-14747	Ability to search for issues with blockers linked to them		Resolved
JRA-14811	Deleting Group Does Not Remove Group From a Subscription		Resolved
JRA-15018	Improved SOAP and HTTP access logging		Resolved
JRA-15112	Adding Update Issue Field workflow postfunction causes OutOfMemoryError		Resolved
JRA-15254	Browse Project Panel: Do not show Closed UNRESOLVED issues as Open in the version list		Resolved
JRA-15266	Would prefer if the "Worklog" heading under "Operations" when viewing an issue were a link, like the rest of the operations.		Closed
JRA-15309	Wrong assignee drop-down list sorting with non-ascii characters		Resolved
JRA-15445	RPC plugin needs to be cleaned up		Resolved

JRA-15484	Tokenizing java exceptions fails if the exception is terminated with a full-stop.		Resolved
JRA-15517	Upgrade JIRA to use the latest version of Lucene indexing framework - v2.3.2		Resolved
JRA-15543	Show release date next to version name in the list of versions on Browse Project screen		Resolved
JRA-15545	Export issues to Excel/Word format with non-ASCII filter name does not handle the file name properly		Resolved
JRA-15546	Versions no longer display descriptions when browsing project		Resolved
JRA-15548	If an attempt to get the Index lock times out, the indexing operation is discarded		Resolved
JRA-15564	JIRA displays error after a reindex.		Resolved
JRA-15625	It is possible to disable plugins that then render jira incapable of restarting		Resolved
JRA-15631	jelly with invalid variables script returns blank page to user		Resolved
JRA-15638	The new dropdown does not appear to render correctly under IE6/7		Resolved
JRA-15646	Convert JIRA to jQuery		Resolved
JRA-15649	Sort list of plugins in Admin section alphabetically		Resolved
JRA-15665	Address issue of plugins pushing filter/report panel off screen		Resolved
JRA-15666	Add project information to the issue XML view		Resolved
JRA-15669	Drag and drop behaviour doesn't work on Modify Issues Type Scheme page		Closed
JRA-15700	Created VS Resolved cumulative + individual graphs' Y axis should should be independent		Resolved
JRA-15702	Migrate to licensing 2.0		Resolved
JRA-15723	Jelly AddComment tag changes the "updated" issue timestamp to execution script timestamp		Resolved
JRA-15761	If issue key contains unicode characters, redirect on create issue doesn't work		Resolved
JRA-15846	Allow changing license on Data Import in new JIRA instance.		Resolved
JRA-15872	"Browse Project" URL doesn't include current project		Resolved
JRA-15886	Add logging notification for index optimization events		Resolved
JRA-15962	Upgrade JIRA to Plugins 2.x		Resolved
JRA-15991	Merge translations files into one to make translating JIRA easier!		Resolved
JRA-16058	Aggressive locking in JiraCachingPropertySet causes high contention		Resolved

JRA-16074	Incorrect error warning message on navigator summary		Resolved
JRA-16080	1px offset in Firefox		Resolved
JRA-16088	Created VS Resolved Issues Report contains Old Filtter / Project Picker		Resolved
JRA-16112	Bug in progressWorkflowAction method in SOAP		Resolved
JRA-16113	Do not show negative values on Y axis in Created vs. Resolved chart		Resolved
JRA-16122	HTTP Basic auth should be enabled by default		Resolved
JRA-16138	Anonymous users should not be considered to "own" all anonymous comments.		Resolved
JRA-16151	Colon : in custom field search causes searching of wrong field		Resolved
JRA-16175	JIRA issues macro does not work with 4.0-m1 on EACJ		Resolved
JRA-16210	Display issue count on JQL execution		Resolved
JRA-16211	Enter / return should execute JQL		Resolved
JRA-16276	Adjust colours of Resoltion date chart to be more distinshable for the colour blind		Resolved
JRA-16278	Add ability to search for versions using regex or similar		Resolved
JRA-16316	Assigned To Me Portlet, selecting all columns to display causes error		Resolved
JRA-16339	The "Perm Gen" memory usage shown on the System Info page is incorrect.		Resolved
JRA-16351	Component plugin modules don't show up in plugins admin section in JIRA 4.0		Resolved
JRA-16363	ServiceProxyDestroyedException when you reactivate an OSGi plugin		Resolved
JRA-16407	JiraModuleDescriptorFactory doesn't define some plugins2 descriptors		Resolved
JRA-16424	log4j output should contain more information		Resolved
JRA-16443	Create a jira.field.resolution.include transition attribute		Resolved
JRA-16451	JIRA home directory created in working directory in JIRA standalone		Resolved
JRA-16485	The long component name, on clicking overlaps the UI element		Resolved
JRA-16498	Version/s and Component/s not validated when updating an issue		Resolved
JRA-16502	Local helper has invalid HTML, causing styling issues		Resolved
JRA-16508	no attachments are returned when 'field=attachment' is specified in XML view		Resolved

JRA-16510	Update to jQuery 1.3.2		Resolved
JRA-16522	Searching according to multiple assignees should be provided.		Resolved
JRA-16526	Catalan translations needs to be changed		Resolved
JRA-16527	Indexing fails (or atleast error thrown) if index location is pointing to an invalid reference		Resolved
JRA-16529	Project Avatars: User is unable to upload PNG and JPEG file for project avatars on IE 7.0		Resolved
JRA-16530	Project Avatars: User is unable to crop the uploaded image in the project avatars dialog		Resolved
JRA-16531	Project avatar: One image is missing from "Choose an avatar" dialog		Resolved
JRA-16532	User gets a system error if after timeout tries to login from project avatar dialog		Resolved
JRA-16560	Upgrading bunded plugins fails		Resolved
JRA-16568	Example for xmlrpc client uses a method that has been removed. Need to update		Resolved
JRA-16570	Text after a thumbnail in a comment has anchors		Resolved
JRA-16656	Go around JIRA and find all the places that are evaluating EL badly.		Resolved
JRA-16660	Project avatar dialog is not coming properly for IE 7.0		Resolved
JRA-16662	Cannot start up bundled plugins		Resolved
JRA-16664	Plugins2.2-beta12 Some UnloadablePlugins don't have a correct key causing JIRA startup to fail		Resolved
JRA-16675	Filter doesn't display errors when first loading into simple edit mode		Resolved
JRA-16685	Enormous font on random pages		Resolved
JRA-16686	DOCTYPE declaration included twice for issue related pages		Resolved
JRA-16698	Have the ability to log all SQL statements issued by JIRA and also have a callback for timing purposes		Resolved
JRA-16714	There is a bug with the way the QuerySearcher works with the "doesItFit"		Resolved
JRA-16715	JQL does not handle EMPTY embedded in lists		Resolved
JRA-16718	Fix the project clause validator to do a permission check to see if the user can see the project		Resolved
JRA-16728	Created vs Resolved: Link from Issue-Numbers in the Table to the corresponding searchrequest in the IssueNavigator		Resolved
JRA-16735	"Bulk Change" text in "Tools" drop down changes to link in certain scenerios		Resolved
JRA-16744	Improve the performance of checking if a user belongs to a particular group.		Resolved
	Fix any memory leaks in JIRA mainly caused by restoring data from XML and refreshing all singleton		

JRA-16750	objects		Resolved
JRA-16757	ThreadLocalCachingPermissionManager.getProjects() returns mutable lists straight from the cache		Resolved
JRA-16788	Velocity errors in logs: "$portletConfig.key is not a valid reference"		Resolved
JRA-16799	JIRA does not handle Plugins 1 plugins in the installed-plugins folder.		Resolved
JRA-16807	Convert Intro Portlet		Resolved
JRA-16808	Convert Quicklinks portlet		Resolved
JRA-16809	Convert Favourite Fitlers		Resolved
JRA-16811	Convert Pie Chart Portlet		Resolved
JRA-16833	Avatar Picker has layout issues while cropping		Resolved
JRA-16838	Ensure Save button is disabled after submitting on the chart popup		Resolved
JRA-16839	Add nicer dashboard tabs		Resolved
JRA-16845	Version functions should validate their arguments based on permissions		Resolved
JRA-16846	Allow for Pluggable Decorators in JIRA		Resolved
JRA-16852	Comment/resolutions/etc are posted empty if previewed		Resolved
JRA-16870	Improve multi-threaded liveness of FieldLayoutManager under load		Resolved
JRA-16903	Convert CreatedVsResolved Chart Portlet		Resolved
JRA-16904	Get Edit HTML of Component Systemfield no longer orders project components		Resolved
JRA-16905	Convert Average Age Chart		Resolved
JRA-16908	Projects are not getting created with default Project Avatar		Resolved
JRA-16909	Changing the color of priority in "Edit priority" dialog, doesn't update the color icon in "Edit priority" dialog		Resolved
JRA-16914	Convert Recently Created Portlet		Resolved
JRA-16915	SOAP and Jelly Progress Workflow calls do not take into account exisiting issue values		Resolved
JRA-16916	Convert Time Since Chart		Resolved
JRA-16920	Add filter option for subtasks of a particular issue		Resolved
JRA-16926	Implement OAuth SPI in JIRA		Resolved

^Top

JIRA 4.0 Upgrade Guide

- Upgrading from JIRA 3.13.x to 4.0
  - Scheduling the Upgrade
  - Java Version
  - JIRA Home
  - Recommended Browsers
  - 'mail.mime.decodeparameters' System Property
  - 'Resolution Date' System Field
  - Charting Plugin
  - JIRA Toolkit
  - Plugin Notes
- Upgrading from JIRA 3.12 and earlier

A **Beta release** is a preliminary release leading up to the official release of a JIRA version. Beta releases are a snapshot of our work in progress and provide an advance preview of new features to the general public. JIRA plugin developers can also use Beta releases to test and fix their plugins in advance of an official release.

Do not use in production
Beta releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **Beta releases are not safe** — Beta releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No upgrade path** — Because Beta releases represent work in progress, we **can not** provide a supported upgrade path between Beta releases, or from any Beta to the eventual final release. Thus, any data you store in a JIRA Beta release may not be able to be migrated to a future JIRA release.

The **only** plugin that is compatible with JIRA 4.0 Beta is the latest JIRA Toolkit. Do not install any other plugins.

Upgrading from JIRA 3.13.x to 4.0

Please follow the JIRA general upgrade instructions, plus note the following:

Scheduling the Upgrade

Please note that upgrading to JIRA 4.0 may take a long time, depending on the size of your instance as well as server and database performance. During the upgrade, several upgrade tasks will need to run to upgrade your data to be ready for JIRA 4.0, such as:

- Calculating a resolution date for all resolved issues in your system
- Re-indexing your issues
- Converting saved filters to use JQL
- Converting existing portlets to gadgets

Please schedule sufficient downtime time for the upgrade in your production environment. It is recommended to run an upgrade first in a test environment to see how long the upgrade will take for your data set and hardware configuration.

Java Version

If you are running JIRA under version 6 (1.6) of the Sun JRE, please ensure that you are running a point release JRE 6 (1.6) Update 10 or higher. The reasons for this are:

- JIRA 4.0 introduces a new REST plugin type based on Jersey, which will not work with JRE 6 - JRE 6 Update 3. If you are running JIRA with one of these versions of the JRE you will see the following errors:

```
java.lang.LinkageError: JAXB 2.0 API is being loaded from the bootstrap classloader,
but this RI (from bundle://16.0:3/com/sun/xml/bind/v2/model/impl/ModelBuilder.class)
needs 2.1 API. Use the endorsed directory mechanism to place jaxb-api.jar in the bootstrap
classloader. (See http://java.sun.com/j2se/1.5.0/docs/guide/standards/)
```

Note: JRE 5 (1.5) doesn't have this problem since it doesn't bundle JAXB.

- JIRA 4.0 uses Lucene v2.3, which is affected by a Sun hotspot compiler bug in JRE 6 (1.6) Update 4 and upwards (see JRA-15681). The bug is fixed in JRE 6 (1.6) Update10.

JIRA Home

JIRA 4.0 has a new directory structure — for details, please see Important Directories and Files.

Please ensure that you set the `jira.home` property as described here.

Recommended Browsers

The following browsers are recommended for use with JIRA 4:

- Internet Explorer 7 and 8
- Firefox 3.x
- Safari 4

'mail.mime.decodeparameters' System Property

The following system property must be set in order for the JIRA mail handler to work correctly with emails from RFC 2231-compliant mail clients:

```
mail.mime.decodeparameters=true
```

System properties are set in different ways depending on your application server.

'Resolution Date' System Field

JIRA 4.0 introduces a new system field, the Resolution Date. This field provides the date when an issue last entered into a 'Resolved' workflow state. When upgrading to JIRA 4.0, an upgrade task will run, calculating the Resolution Date for every resolved issue in your system. If you have a large number of issues, this may take a long time. The speed at which this upgrade task runs can be improved by ensuring that your database statistics are up to date for your changegroup and changeitem tables (to ensure the database will select the most effective query plan).

For example, on Postgres this can be done by executing the following commands:

```
jiratest=# ANALYZE changegroup;
ANALYZE
jiratest=# ANALYZE changeitem;
ANALYZE
```

JIRA's RPC interface now provides two new methods to retrieve an issue's Resolution Date:

- getResolutionDateById(String token, Long issueId) – retrieves the Resolution Date given an issue id
- getResolutionDateByKey(String token, String issueKey) – retrieves the Resolution Date given an issue key

The RemoteIssue class was left unchanged, to ensure backwards compatibility of RPC clients.

Charting Plugin

JIRA 4.0 now bundles most of the charts previously provided by the JIRA charting plugin. If you currently have the JIRA charting plugin installed (v1.4.1 or previous) in `WEB-INF/lib`, please remove it as otherwise JIRA will fail to start.

The following three charts have not been bundled with JIRA 4.0. If you are using any of the following three charts, you will need to upgrade to the Advanced JIRA charting plugin NOTE: NEED to do a new release of the charting plugin and link to it here for JIRA 4.0.:

- Time to First Response Chart
- Average Number of Times in Status Chart
- Average Time in Status Chart

JIRA Toolkit

If you are using the JIRA Toolkit, you will need to upgrade it to the latest version.

You will also need to install it in your JIRA home directory, rather than your `atlassian-jira/WEB-INF/lib/` directory as it now runs in an OSGi container. Read Managing JIRA's Plugins for more information.

Plugin Notes

JIRA 4.0 introduces several changes that may break existing plugins.

There are now two different types of plugins. Each type of plugin needs to be installed into a different directory to work. Read Managing JIRA's Plugins for more information.

If you are using a plugin that is not shipped with JIRA, the plugin may need to be updated to work with JIRA 4.0. If the plugin was written by you, please read through Updating JIRA Plugins for JIRA 4.0 and see if any of it is relevant to your plugin. If you are using a plugin written by a third party, please check with the plugin's author to see if the plugin has been tested with JIRA 4.0.

Upgrading from JIRA 3.12 and earlier

In addition to the points listed above, please read the *Upgrade Guide* for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Updating JIRA Plugins for JIRA 4.0

Plugin Developer Notes

JIRA 4.0 introduces several changes that may break existing plugins. If you are using a plugin that is not shipped with JIRA, the plugin may need to be updated to work with JIRA 4.0. If the plugin was written by you, please read through the information below and see if any of it is relevant to your plugin. If you are using a plugin written by a third party, please check with the plugin's author to see if the plugin has been tested with JIRA 4.0.

- RPC plugin
- Responses from Servlet Plugin Modules are no longer decorated
- Combined JavaScript servlet has been removed
- Project/Component/Version Tab Panel Plugins
- Issue View Plugins
- Issue Tab Panel Plugins
- Search Request View Plugins
- PortalManager and PortalPageConfiguration removed
- New Searching
- Plugging into JQL and what happened to my Custom Field Searchers
    - What is a JQL Clause?
    - Integrating with JQL
    - Integrating into the Issue Navigator
    - JIRA 3.x to 4.0 Filter Upgrade
- Converting Portlets to Gadgets

RPC plugin

A number of methods in the RPC plugin were refactored to use services provided by JIRA's core services layer. As a result they may now perform stricter validation on input data, in line with behaviour exhibited by JIRA's UI. A side effect of these changes is that method calls will now throw `RemoteValidationException` instead of `RemotePermissionException` for certain method calls. This change won't break client stubs, since all methods throw `RemoteException` which is the superclass for `RemoteValidationException` and `RemotePermissionException`. However, if client code depends on `RemotePermissionException`, it will need to be updated to expect a `RemoteException` or `RemoteValidationException`.

If you have developed custom code that uses JIRA's SOAP interface, the client code may need to be updated.

Responses from Servlet Plugin Modules are no longer decorated

The response generated by Servlet Plugin Modules served under `/plugins/servlet` will no longer be decorated by SiteMesh. This means that if you are using servlets to display contents directly in the browser, they may be missing the JIRA header and footer. If the response from your servlet needs to be decorated, you have two possible solutions:

1. The best is to convert the servlet to a Webwork Plugin Modules as this is better suited for processing requests that generate HTML responses.
2. Alternatively, add

    ```
    <meta content="decorator_name" name="decorator" />
    ```

    in the `<head>` element of your HTML response, where `decorator_name` is the name of the SiteMesh decorator that should be applied.

Combined JavaScript servlet has been removed

In JIRA 4.0 we cleaned up a lot of the JavaScript resources that are included on every page. As a result, the `combined-javascript` servlet was removed, in favour of Web Resources. This means that if your plugin defines javascript resources of the form:

```
<resource type="javascript">/path/to/my/resource.js</resource>
```

they will no longer be included. They should be replaced by Web Resources.

Project/Component/Version Tab Panel Plugins

The API for this plugin has changed. We removed the action being passed in (what were we thinking) and made it a cleaner, more consistent interface. If you have any custom Tab Panel Plugins plugins, you will need to update them to use the new interface:

```
/**
 * Unified interface for all fragment-based tab panels.
 *
 * @since v4.0
 */
public interface TabPanel<D extends AbstractTabPanelModuleDescriptor, C extends BrowseContext>
{
    /**
     * Initialize the tab panel panel with the plugins ProjectTabPanelModuleDescriptor.  This is
usually used for
     * rendering velocity views.
     *
     * @param descriptor the descriptor for this module as defined in the plugin xml descriptor.
     */
    void init(D descriptor);

    /**
     * Used to render the tab.
     *
     * @param ctx The current context the tab is rendering in.
     * @return Escaped string with the required html.
     */
    String getHtml(C ctx);

    /**
     * Determine whether or not to show this.
     *
     * @param ctx The current context the tab is rendering in.
     * @return True if the conditions are right to display tab, otherwise false.
     */
    boolean showPanel(C ctx);
}
```

The specific plugin endpoints extend this in the following manner:

```
/**
 * A Tab panel to be displayed on the Browse Project page.
 */
public interface ProjectTabPanel extends TabPanel<ProjectTabPanelModuleDescriptor, BrowseContext>

/**
 * A Tab panel to be displayed on the Browse Component page.
 */
public interface ComponentTabPanel extends TabPanel<ComponentTabPanelModuleDescriptor,
BrowseComponentContext>

/**
 * A Tab panel to be displayed on the Browse Version page.
 */
public interface VersionTabPanel extends TabPanel<VersionTabPanelModuleDescriptor,
BrowseVersionContext>
```

If you are using `WebResourceManager.requireResource("...")`, your javascript will not be loaded when your tab is loaded via AJAX. You can include it via `WebResourceManager.getStaticPluginResource()` in your actual content. Note: this will be fixed in the next beta.

Issue View Plugins

The `com.atlassian.jira.plugin.issueview.IssueView` interface has changed such that the following methods:

```
public String getContent(Issue issue, IssueViewRequestParams issueViewRequestParams);

public void writeHeaders(Issue issue, RequestHeaders requestHeaders, IssueViewRequestParams
issueViewRequestParams);
```

now take in the `IssueViewRequestParams` parameter. This allows the plugin to access the parameters that were submitted with the request.

If you have written an Issue View plugin, you will need to update it such that in conforms to the new interface.

Issue Tab Panel Plugins

In JIRA 4.0, a new 'sortable' property was introduced to distinguish if the contents of an issue tab panel are sortable. If they are not, the sortable link in the top right corner will not be shown. By default issue tab panels are now **not** sortable. To make a tab panel sortable, plugin developers will have to add the following attribute:

```
<issue-tabpanel key="all-tabpanel" i18n-name-key="admin.issue.tabpanels.plugin.all.name" name="All Tab
Panel" class="com.atlassian.jira.issue.tabpanels.AllTabPanel">
        <description key="admin.issue.tabpanels.plugin.all.desc">Display all tab panels as
one</description>
        <label key="viewissue.tabs.all">All</label>
        <order>0</order>
        <sortable>true</sortable>
    </issue-tabpanel>
```

Search Request View Plugins

In JIRA 4.0, the `com.atlassian.jira.plugin.searchrequestview.SearchRequestView` has the following new method:

```
/**
 * Prints the HTML headers for non-typical HTML such as Word or Excel views. (e.g.:
requestHeaders.addHeader("content-disposition", "attachment;filename="sample.doc";");)
 *
 * @param searchRequest the original search request submitted by the user
 * @param requestHeaders subset of HttpServletResponse responsible for setting headers only
 * @param searchRequestParams context about the current search request
 */
public void writeHeaders(SearchRequest searchRequest, RequestHeaders requestHeaders,
SearchRequestParams searchRequestParams);
```

If you have written a Search Request View Plugin, and the plugin implements the interface without extending `com.atlassian.jira.plugin.searchrequestview.AbstractSearchRequestView`, you will need to update the plugin and implement the new method. The easiest thing to do is to proxy the call straight to the existing method:

```
/**
 * Prints the HTML headers for non-typical HTML such as Word or Excel views. (e.g.:
requestHeaders.addHeader("content-disposition", "attachment;filename="sample.doc";");)
 *
 * @deprecated since v3.13.3 please use {@link
#writeHeaders(com.atlassian.jira.issue.search.SearchRequest, RequestHeaders, SearchRequestParams)}
 * @param searchRequest the original search request submitted by the user
 * @param requestHeaders subset of HttpServletResponse responsible for setting headers only
 */
public void writeHeaders(SearchRequest searchRequest, RequestHeaders requestHeaders);
```

Note that the `SearchRequestParams` object used by Search Request View Plugins now extends `IssueViewRequestParams` and therefore allows the plugin to access request parameters.

PortalManager and PortalPageConfiguration removed

The deprecated components `PortalManager` and `PortalPageConfiguration` have been removed. Developers should now be using the `PortalPageService` (and maybe the `PortalPageManager`) and `PortalPage` to obtain similar functionality.

The `PortalPageConfiguration` had methods that made changes directly to the database (e.g. `store`, `addPortletConfig`, `deletePortletConfig`, `deletePortletConfigs`, `reload`). The `PortalPage` does not have such methods. All persistent changes must now be made through the `PortalPageService` passing the required `PortalPage` as an argument. For example, the call to `PortalPageConfiguration.deletePortletConfig` should be replaced with a call to `PortalPageService.deletePortalPage` passing in the desired `PortalPage`.

The `PortalPageManager` may also be used to manipulate a `PortalPage` within JIRA. The manager should only be used where there is no equivalent method on the `PortalPageService`. The manager is a lower level JIRA object that does not handle errors as well as the `PortalPageService`.

New Searching

The way a search is performed in JIRA has significantly changed. The introduction of advanced searching (JQL) necessitated a rewrite of the JIRA searching subsystem. In the process, the API for searching has also been changed (and improved) significantly. Unfortunately these changes will almost certainly mean that plugins that search will need to be updated for JIRA 4.0.

In JIRA 3.x and earlier, searching was achieved using a `SearchRequest` in combination with `SearchParameters` and `SearchSorts`. While the `SearchRequest` still continues to exist in JIRA 4.0, the `SearchParameters` have been replaced with the `Query` object.

```
/**
 * The representation of a query.
 *
 */
public interface Query
{
    /**
     * @return the main clause of the search which can be any number of nested clauses that will make
up the full
     * search query. Null indicates that no where clause is available and all issues should be
returned.
     */
    Clause getWhereClause();

    /**
     * @return the sorting portion of the search which can be any number of
     * {@link com.atlassian.query.order.SearchSort}s that will make up the full order by clause. Null
indicates that
     * no order by clause has been entered and we will not sort the query, empty sorts will cause the
default
     * sorts to be used.
     */
    OrderBy getOrderByClause();

    /**
     * @return the original query string that the user inputted into the system. If not provided, will
return null.
     */
    String getQueryString();
}
```

The `Query` is JIRA's internal representation of a JQL search. It contains the search condition (i.e. the "where" clause) and the search order (i.e. the "order by" clause). The `Query` object can be created using the `JqlQueryBuilder`. For example, to create a query "find all issues assigned to either Dylan or Tokes that are unresolved and due in the next week" you would call:

```
final JqlQueryBuilder builder = JqlQueryBuilder.newBuilder();
  builder.where().assignee().in("Dylan", "Tokes").and().unresolved().and().due().lt().string("+1w");
  builder.orderBy().dueDate(SortOrder.ASC);
  Query query = builder.buildQuery();
```

Once the `Query` has been obtained, it can be used to execute a search. In JIRA 4.0 a new `SearchService` has been added to provide a central location for `Query` related operations. To run the search you can simply call `SearchService.search()` as documented on the `SearchService`. The `SearchProvider` is still available for those who need to control the finer details of searching.

The `Query` object is immutable; once it is created it cannot be changed. The `JqlQueryBuilder` represents the mutable version of a `Query` object. The `JqlQueryBuilder` can be primed with an already existing `Query` by calling `JqlQueryBuilder.newBuilder(existingQuery)`.

In JIRA 3.x the `SearchRequest` was the object that was passed to the searching system to perform a search. The `Query` object has taken over this role in JIRA 4.0; the `SearchProvider` and `SearchService` now take in `Query` objects rather than `SearchRequests`. The `SearchRequest` object has been reworked in JIRA 4.0 to significantly reduce its responsibility. For instance, ordering information is now stored on the `Query` object rather than on the `SearchRequest` object. The `SearchRequest` really represents a saved search (aka. filter). You should only need to deal with `SearchRequests` if you are working with filters. Even in this case, all searching operations need to be performed on `Query` objects by calling `SearchRequest.getQuery()`.

It is often necessary to get a URL for a particular `Query`. The `SearchService` provides the `getQueryString(query)` method for this. The method returns a parameter snippet of the form `jqlQuery=<jqlUrlEncodedQuery>`, which can be appended safely to an existing URL that points at the Issue Navigator. Note that the links that JIRA 4.0 generates are JQL based, so are incompatible with JIRA 3.x and before. Old valid JIRA 3.x URLs will still work with JIRA 4.0.

Given a `Query` object it is possible to retrieve its JQL representation by calling either `getGeneratedJqlString(query)` or `getJqlString(query)` on the SearchService. The service makes sure that any values in the `Query` that need to be escaped are handled correctly. Importantly, the `Query.toString()` method does not return valid JQL (on purpose).

The `SearchService.parseQuery(jqlString)` method can be used to turn a JQL string into its `Query` representation. The return from this method has details on any parse errors encountered.

A `Query` object, especially those parsed directly from the user, may not be valid. For example, the user may be trying to find issues in a status that does not exist. The `SearchService.validateQuery(query)` method can be used to see if a particular `Query` object is valid. Errors are returned with messages that can be displayed to the user. Executing an invalid `Query` will not result in any errors and in fact may return results. To run an invalid query, JIRA will just make the invalid conditions equate to false and run the query. For example, searching for `status = "I don't Exist" or user = bbain` will result in the query `<false> or user = bbain` actually being run.

There are some methods on the `SearchService` that we did not discuss here. Check out documentation on the `SearchService` for more information.

Plugging into JQL and what happened to my Custom Field Searchers

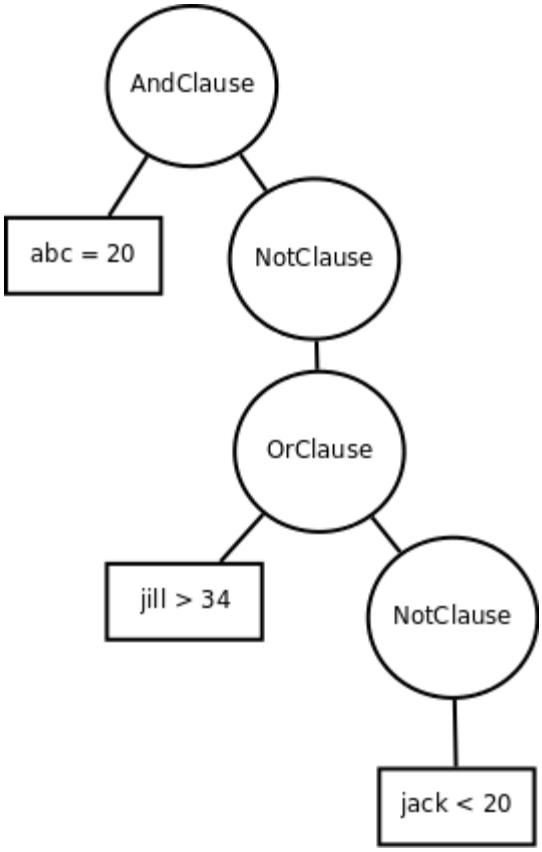
The introduction of advanced searching (JQL) necessitated a rewrite of the JIRA searching subsystem. Unfortunately these changes will certainly mean that any `CustomFieldSearchers` will need to be updated to work in 4.0.

The most fundamental change is that all JIRA 4.0 searching is implemented using JQL. A JQL search consists of two components: firstly, a number of conditions, or `Clauses`, that must be matched for an issue to be returned; and secondly, a collection of search orderings that define the order in which the issues should be returned. The `Query` object is JIRA's internal representation of a search. It is now the responsibility of the `CustomFieldSearcher` to take a relevant `Query`, validate its correctness and generate a Lucene query to find issues that match it. By doing this your custom field becomes searchable using JQL.

The `CustomFieldSearcher` and/or the custom field is also responsible for ordering results if the order in the search includes the custom field. If your custom field ordered correctly in JIRA 3.x, then it will order correctly in JIRA 4.0. While the internal representation of an order has changed in JIRA 4.0, it still uses the same interfaces to order the search results. We will not address ordering again.

What is a JQL Clause?

A custom field must process the `Clauses` from a JQL search to integrate into JQL. Each `Clause` consists of a number of conditions (e.g. `abc != 20`) combined by the AND and OR logical operators (e.g. `abc = 20 AND (jack < 20 OR jill > 34`). In JIRA a condition is represented by a `TerminalClause`, the logical AND by an `AndClause` and a logical OR by an `OrClause`, all of which implement the `Clause` interface. Finally, the logical NOT operator can be used to negate any other `Clause`. It is represented by a `NotClause` that also implements Clause. These `Clause` objects are composed together to represent a complex conditions. For example, the condition `abc = 20 AND NOT(jill > 34 OR NOT jack < 20)` is represented by the following tree:



A `Clause` can be navigated by passing an instance of a `ClauseVisitor` to the `accept` method of a Clause. This follows the traditional visitor pattern.

The `TerminalClause` represents a `Clause` of the form "field operator value". Inside the `TermincalClause` the "operator" is one of the values from `Operator` enumeration while the "value" is represented as an `Operand`. An `Operand` can represent a single value (e.g. `field = "single"`), a list of values (e.g. `field in ("one", 1235)`), a function (e.g. `field = function(arg1, arg2)`) or EMPTY (e.g. `field is EMPTY`). In the end, all you want is the values from the `Operand`. These can be obtained as a list of `QueryLiteral` (see below) by calling `JqlOperandResolver.getValues()`. The `JqlOperandResolver` also has the `isEmptyOperand`, `isFunctionOperand`, `isListOperand` and `isValidOperand` methods that can be used to determine the type of the `Operand`.

A `QueryLiteral` represents either a `String`, `Long` or `EMPTY` value. These three represent JQL's distinguishable types. It is up to the `CustomFieldSearcher` to convert these values into something that makes sense to it. The type of a `QueryLiteral` can be determined by calling its `isEmpty`, `getLongValue` or `getStringValue` methods. The get methods will return `null` or `false` when the method and the `QueryLiteral` type do not match.

Integrating with JQL

In JIRA 3.x a `CustomFieldSearcher` was the way to provide customised searching functionality for custom fields. In JIRA 4.0 it is still the plugin point for searching; however, the `CustomFieldSearcher` interface has changed significantly to accommodate the introduction of JQL. One of the major changes is that the `CustomFieldSearcher` must return a `CustomFieldSearcherClauseHandler` in JIRA 4.0. This object is a composition of a `ClauseValidator` and a `ClauseQueryFactory`.

The `ClauseValidator` is used by JIRA to ensure that a JQL query is valid according to the `CustomFieldSearcher`.

```
/**
 * Validates a clause and adds human readable i18n'ed messages if there is a problem.
 *
 * @since v4.0
 */
public interface ClauseValidator
{
    /**
     * Validates a clause and adds human readable i18n'ed messages if there is a problem.
     *
     * @param searcher the user who is executing the search.
     * @param terminalClause the clause to validate.
     *
     * @return an MessageSet that will contain any messages relating to failed validation. An empty
message set must
     * be returned to indicate there were no errors. null can never be returned.
     */
    @NotNull
    MessageSet validate(User searcher, TerminalClause terminalClause);
}
```

It is up to the validator to ensure that the operator and the value from the passed `TerminalClause` makes sense for the `CustomFieldSearcher` and its associated custom field. Any errors can be placed in the returned `MessageSet`. They should be internationalised with respect to the passed user.

The `ClauseValidator` does **not** need to check if the passed `TerminalClause` is meant for the for it, JIRA makes sure that it only passes `TerminalClauses` that the `ClauseValidator` is meant to process. It does that by only passing `TerminalClauses` whose "field" matches one of the names the custom field must handle.

The `ClauseQueryFactory` is used by JIRA to generate the Lucene search for a JQL Clause.

```
public interface ClauseQueryFactory
{
    /**
     * Generates a lucene query for the passed {@link TerminalClause}.....
     *
     * @param searcher the user who is performing the search
     * @param terminalClause the clause for which this factory is generating a query.
     * @return QueryFactoryResult contains the query that lucene can use to search and metadata about
the query. Null
     *  cannot be returned.
     */
    @NotNull
    QueryFactoryResult getQuery(User searcher, TerminalClause terminalClause);
}
```

It is the responsibility of the `ClauseQueryFactory` to create the Lucene search for the passed `TerminalClause`. The generated Lucene search is returned in the `QueryFactoryResult`. The result contains the search (a Lucene Query object which is not related the the JQL Query object) and a flag to indicate whether or not the Lucene search should be negated. When set to true, JIRA will actually only match issues that do not match the returned Lucene search. For example, a `ClauseQueryFactory` may decide to implement a condition like `field != value` by returning a Lucene search that matches `field = value` and setting the flag to true. You can also implement this condition by returning a Lucene search that matches `field != value` and setting the flag to false.

The `ClauseQueryFactory` does **not** need to check if the passed `ClauseQueryFactory` is meant for it; JIRA makes sure that it only passes `TerminalClauses` that the `ClauseQueryFactory` is meant to process. It does that by only passing `TerminalClauses` whose "field" matches one of the JQL names the custom field must handle. Put simply, the `ClauseQueryFactory` must handle any passed `TerminalClause`.

The `ClauseQueryFactory` must also handle the situation when an invalid `TerminalClause` is passed to it. An invalid `TerminalClause` is one whose associated `ClauseValidator` would not validate. The `ClauseQueryFactory` must return an empty Lucene search if the passed `TerminalClause` is invalid. Most importantly, the `ClauseQueryFactory` must not throw an exception on an invalid `TerminalClause`.

A `ClauseFactory` needs to be careful when implementing any of the negating operators (i.e. !=, !~, "not in"). These operators should not match what is considered empty by the custom field and `CustomFieldSearcher`. For example, the JIRA query `resolution is EMPTY` will return all unresolved issues in JIRA. The query `resolution != fixed` will only return all resolved issues that have not been resolved as "fixed", that is, it will not return any unresolved issues. The user has to enter the query `resolution != fixed or resolution is EMPTY` to find all issues that are either unresolved or not resolved as "fixed".

There are some extra interfaces that the `CustomFieldSearcherClauseHandler` may also implement to provide optional functionality to the searching subsystem:

- ValueGeneratingClauseHandler: Gives the `CustomFieldSearcher` the ability to suggest some values during JQL entry auto-complete. This is really only useful for custom fields whose values come from an allowable finite set.
- CustomFieldClauseSanitiserHandler: Gives the `CustomFieldSearcher` the ability pre-process the query and remove sensitive information from the query before it is displayed to the passed user.
- CustomFieldClauseContextHandler: Gives the `CustomFieldSearcher` the ability to customise JIRA's query context calculation. This interface is best left alone, unexplained and unimplemented.

Integrating into the Issue Navigator

The good old Issue Navigator still exists. The Issue Navigator actually has two modes: simple and advanced. The simple mode is what was considered the Issue Navigator in JIRA 3.x. Each searcher on the simple Issue Navigator represents a `Clause`. For example, selecting "JIRA" in the project searcher produces the `Clause` `project = JIRA`. Using multiple searchers is achieved by ANDing the multiple implied `Clauses` together. In this way the simple Issue Navigator actually generates JQL.

The advanced mode shows the raw JQL to the user. It allows a user to search by entering arbitrary JQL. Since it simply shows JQL, it is possible to create a query using the simple Issue Navigator and then view it in the advanced Issue Navigator. However, it may not always be possible to go from the advanced Issue Navigator to the simple Issue Navigator, as the simple view only allows a very limited set of JQL. A JIRA user will be able to move from the advanced to the simple Issue Navigator when the current JQL can be represented in the simple view. JIRA will stop a user from transitioning from the advanced to the simple Issue Navigator when the JQL is just too complicated to represent correctly.

The `CustomFieldSearcher` itself is still responsible for integrating into the Issue Navigator. The `CustomFieldSearcher` extends from the `IssueSearcher`, which has undergone major cosmetic surgery in JIRA 4.0. The main change is that the methods on the `IssueSearcher` have been relocated to new interfaces that the `IssueSearcher` composes. For example, JIRA 3.x used to call `issueSeacher.getEditHtml()` to get the searcher's HTML but now in 4.0 it calls `issueSeacher.getSearchRenderer().getEditHtml()`. The following table shows a summary of all the changes:

Old Searcher Method	New Seacher Interface	New Seacher Method	Comments
`getEditHtml`	`SearchRenderer`	`getEditHtml`	
`getViewHtml`	`SearchRenderer`	`getEditHtml`	
`isShown`	`SearchRenderer`	`isShown`	
`isRelevantForSearchRequest`	`SearchRenderer`	`isRelevantForQuery`	See description below.
`getId`	`SearcherInformation`	`getId`	
`getNameKey`	`SearcherInformation`	`getNameKey`	
	`SearcherInformation`	`getField`	Added in JIRA 4.0. See description below.
`getRelatedIndexers`	`SearcherInformation`	`getRelatedIndexers`	
	`SearcherInformation`	`getSearcherGroupType`	Added in JIRA 4.0. See description below.
`populateFromParams`	`SearchInputTransformer`	`populateFromParams`	Inserted a new `User` parameter as the first argument.
`validateParams`	`SearchInputTransformer`	`validateParams`	Inserted a new `User` parameter as the first argument.
`populateFromSearchRequest`	`SearchInputTransformer`	`populateFromQuery`	Changed the method name and arguments to work with JQL. See the discussion below.
	`SearchInputTransformer`	`doRelevantClausesFitFilterForm`	Added in JIRA 4.0. See discussion below.
`populateSearchRequest`	`SearchInputTransformer`	`getSearchClause`	Changed the method name and arguments to work with JQL. See the discussion below.
`register`			Removed as it is no longer necessary.

getQuerySnippet			Removed as it is no longer necessary.
getStringValue			Removed as it is no longer necessary.
getName			Removed as it is no longer necessary.

The SearcherRender interface groups together the rendering related IssueSearcher actions. The new method isRelevantForQuery takes over the role from the isRelevantForSearchRequest method. Its job is to take a complete Query object and determine if the CustomFieldSearcher is relevant for that Query. The result is used to decide if the HTML from the getViewHtml is included on some JIRA pages. As a general rule, this essentially involves walking the Query and looking for TerminalClauses related to the CustomFieldSearcher. For example:

```
final NamedTerminalClauseCollectingVisitor clauseVisitor = new
NamedTerminalClauseCollectingVisitor(clauseNames.getJqlFieldNames());
if (query != null && query.getWhereClause() != null)
{
    query.getWhereClause().accept(clauseVisitor);
}
return clauseVisitor.containsNamedClause();
```

This code essentially walks the tree looking for all TerminalClauses that have a particular set of names. The Query is relevant if such a Clause exists or is not relevant otherwise.

The isRelevantForQuery method is only called if the passed Query fits in the simple Issue Navigator.

The SearcherInformation interface groups together methods that return data about the IssueSeacher into a single interface. The SearcherInformation.getField method simply returns the Field associated with the searcher. This information is available to the searcher once the CustomFieldSearcher.init() is called by JIRA.

SearcherInformation.getSearcherGroupType is a method that returns the group the searcher should be seen in on the navigator. The custom field has to return SearcherGroupType.CUSTOM. JIRA will always force this value even if it is specified as something different.

The SearchInputTransformer interface groups together those methods on the IssueSearcher that convert Query objects into different forms so that they can be displayed and manipulated using the simple Issue Navigator. The simple Issue Navigator does not have the ability to represent all possible JQL queries. The SearchInputTransformer.doRelevantClausesFitFilterForm method allows JIRA to ask the CustomFieldSearcher if the passed Query can be represented in the simple Issue Navigator. This is used by JIRA to stop people trying to view complex JQL in the simple Issue Navigator. When this call is made, the CustomFieldSearcher must decide if the relevant sections of the passed Query can be represented in the simple Issue Navigator form. Irrelevant Clauses (i.e. those Clauses unrelated to the Searcher) should be ignored. The method must return true when the Query is not at all relevant. This method is normally implemented by walking the Query and checking that any relevant TerminalClauses are connected via the correct set of logical conditions. For example, here is some common code encountered with JIRA's internal searchers:

```
if (query != null && query.getWhereClause() != null)
{
    final Clause whereClause = query.getWhereClause();
    final SimpleNavigatorCollectorVisitor collector = new
SimpleNavigatorCollectorVisitor(clauseNames.getJqlFieldNames());
    whereClause.accept(collector);
    if (!collector.isValid() || collector.getClauses().size() > 1)
    {
        return false;
    }
    else if (collector.getClauses().size() == 1)
    {
        final TerminalClause terminalClause = collector.getClauses().get(0);
        return checkOperator(terminalClause.getOperator()) &&
checkOperand(terminalClause.getOperand(), true);
    }
}
return true;
```

The code starts by creating a ClauseVisitor that will find all the TerminalClauses with particular names. This visitor will also detect whether or not the all paths from the root Clause of the tree to the TerminalClauses are only through AndClauses. This check is made to ensure that

these `TerminalClauses` form part of a simple AND expression since the simple Issue Navigator can only support AND operators between Clauses. The code also ensures that only one `TerminalClause` is found since this is what the `CustomFieldSearcher` generates for the simple Issue Navigator. Note that the method will return true if no relevant `TerminalClauses` are found.

The new `SearchInputTransformer.populateFromQuery` method replaces the old `populateFromSearchRequest`. It essentially takes the passed `Query` and serialises the relevant parts into their associated `FieldValuesHolder` representation. It is up to this `CustomFieldSearcher` to work out which parts of the `Query` are relevant to it. It must ignore those parts of the `Query` that it was not designed to handle. This method will only be called if it is known that the `Query` fits in into the simple Issue Navigator. It is generally implemented by walking the tree and looking for the relevant `TerminalClauses` and subsequently serialising them into the passed `FieldValuesHoldler`. For example:

```
if (query.getWhereClause() != null)
{
    final ClauseVisitor visitor = new DateSerializer();
    query.getWhereClause().accept(visitor);
    fieldValuesHolder.put(dateSeacherConfig.getPreviousField(), visitor.getPreviousDate());
    fieldValuesHolder.put(dateSeacherConfig.getNextField(), visitor.getPreviousDate());
}
```

In this example we used a `ClauseVisitor` that walks the `Query` and calculates the parameters for a date-based searcher. Once the visitor is run, we simply add the calculated parameters to the `FieldValuesHolder`.

The `SearchInputTransformer.getSearchClause` method replaces the old `populateSearchRequest`. Its job it to take a take the relevant values from the `FieldValuesHolder` and generate a `Clause` for them. This `Clause` will be combined with the `Clauses` from other active searchers using the AND operator to produce the final Query on the simple Issue Navigator. The irrelevant values from the `FieldValueHolder` must be ignored. This method is generally called after JIRA has called `SearchInputTransformer.populateFromParams` with the web parameters returned from the filter form, that is, this method is how the filter form is converted into a `Clause` and subsequently a Query. Consider the following example:

```
final Clause relativeClause = createPeriodClause((String)
fieldValuesHolder.get(dateSearcherConfig.getPreviousField()),
    (String) fieldValuesHolder.get(dateSearcherConfig.getNextField()));
final Clause absoluteClause = createDateClause((String)
fieldValuesHolder.get(dateSearcherConfig.getAfterField()),
    (String) fieldValuesHolder.get(dateSearcherConfig.getBeforeField()));
return createCompoundClause(relativeClause, absoluteClause);
```

This example demonstrates how a date field looks in the `FieldValueHolder` for its relevant properties and uses them to create a `Clause`. This example also shows that the returned `Clause` can be as complex as the `CustomFieldSearcher` wants.

The `SearchInputTransformer.getSearchClause` and `SearchInputTransformer.populateFromQuery` really form a pair. The `Clause` returned from `SearchInputTransformer.getSearchClause` must be correctly processed by `SearchInputTransformer.populateFromQuery`. If this does not occur, then it would be possible to generate a query in the simple Issue Navigator view that cannot actually be viewed in it. This also implies passing the `Clause` object returned from `SearchInputTransformer.getSearchClause` to the `SearchInputTransformer.doRelevantClausesFitFilterForm` must return true.

JIRA 3.x to 4.0 Filter Upgrade

In JIRA 3.x saved seaches (aka. filters) were stored in the database as XML. In JIRA 4.0, all searchers are stored directly as JQL. An upgrade task has been written to convert 3.x filters into JQL. Unfortunately, there is no way for plugin developers to integrate into this upgrade task. This essentially means that the upgrade may fail if you have a custom `SearchParameter` or use an existing `SearchPamater` in an unorthodox way. JIRA will inform users through e-mail if any of their filters could not be upgraded cleanly. The administrator is also made aware of any problems through JIRA's log files.

Converting Portlets to Gadgets

JIRA 4.0 introduces a new dashboard based on the OpenSocial specification. Legacy portlets will still be supported, but they will miss out on a lot of new features (e.g. displaying the gadget on iGoogle). As such you may wish to convert your plugin's portlets to gadgets. To do so please follow the documentation available in the Gadget Development Hub, as well as the instructions for writing a plugin upgrade task to convert any portlet settings that users may have saved.

Writing a Plugin Upgrade Task for JIRA 4.0

Overview

JIRA 4.0 will introduce a new dashboard, effectively making the Portlet Plugin Module obsolete. Legacy portlets will still be supported via a Legacy Gadget bridge; however, they will miss out on a lot of the new features that gadgets offer (e.g. the ability to share gadgets with other apps such as iGoogle). It therefore makes sense to convert portlets over to gadgets. Information about how to write a gadget can be found in the

Gadget Development Hub, and specifically the page about gadgets and JIRA portlets.

If you've converted a portlet to a gadget, you will most likely need an upgrade task to convert existing data of your users into the new format used by the gadget you have written. This page describes the process of creating such an upgrade task.

Why an upgrade task?

Portlets generally have some configuration data associated with them by their users. For example, the First Response Time chart portlet, available in the Charting Plugin, allows users to configure how many days previous to draw the chart for (among other things). For efficiency reasons, gadgets do not use the same storage mechanism as portlets do to store these user preferences. An upgrade task is thus needed to convert existing user data over to the new format required by the new gadget.

Upgrade framework

JIRA 4.0 introduces a new plugin framework (version 2.2 or later of the Atlassian Plugin Framework, affectionately known as 'Plugins2'), which provides an events system that lets plugins register to listen for certain events (such as a 'Framework started' event). JIRA 4.0 also bundles SAL, which already includes a plugin upgrade framework. SAL provides a plugin upgrade manager that listens for the 'Framework started' event and will look for Plugin Upgrade Tasks to run in order to upgrade data for plugins.

What does all this mean? Effectively, plugin writers don't have to worry about providing an upgrade task framework. They can simply provide a Plugin Upgrade Task component and SAL will guarantee that their upgrade task is run on startup.

Example

Let's look at what needs to be done to run an upgrade task to convert the First Response Time chart portlet data over to gadget data.

1. Convert your Portlet to a Gadget

Please follow the documentation available in the Gadget Development Hub for this step, and specifically the page about gadgets and JIRA portlets.

2. Add dependency on SAL

First we'll need access to the SAL API in the charting plugin project. Add the following dependency to the plugin's `pom.xml`:

pom.xml

```
...
  <dependency>
     <groupId>com.atlassian.sal</groupId>
     <artifactId>sal-api</artifactId>
     <version>2.1</version>
     <scope>provided</scope>
  </dependency>
...
```

Re-generate your IDE's project descriptor (`mvn idea:idea` or `mvn eclipse:eclipse`) after this step to allow you to access the new SAL API classes in your project.

3. Convert your plugin to Plugins2

SAL is a Plugins2 bundle and your plugin will have to be converted to the Plugins2 format first before you can write an upgrade task that will be picked up by the PluginUpgradeManager. Gadgets are also only supported in Plugins2 bundles.

There are generic instructions available for how to do this, but let's look specifically at the Charting plugin example. The only thing that is needed is to add the **`plugins-version="2"`** attribute in `atlassian-plugins.xml`:

atlassian-plugins.xml

```
<atlassian-plugin key="${atlassian.plugin.key}" name="${project.name}" system="true" plugins-version=
"2">
...
```

4. Writing your upgrade task

Now that all the prerequisites are done, the Upgrade task for the plugin can be written. This class simply needs to implement the PluginUpgradeTask interface provided by SAL.

Here's an example implementation:

GadgetUpgradeTask.java

```
package com.atlassian.jira.ext.charting.upgrade;

import com.atlassian.gadgets.dashboard.Color;
import com.atlassian.jira.ComponentManager;
import com.atlassian.jira.portal.OfbizPortletConfigurationStore;
import com.atlassian.jira.portal.PortletConfiguration;
import com.atlassian.jira.portal.PortletConfigurationImpl;
import com.atlassian.jira.portal.PortletConfigurationStore;
import com.atlassian.jira.propertyset.JiraPropertySetFactory;
import com.atlassian.jira.upgrade.util.AbstractLegacyPortletUpgradeTask;
import com.atlassian.jira.util.Consumer;
import com.atlassian.jira.util.NotNull;
import com.atlassian.jira.util.collect.EnclosedIterable;
import com.atlassian.sal.api.message.Message;
import com.atlassian.sal.api.upgrade.PluginUpgradeTask;
import com.opensymphony.module.propertyset.PropertySet;
import org.apache.log4j.Logger;

import java.net.URI;
import java.util.Collection;
import java.util.Map;

public class GadgetUpgradeTask implements PluginUpgradeTask
{
    private static final Logger log = Logger.getLogger(GadgetUpgradeTask.class);

    private final PortletConfigurationStore portletConfigurationStore;
    private final JiraPropertySetFactory propertySetFactory;

    public GadgetUpgradeTask(JiraPropertySetFactory propertySetFactory)
    {
        //NOTE: Can't get the portletConfigStore injected here since it is not made available to
plugins2
this.portletConfigurationStore =
ComponentManager.getComponentInstanceOfType(PortletConfigurationStore.class);
        this.propertySetFactory = propertySetFactory;
    }

    /**
     * The build number for this upgrade task. Once this upgrade task has run the plugin manager will
store this
     * build number against this plugin type.  After this only upgrade tasks with higher build numbers
will be run
     */
    public int getBuildNumber()
    {
        return 1;
    }

    public String getShortDescription()
    {
        return "Upgrades legacy portlet configuration to new gadget user prefs.";
    }

    public Collection<Message> doUpgrade() throws Exception
    {
        final FirstResponseUpgradeTask upgradeTask = new FirstResponseUpgradeTask();
        //First get all the portletConfigurations in the database.
final EnclosedIterable<PortletConfiguration> iterable =
portletConfigurationStore.getAllPortletConfigurations();
        iterable.foreach(new Consumer<PortletConfiguration>()
        {
            public void consume(@NotNull final PortletConfiguration pc)
            {
                //for each portletconfiguration, check if it's key matches the portlet key we want to
upgrade
if (pc.getKey() != null && pc.getKey().startsWith(upgradeTask.getPortletKey()))
```

```
                {
                    log.info("Upgrading portletconfig with id '" + pc.getId() + "'");
                    //first lets convert the preferences for this portlet to the new prefs format used
for gadgets.
final Map<String, String> prefs = upgradeTask.convertUserPrefs(pc);
                    //then create essentially a copy of the old portletConfig.  This new copy no
longer needs to have
//the portletKey and propertySet set to any values.  It however does require the GadgetUri and user
prefs to be set.
final PortletConfiguration newConfig =
                            new PortletConfigurationImpl(pc.getId(), pc.getDashboardPageId(), null,
null,
                                    pc.getColumn(), pc.getRow(), null, upgradeTask.getGadgetUri(),
Color.color8, prefs, false);
                    //Now lets store this new config back to the database.
portletConfigurationStore.store(newConfig);
                    //clear out the old properties for this portlet
removePropertySet(pc);
                }
            }
        });

        return null;
    }

    private void removePropertySet(final PortletConfiguration pc)
    {
        final PropertySet livePropertySet =
propertySetFactory.buildNoncachingPropertySet(OfbizPortletConfigurationStore.TABLE, pc.getId());
        @SuppressWarnings ("unchecked")
        final Collection<String> keys = livePropertySet.getKeys();
        for (String propertyKey : keys)
        {
            livePropertySet.remove(propertyKey);
        }
    }

    /**
     * Identifies the plugin that will be upgraded.
     */
    public String getPluginKey()
    {
        return "com.atlassian.jira.ext.charting";
    }

    private static class FirstResponseUpgradeTask extends AbstractLegacyPortletUpgradeTask
    {
        /**
         * This is the key for the portlet we want to convert to a new gadget
         */
        public String getPortletKey()
        {
            return "com.atlassian.jira.ext.charting:firstresponsetime";
        }

        /**
         * Gadgets are now simply defined by a URI which points to the Gadget spec.  This method
returns the gadget
         * spec that will be used to replace the legacy portlet.
         */
        public URI getGadgetUri()
        {
            return URI.create("gadgets/firstresponse.xml");
        }
```

```
    }
}
```

There are a few things to note about this implementation:

- `getBuildNumber()` and `getPluginKey()` determine if this upgrade task will run. `getPluginKey()` needs to match the key of the plugin that is being upgraded (in this case the charting plugin). `getBuildNumber()` returns the buildnumber for this upgrade task. `'1'` will do for any plugin that hasn't had any upgrade tasks run against it yet. SAL's PluginUpgradeManager will run this upgrade task and store the buildnumber against the plugin once completed. After this, **only** upgrade tasks with a higher build number than '1' will be executed.
- `doUpgrade()` uses some helpers provided by JIRA (i.e. the AbstractLegacyPortletUpgradeTask) to convert the legacy portlet to a gadget. This is entirely optional, however, and plugin authors are free to implement this method however they like.

Please ensure that the plugin upgrade task **ONLY** upgrades `portletConfigurations` for the plugin that's being upgraded! Any other `portletConfigurations` MUST be left untouched, as otherwise there's a risk of clobbering other portlets' data!

5. Register the upgrade task

Now we simply need to register the upgrade task as a component in the plugin:

atlassian-plugin.xml

```
...
 <component key="gadgetUpgradeTask" name="Gadget Upgrade Task"
 class="com.atlassian.jira.ext.charting.upgrade.GadgetUpgradeTask" public="true">
     <interface>com.atlassian.sal.api.upgrade.PluginUpgradeTask</interface>
 </component>
...
```

The PluginUpgradeManager in SAL will automatically scan for components that implement the `PluginUpgradeTask` interface. Please note that they have to be declared as **`public="true"`**.

That's it. Simply re-package the plugin, deploy it to the instance of JIRA to upgrade and restart the JIRA instance. The plugin upgrade task should be executed when JIRA starts up.

It's highly recommended that you perform a backup of your JIRA instance before attempting this!

JIRA 4.0 Beta 1 Release Notes

JIRA 4.0 Beta 1 is a public development release leading up to **JIRA 4.0**. For all production use and testing of JIRA, please use the latest official release.

A **Beta release** is a preliminary release leading up to the official release of a JIRA version. Beta releases are a snapshot of our work in progress and provide an advance preview of new features to the general public. JIRA plugin developers can also use Beta releases to test and fix their plugins in advance of an official release.

Do not use in production
Beta releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **Beta releases are not safe** — Beta releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No upgrade path** — Because Beta releases represent work in progress, we **can not** provide a supported upgrade path between Beta releases, or from any Beta to the eventual final release. Thus, any data you store in a JIRA Beta release may not be able to be migrated to a future JIRA release.

The **only** plugin that is compatible with JIRA 4.0 Beta is the latest JIRA Toolkit. Do not install any other plugins.

The Atlassian JIRA team is delighted to present a brand new version of one of the world's favourite issue-trackers.

Highlights of JIRA 4.0 Beta 1:

- Advanced Searching
- Issue Actions in the Issue Navigator
- Charting Now Comes Standard
- New-look "Browse Project"
- Project Icons
- Dashboard Gadgets
- Activity Stream
- "History" is now permanent
- Plus over 150 other fixes and improvements

Thank you for your interest in JIRA 4.0 Beta 1

Download Beta

Upgrading to JIRA 4.0 Beta
JIRA 4.0 Beta can be downloaded here. Before upgrading, please refer to the JIRA 4.0 Upgrade Guide.

Highlights of JIRA 4.0 Beta 1

1

Advanced Searching

The new advanced search (JQL) provides support for logical operations (including AND/OR/NOT/NULL, even on custom fields) when filtering issues:

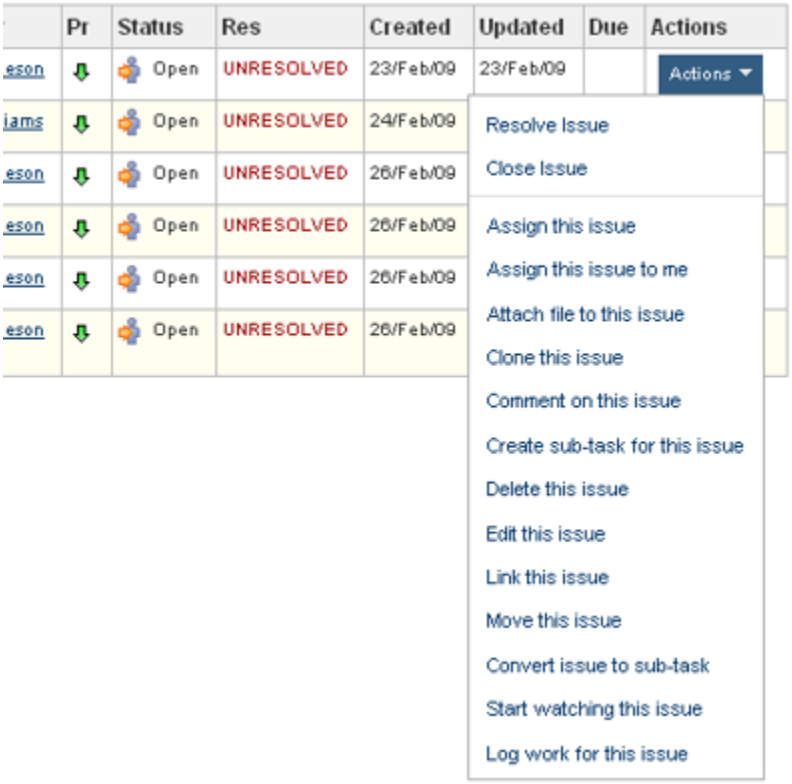
For more on the new JQL search syntax, please see the documentation.

^Top

2

Issue Actions in the Issue Navigator

By popular request, issues can now be actioned directly from the Issue Navigator:



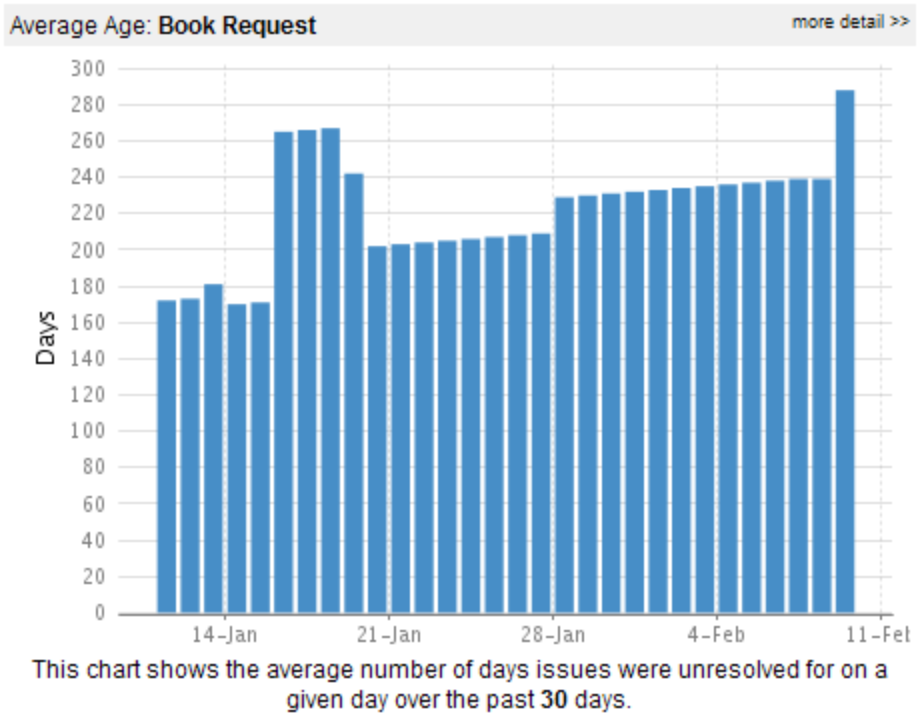
The "Actions" menu is also available for the list of sub-tasks within an issue.

^Top

Charting Now Comes Standard

The following reports and gadgets from the Charting plugin have now been integrated into JIRA:

- "Average Age" report and gadget — Shows the average age (in days) of unresolved issues, e.g.:



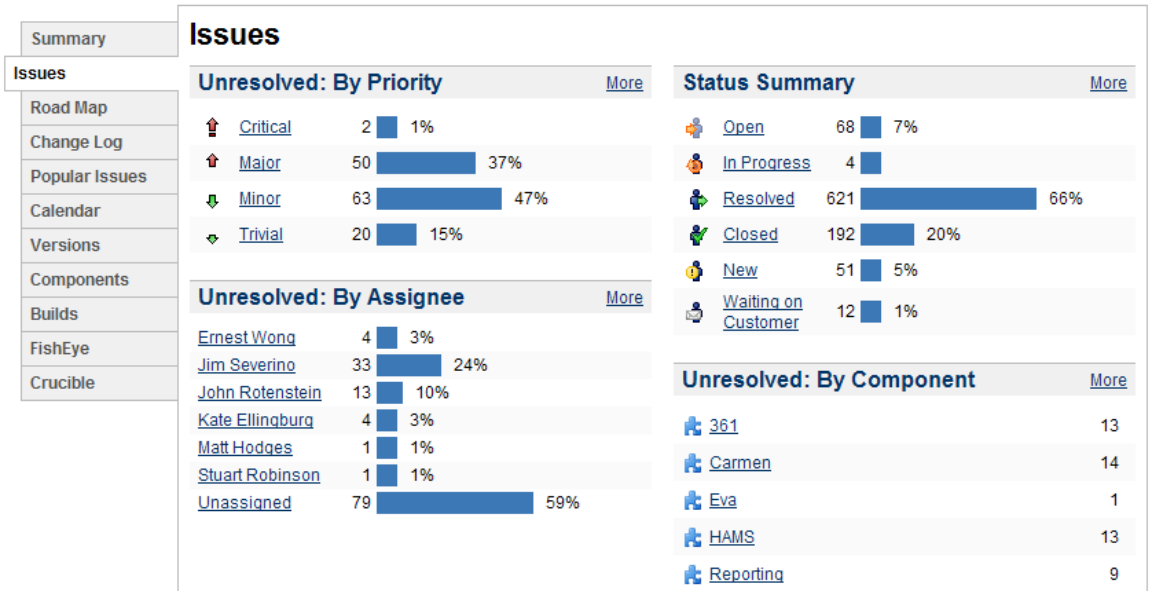
- "Created vs Resolved Issues" report and gadget — Shows the number of issues created vs number of issues resolved over a given period of time.
- "Pie Chart" report and gadget — Shows the search results from a specified issue filter (or project) in a pie-chart, based on a statistic of your choice.
- "Recently Created Issues" report and gadget — Shows the rate at which issues are being created.
- "Resolution Time" report and gadget — Shows the average time taken to resolve issues.
- "Time Since Issues" report and gadget — Shows the number of issues for which your chosen date field (e.g. 'Created') was set on a given date.

Also, the "Resolved" field from the Charting plugin is now part of JIRA, so every issue now automatically has its resolution date recorded.

^Top

New-look "Browse Project"

JIRA 4.0 provides a cleaner, more interactive view in to a project:



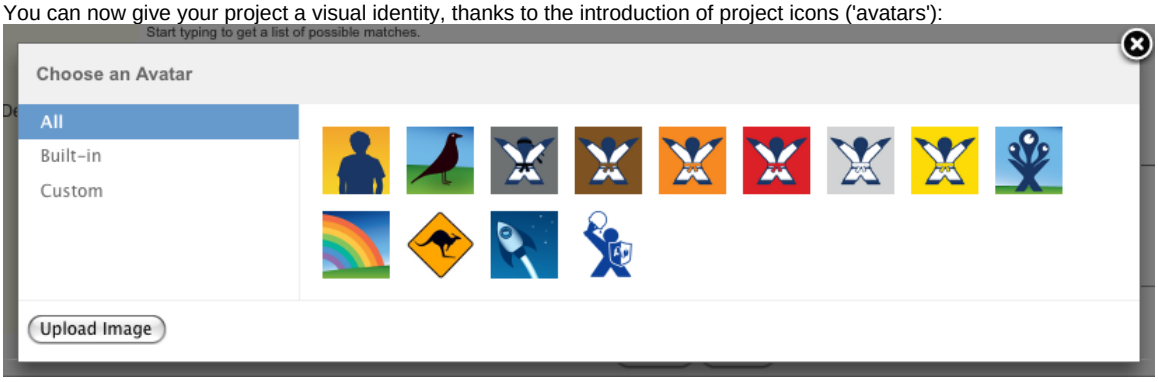
See the documentation for more about browsing projects, versions and components.

^Top

5

Project Icons

You can now give your project a visual identity, thanks to the introduction of project icons ('avatars'):



^Top

6

Dashboard Gadgets

The new-look JIRA dashboard now uses industry-standard 'gadgets'. So you can add external gadgets to your JIRA dashboard, as well as displaying JIRA gadgets in other places (such as iGoogle).

What's happened to your favourite JIRA portlets? Don't worry: every portlet that previously shipped with JIRA has been converted to a 'legacy gadget'. And if you are a plugin developer and have created your own portlets, see the instructions for converting your portlets to gadgets.

The following new gadgets are available in Beta 1:

- 'Activity Stream' gadget (see below)
- 'Create Issue' gadget
- 'Filter Results' gadget
- 'Issue Completed This Iteration' gadget

For optimal experience of the new dashboard, please use Firefox 3.x. Support for other browsers will be added prior to the launch of JIRA 4.0.

^Top

Activity Stream

The new 'Activity Stream' gadget displays a summary of the latest activity in JIRA projects (and/or by particular people) in which you are interested.

See the documentation for more details.

^Top

8

"History" is now permanent

Your list of recently-viewed issues is now stored in JIRA's database — so it's available after you log out and back in, even if you use a different machine.

^Top

9

Plus over 150 other fixes and improvements

JIRA Issues (200 issues)				
Key	**Type**	**Summary**	**Priority**	**Status**
JRA-923		Allow filter by "No Fix For" across projects		Resolved
JRA-1538		Filter on Versions and Components across Projects		Resolved
JRA-1560		Better support for logical operation (and/or/not) type of filters.		Resolved
JRA-1635		"not" qualifier on fields for searching		Resolved
JRA-1642		Create home directory instead of index & attachment directory		Resolved
JRA-1983		Enable filtering on "older than 1 month"		Resolved
JRA-1994		Ability to filter on time tracking related fields		Resolved
JRA-2033		Add an RSS feed query for comments to individual issues		Resolved

JRA-2469		It would be really nice to specficy several Asignee options in filters	Resolved
JRA-2607		Would like to create a filter also with OR conditions	Resolved
JRA-2681		Extend filter capablities by adding negative clauses	Resolved
JRA-2852		search for issues on version lower or equal to a given version	Resolved
JRA-2916		Allow Previous version searching	Resolved
JRA-2925		Can't filter by Security Level	Resolved
JRA-3000		Add key NUMBER (only number) searching to default search filter.	Resolved
JRA-3101		Jira - query / search / filter by issue links	Resolved
JRA-3206		View issues without an estimate	Resolved
JRA-3451		Enable filtering by Date Resolved	Resolved
JRA-3464		allow filtering by project category	Resolved
JRA-3624		released/unreleased version filter	Resolved
JRA-4227		Recent History Popup - persistance across sessions & more data	Resolved
JRA-4605		new filter criteria: add NOT to all existing criteria	Resolved
JRA-4688		Browse Project: Within the tab panel, if components are hidden - the version info appears to be right aligned.	Resolved
JRA-5121		Filter Portlet with configurable columns	Resolved
JRA-5152		Show issue linked to another issue.	Resolved
JRA-5201		Enable filter to specify more than 1 user	Resolved
JRA-5310		Watchlist should be exportable	Resolved
JRA-5383		My Votes and My Watches as filters	Resolved
JRA-5435		Issue actions and operations on Issue Navigator	Resolved
JRA-5560		Improved query functionality	Resolved
JRA-5844		allow multiple users when creating filters	Resolved
JRA-5965		Allow configure units of time tracking	Resolved
JRA-6010		Thought processing	Resolved

JRA-6180		Search for a custom field that is empty		Resolved
JRA-6344		Send to both previous and current assignees for all notifications		Resolved
JRA-6527		Allow filters to be built upon other shared filters (combined filters)		Resolved
JRA-6550		if text contains certain characters, cdata in xml based on this will be badly formed		Resolved
JRA-7068		Allow for list of issues to be saved as a filter		Resolved
JRA-7551		Provide capability to find issues by resolution date		Resolved
JRA-7626		Build search queries remotely		Resolved
JRA-7772		Ability to create advanced queries to search across all data		Resolved
JRA-7909		Search/ filter for "empty" fields		Resolved
JRA-8009		java.lang.IllegalStateException: getAttribute: Session already invalidated when trying to call getAvailableActions or getFieldsForAction		Resolved
JRA-8159		Add ability to issue navigator to find all issues linked to x issue - with option to constrain by link type		Resolved
JRA-8293		Import fails if searchrequest:request data too large		Resolved
JRA-8487		Bad logging from uk.ltd.getahead.dwr.util.CommonsLoggingOutput on startup		Resolved
JRA-8527		Put task actions directly in filter output		Resolved
JRA-8686		Allow searching of issues by Full Name for all user fields		Resolved
JRA-8758		Cannot create filter for multiple projects all issues in version "Released Versions"		Resolved
JRA-8852		Sort filter results by non-visible field		Resolved
JRA-8973		RSS of Project Changes		Resolved
JRA-9048		Calendar week begins with sunday independently from locale		Resolved
JRA-9115		Ability to search for issues with no due date associated		Resolved
JRA-9278		New Field "Resolution Date" automatically filled with date of setting resolution		Resolved
JRA-9651		User Activity Log		Resolved
JRA-9658		Minor css bug (cursor)		Resolved
JRA-9823		Allow to optionally clone an issue's attachments when cloning an issue.		Resolved
JRA-10245		Ability to filter/view Issues upon "Versions" across multiple "Projects"		Resolved

JRA-10422	Error in logs when nonexistent key used in wiki-rendered text		Resolved
JRA-10427	Changing field descriptions in "Field Configurations" for custom fields does not work		Resolved
JRA-10443	"Not Assigned to User" criteria in filters		Resolved
JRA-10492	Search for several users as Assignee or Reporter		Resolved
JRA-10603	MultipleSelect searcher for cascading selection field		Resolved
JRA-10644	Make filters more accessible		Resolved
JRA-11134	Allow setting of column order/sort with no issues in result set		Resolved
JRA-11933	AutoTransitionListener - Reopen transition deletes issue summary		Resolved
JRA-12165	Unclear error message when bulk moving issues whose reporter cannot create issues		Resolved
JRA-12200	Reporter system field throws ClassCastException after populateFromIssue() and validateFromParams()		Resolved
JRA-12525	Emails containing attachments with non-ASCII names lost		Resolved
JRA-12596	Enable cross-project filtering on special versions		Resolved
JRA-12656	Add paging/optimization for Change Log scope		Resolved
JRA-12816	OutOfMemoryError PermGen Space on Windows Func Test (under VMWare)		Resolved
JRA-12921	Ability to export Watched Issues to excel		Resolved
JRA-12976	AbstractMessageHandler might not be removing spaces from email addresses before using them to determine if a user exists when creating an issue from an email		Resolved
JRA-13011	Component of a subtask is still component of original project after moving an issue		Resolved
JRA-13035	CSV import can not import resolution date.		Resolved
JRA-13426	Next/previous version links for 'Browse Version' screen		Resolved
JRA-13625	Implicit profiling functionality broken		Resolved
JRA-13689	Saved filters reverted to "All projects" when we deleted a project contained		Resolved
JRA-13793	Confusing "The 'Project Information' panel is not available" message when fields are disabled		Resolved
JRA-13801	Call method addWorklogAndAutoAdjustRemainingEstimate, the soap server reponse with this type IssueServiceImpl$RemoteWorklogImpl		Resolved
JRA-13850	Servlet Content Listeners should implement the catch / log / rethrow pattern		Resolved
JRA-14031	Form data lost when using back and forward web browser buttons		Resolved

JRA-14220	Ensure the index optimize operation does not cause index lock timeouts		Resolved
JRA-14416	Move Issue with SubTask between differents project, IssueType and SubIssueType		Resolved
JRA-14490	Deleting project can cause filter to select all projects		Resolved
JRA-14513	JIRA Soap Service log and Access filter log footprint needs improving - Invoked Method would be handy		Resolved
JRA-14516	JIRA upgrade page should warn about possible character encoding issue if JIRA is moved between two servers.		Resolved
JRA-14598	Add access key for administer project from browse project		Resolved
JRA-14616	Ability to query for issues that you are not watching		Resolved
JRA-14654	ColorPicker for LookAndFeel page does not work on all browsers		Resolved
JRA-14701	OSPropertyText table should have the value column set to extremely-long datatype		Resolved
JRA-14747	Ability to search for issues with blockers linked to them		Resolved
JRA-14811	Deleting Group Does Not Remove Group From a Subscription		Resolved
JRA-15018	Improved SOAP and HTTP access logging		Resolved
JRA-15112	Adding Update Issue Field workflow postfunction causes OutOfMemoryError		Resolved
JRA-15254	Browse Project Panel: Do not show Closed UNRESOLVED issues as Open in the version list		Resolved
JRA-15266	Would prefer if the "Worklog" heading under "Operations" when viewing an issue were a link, like the rest of the operations.		Closed
JRA-15309	Wrong assignee drop-down list sorting with non-ascii characters		Resolved
JRA-15445	RPC plugin needs to be cleaned up		Resolved
JRA-15484	Tokenizing java exceptions fails if the exception is terminated with a full-stop.		Resolved
JRA-15517	Upgrade JIRA to use the latest version of Lucene indexing framework - v2.3.2		Resolved
JRA-15543	Show release date next to version name in the list of versions on Browse Project screen		Resolved
JRA-15545	Export issues to Excel/Word format with non-ASCII filter name does not handle the file name properly		Resolved
JRA-15546	Versions no longer display descriptions when browsing project		Resolved
JRA-15548	If an attempt to get the Index lock times out, the indexing operation is discarded		Resolved
JRA-15564	JIRA displays error after a reindex.		Resolved
JRA-15625	It is possible to disable plugins that then render jira incapable of restarting		Resolved

JRA-15631	jelly with invalid variables script returns blank page to user		Resolved
JRA-15638	The new dropdown does not appear to render correctly under IE6/7		Resolved
JRA-15646	Convert JIRA to jQuery		Resolved
JRA-15649	Sort list of plugins in Admin section alphabetically		Resolved
JRA-15665	Address issue of plugins pushing filter/report panel off screen		Resolved
JRA-15666	Add project information to the issue XML view		Resolved
JRA-15669	Drag and drop behaviour doesn't work on Modify Issues Type Scheme page		Closed
JRA-15700	Created VS Resolved cumulative + individual graphs' Y axis should should be independent		Resolved
JRA-15702	Migrate to licensing 2.0		Resolved
JRA-15723	Jelly AddComment tag changes the "updated" issue timestamp to execution script timestamp		Resolved
JRA-15761	If issue key contains unicode characters, redirect on create issue doesn't work		Resolved
JRA-15846	Allow changing license on Data Import in new JIRA instance.		Resolved
JRA-15872	"Browse Project" URL doesn't include current project		Resolved
JRA-15886	Add logging notification for index optimization events		Resolved
JRA-15962	Upgrade JIRA to Plugins 2.x		Resolved
JRA-15991	Merge translations files into one to make translating JIRA easier!		Resolved
JRA-16058	Aggressive locking in JiraCachingPropertySet causes high contention		Resolved
JRA-16074	Incorrect error warning message on navigator summary		Resolved
JRA-16080	1px offset in Firefox		Resolved
JRA-16088	Created VS Resolved Issues Report contains Old Filtter / Project Picker		Resolved
JRA-16112	Bug in progressWorkflowAction method in SOAP		Resolved
JRA-16113	Do not show negative values on Y axis in Created vs. Resolved chart		Resolved
JRA-16122	HTTP Basic auth should be enabled by default		Resolved
JRA-16138	Anonymous users should not be considered to "own" all anonymous comments.		Resolved
JRA-16151	Colon : in custom field search causes searching of wrong field		Resolved

JRA-16175	JIRA issues macro does not work with 4.0-m1 on EACJ		Resolved
JRA-16210	Display issue count on JQL execution		Resolved
JRA-16211	Enter / return should execute JQL		Resolved
JRA-16276	Adjust colours of Resoltion date chart to be more distinshable for the colour blind		Resolved
JRA-16278	Add ability to search for versions using regex or similar		Resolved
JRA-16316	Assigned To Me Portlet, selecting all columns to display causes error		Resolved
JRA-16339	The "Perm Gen" memory usage shown on the System Info page is incorrect.		Resolved
JRA-16351	Component plugin modules don't show up in plugins admin section in JIRA 4.0		Resolved
JRA-16363	ServiceProxyDestroyedException when you reactivate an OSGi plugin		Resolved
JRA-16407	JiraModuleDescriptorFactory doesn't define some plugins2 descriptors		Resolved
JRA-16424	log4j output should contain more information		Resolved
JRA-16443	Create a jira.field.resolution.include transition attribute		Resolved
JRA-16451	JIRA home directory created in working directory in JIRA standalone		Resolved
JRA-16485	The long component name, on clicking overlaps the UI element		Resolved
JRA-16498	Version/s and Component/s not validated when updating an issue		Resolved
JRA-16502	Local helper has invalid HTML, causing styling issues		Resolved
JRA-16508	no attachments are returned when 'field=attachment' is specified in XML view		Resolved
JRA-16510	Update to jQuery 1.3.2		Resolved
JRA-16522	Searching according to multiple assignees should be provided.		Resolved
JRA-16526	Catalan translations needs to be changed		Resolved
JRA-16527	Indexing fails (or atleast error thrown) if index location is pointing to an invalid reference		Resolved
JRA-16529	Project Avatars: User is unable to upload PNG and JPEG file for project avatars on IE 7.0		Resolved
JRA-16530	Project Avatars: User is unable to crop the uploaded image in the project avatars dialog		Resolved
JRA-16531	Project avatar: One image is missing from "Choose an avatar" dialog		Resolved
JRA-16532	User gets a system error if after timeout tries to login from project avatar dialog		Resolved

JRA-16560	Upgrading bunded plugins fails		Resolved
JRA-16568	Example for xmlrpc client uses a method that has been removed. Need to update		Resolved
JRA-16570	Text after a thumbnail in a comment has anchors		Resolved
JRA-16656	Go around JIRA and find all the places that are evaluating EL badly.		Resolved
JRA-16660	Project avatar dialog is not coming properly for IE 7.0		Resolved
JRA-16662	Cannot start up bundled plugins		Resolved
JRA-16664	Plugins2.2-beta12 Some UnloadablePlugins don't have a correct key causing JIRA startup to fail		Resolved
JRA-16675	Filter doesn't display errors when first loading into simple edit mode		Resolved
JRA-16685	Enormous font on random pages		Resolved
JRA-16686	DOCTYPE declaration included twice for issue related pages		Resolved
JRA-16698	Have the ability to log all SQL statements issued by JIRA and also have a callback for timing purposes		Resolved
JRA-16714	There is a bug with the way the QuerySearcher works with the "doesItFit"		Resolved
JRA-16715	JQL does not handle EMPTY embedded in lists		Resolved
JRA-16718	Fix the project clause validator to do a permission check to see if the user can see the project		Resolved
JRA-16728	Created vs Resolved: Link from Issue-Numbers in the Table to the corresponding searchrequest in the IssueNavigator		Resolved
JRA-16735	"Bulk Change" text in "Tools" drop down changes to link in certain scenerios		Resolved
JRA-16744	Improve the performance of checking if a user belongs to a particular group.		Resolved
JRA-16750	Fix any memory leaks in JIRA mainly caused by restoring data from XML and refreshing all singleton objects		Resolved
JRA-16757	ThreadLocalCachingPermissionManager.getProjects() returns mutable lists straight from the cache		Resolved
JRA-16788	Velocity errors in logs: "$portletConfig.key is not a valid reference"		Resolved
JRA-16799	JIRA does not handle Plugins 1 plugins in the installed-plugins folder.		Resolved
JRA-16807	Convert Intro Portlet		Resolved
JRA-16808	Convert Quicklinks portlet		Resolved
JRA-16809	Convert Favourite Fitlers		Resolved
JRA-16811	Convert Pie Chart Portlet		Resolved

JRA-16833	Avatar Picker has layout issues while cropping		Resolved
JRA-16838	Ensure Save button is disabled after submitting on the chart popup		Resolved
JRA-16839	Add nicer dashboard tabs		Resolved
JRA-16845	Version functions should validate their arguments based on permissions		Resolved
JRA-16846	Allow for Pluggable Decorators in JIRA		Resolved
JRA-16852	Comment/resolutions/etc are posted empty if previewed		Resolved
JRA-16870	Improve multi-threaded liveness of FieldLayoutManager under load		Resolved
JRA-16903	Convert CreatedVsResolved Chart Portlet		Resolved
JRA-16904	Get Edit HTML of Component Systemfield no longer orders project components		Resolved
JRA-16905	Convert Average Age Chart		Resolved
JRA-16908	Projects are not getting created with default Project Avatar		Resolved
JRA-16909	Changing the color of priority in "Edit priority" dialog, doesn't update the color icon in "Edit priority" dialog		Resolved
JRA-16914	Convert Recently Created Portlet		Resolved
JRA-16915	SOAP and Jelly Progress Workflow calls do not take into account exisiting issue values		Resolved
JRA-16916	Convert Time Since Chart		Resolved
JRA-16920	Add filter option for subtasks of a particular issue		Resolved
JRA-16926	Implement OAuth SPI in JIRA		Resolved

^Top

EAP Releases

An **Early Access Preview (EAP) release** is a public development release leading up to the official release of a JIRA version. Development releases are a snapshot of our work in progress, primarily focused on allowing JIRA plugin developers to test and fix their plugins in advance of an official release.

Do not use in production
EAP releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **EAP Releases are Not Safe** — EAP releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No Upgrade Path** — Because EAP releases represent work in progress, we **can not** provide a supported upgrade path between EAP releases, or from any EAP to the eventual final release. Thus, any data you store in a JIRA EAP release may not be able to be migrated to a future JIRA release.

The following EAP releases are currently available for download. Your help with testing them is very appreciated! Please log the bugs you find on http://jira.atlassian.com in the "JIRA" project.

EAP Releases

- JIRA 3.11 EAP Release Notes

Related Information

Beta Releases
You may also be interested in the JIRA Beta Releases, which are early releases intended for the general public as well as developers.

Latest Beta Release:

JIRA 3.11 EAP Release Notes

An **Early Access Preview (EAP) release** is a public development release leading up to the official release of a JIRA version. Development releases are a snapshot of our work in progress, primarily focused on allowing JIRA plugin developers to test and fix their plugins in advance of an official release.

Do not use in production
EAP releases should not be used in production environments as they are not officially supported.

Please also take note of the following information:

- **EAP Releases are Not Safe** — EAP releases are snapshots of the ongoing JIRA development process. As such:
  - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
  - Features in development releases may be incomplete, or may change or be removed before the next full release.
- **No Upgrade Path** — Because EAP releases represent work in progress, we **can not** provide a supported upgrade path between EAP releases, or from any EAP to the eventual final release. Thus, any data you store in a JIRA EAP release may not be able to be migrated to a future JIRA release.

JIRA 3.11 EAP is a public development release leading up to JIRA 3.11.

Who should download this release?

This EAP release is being made available specifically for **JIRA plugin developers** to test their existing plugins against indexing changes in JIRA 3.11. This affects all plugins that maintain their own indexes or interact with JIRA's index (e.g. via a custom field with a custom searcher that the plugin implements).

For all production use and testing of JIRA, please use the latest official release.

Plugins that depend on indexing will have to be updated to remove the use of any methods that were deprecated in Lucene 1.9.1. These plugins will have to be re-compiled against Lucene 2.2.0 and re-released for JIRA 3.11.

What's new?

This EAP release includes the following features:

- Aggregate Time Tracking information across sub-tasks
- Lucene upgrade to v2.2.0

JIRA 3.11 aims to upgrade Lucene to version 2.2.0. A number of deprecated methods that JIRA and JIRA plugins relied on where removed from this version of Lucene. Aside from the API changes, there were also changes to how certain fields are stored in the index themselves. Luckily Lucene 2.2.0 is backwards compatible and is able to read indexes created by earlier versions of Lucene (indexes will be converted to the Lucene 2.2.0 format as soon as documents are added). Once an index has been upgraded to Lucene 2.2.0 any previous version of Lucene will not be able to read this index any longer.

One change that may require you to re-index are how dates are stored in the index. For example, DateFields have been deprecated in favour of DateTools, which stores dates in a human readable form. If you convert your plugins to store dates using DateTools, you will need to advise all users that they will need to perform a re-index (currently the only way to do this for plugins is to delete the plugin's index files).

What should I do?

If you have developed custom plugins that use Lucene, either publicly in the JIRA Plugin Library, or privately for you company's internal use, it is critical that you test those plugins against the JIRA 3.11 EAP. Because of the upgrade Lucene 2.2 and the large number of deprecations, it is likely that you will need to make changes.

You should make any necessary changes to the plugin, and be prepared to release thhose fixes in conjunction with JIRA 3.11, so that users will have a smooth upgrade experience.

If you have any trouble with the EAP or the upgrade process, please contact developer-support@atlassian.com for assistance.

The EAP is available for download here.

Security Advisories

As a public-facing web application, JIRA's application-level security is important. This document contains links to version-specific security advisories and related documents for the JIRA application.

This document is intended to provide information to system administrators about the security of the JIRA application. It does not address JIRA's internal security model — user management and permissions — except as it relates to the overall application security.

On this page:

- Finding and Reporting a Security Vulnerability
- Publication of JIRA Security Advisories
- Severity Levels
  - Severity Level: Critical
  - Severity Level: High
  - Severity Level: Moderate
  - Severity Level: Low
- Our Patch Policy
- Security Advisories

Finding and Reporting a Security Vulnerability

If you find a security bug in JIRA, please open an issue on http://jira.atlassian.com in the JIRA project.

- Set the priority of the bug to 'Blocker'.
- Provide as much information on reproducing the bug as possible.
- Set the security level of the bug to 'Developer and Reporters only'.

All communication about the vulnerability should be performed through JIRA, so we can keep track of the issue and get a patch out as soon as possible.

Publication of JIRA Security Advisories

When a security issue in JIRA is discovered and resolved, we will inform customers through the following mechanisms:

- A security advisory will be posted on this page.
- A copy of the advisory will be sent to the `jira-users` and `jira-announce` mailing-lists (subscribe here). These lists are mirrored on our forums.
- If the person who reported the issue wants to publish an advisory through some other agency (for example, CERT), we'll assist in the production of that advisory, and link to it from our own.

Latest security advisory:

- JIRA Security Advisory 2009-04-02

Severity Levels

Atlassian security advisories include a severity level, rating the vulnerability as one of the following:

- Critical
- High
- Moderate
- Low

Below is a summary of the factors which we use to decide on the severity level, and the implications for your installation.

Severity Level: Critical

We classify a vulnerability as critical if most or all of the following are true:

- Exploitation of the vulnerability results in root-level compromise of servers or infrastructure devices.
- The information required in order to exploit the vulnerability, such as example code, is widely available to attackers.
- Exploitation is usually straightforward, in the sense that the attacker does not need any special authentication credentials or knowledge about individual victims, and does not need to persuade a target user, for example via social engineering, into performing any special functions.

Severity Level: High

We give a high severity level to those vulnerabilities which have the potential to become critical, but have one or more mitigating factors that make exploitation less attractive to attackers.

For example, given a vulnerability which has many characteristics of the critical severity level, we would give it a level of high if any of the following are true:

- The vulnerability is difficult to exploit.
- Exploitation does not result in elevated privileges.
- The pool of potential victims is very small.

Note: If the mitigating factor arises from a lack of technical details, the severity level would be elevated to critical if those details later became available. If your installation is mission-critical, you may want to treat this as a critical vulnerability.

Severity Level: Moderate

We give a moderate severity level to those vulnerabilities where the scales are slightly tipped in favour of the potential victim.

The following vulnerabilities are typically rated moderate:

- Denial of service vulnerabilities, since they do not result in compromise of a target.
- Exploits that require an attacker to reside on the same local network as the victim.
- Vulnerabilities that affect only nonstandard configurations or obscure applications.
- Vulnerabilities that require the attacker to manipulate individual victims via social engineering tactics.
- Vulnerabilities where exploitation provides only very limited access.

Severity Level: Low

We give a low severity level to those vulnerabilities which by themselves have typically very little impact on an organisation's infrastructure.

Exploitation of such vulnerabilities usually requires local or physical system access. Exploitation may result in client-side privacy or denial of service issues and leakage of information about organisational structure, system configuration and versions, or network topology.

Original ranking compiled by the SANS Institute
Our vulnerability ranking is based on a scale originally published by the SANS Institute.

Our Patch Policy

When a security issue is discovered, we will endeavour to:

- issue a new, fixed JIRA version as soon as possible.
- issue a patch to the current stable version of JIRA.
- issue patches for older versions of JIRA if feasible.

Patches will generally be attached to the relevant JIRA issue.

Security Advisories

- JIRA Security Advisory 2009-04-02
- JIRA Security Advisory 2008-12-09
- JIRA Security Advisory 2008-10-29
- JIRA Security Advisory 2008-08-26
- JIRA Security Advisory 2008-02-21
- JIRA Security Advisory 2007-12-24

JIRA Security Advisory 2007-12-24

In this advisory:

- Security vulnerabilities
  - XSS vulnerability in Issue Actions
  - Anyone can delete a filter which is shared with them
  - Default language setting can be changed by an unauthorised user
- Available JIRA Patches
  - JIRA 3.12
  - JIRA 3.11
  - JIRA 3.10.2

Security vulnerabilities

XSS vulnerability in Issue Actions

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in JIRA's issue actions, which potentially allows a malicious user (hacker) to insert their own HTML tags or script into an action.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker's text and script might be displayed to other people viewing the JIRA issue. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.12.1, or download the patch for JIRA 3.11 or 3.10.2, to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

All issue actions (e.g. 'Create issue') are affected. The problem is with `500page.jsp`. It does not HTML-escape the error messages it prints out.

Fix

The fix is to escape all of the error messages rendered on the 500 page, so that no user input, which is propagated to error messages, is interpreted as HTML or CSS.

This issue has been fixed in JIRA 3.12.1. The fix is also provided as a patch for JIRA 3.12, 3.11 and 3.10.2. For more information, please see JRA-14105.

Anyone can delete a filter which is shared with them

Severity

Atlassian rates this vulnerability as **LOW**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw allows users to delete filters which are shared with them, which is an inconvenience to the user who is the true owner of the filter.

Atlassian recommends that you upgrade to JIRA 3.12.1, or download the patch for JIRA 3.12, 3.11 or 3.10.2, to fix the vulnerabilities described below.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could instruct all users to share their filters with trusted groups only (i.e. instruct them not to use 'Global' sharing).

Vulnerability

When a user commences deleting one of their own filters, if they replace their filter ID with the ID of another user's filter which is shared with them, they can delete the other user's filter.

Fix

The fix is to check that the currently logged-in user is indeed the owner of the filter, before deleting a filter.

This issue has been fixed in JIRA 3.12.1. The fix is also provided as a patch for JIRA 3.12, 3.11 and 3.10.2. For more information, please see JRA-13999.

Default language setting can be changed by an unauthorised user

Severity

Atlassian rates this vulnerability as **LOW**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw potentially allows a malicious user (hacker) to change the default language of your JIRA instance, which is potentially damaging to your company's reputation, and an inconvenience to users.

Atlassian recommends that you upgrade to JIRA 3.12.1, or download the patch for JIRA 3.11 or 3.10.2, to fix the vulnerabilities described below.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

After a JIRA instance has been setup, the first page of the Setup Wizard can still be accessed by manually browsing to the URL.

Attempting to advance beyond this screen, or import data, correctly results in the "Already Setup" page being displayed. However, the default language for the JIRA instance can be modified without any security checks.

Fix

The fix is to check that JIRA has not already been setup, when a user attempts to access the any page of the Setup Wizard. Similar checks also occur when a user attempts direct access to the setup JSPs.

This issue has been fixed in JIRA 3.12.1. The fix is also provided as a patch for JIRA 3.11 and 3.10.2. For more information, please see

JRA-14086.

Available JIRA Patches

JIRA 3.12

The patches for JIRA 3.12 are available in the file jira_3_12_xss_patch.zip

Patch Zip File	jira_3_12_xss_patch.zip
Patch Instructions	jira_3_12_xss_patch_instructions.txt
Patch CheckSum	jira_3_12_xss_patch.zip.md5

JIRA 3.12 can also be fixed by upgrading to JIRA 3.12.1

JIRA 3.11

The patches for JIRA 3.11 are available in the file jira_3_11_xss_patch.zip

Patch Zip File	jira_3_11_xss_patch.zip
Patch Instructions	jira_3_11_xss_patch_instructions.txt
Patch CheckSum	jira_3_11_xss_patch.zip.md5

JIRA 3.10.2

The patches for JIRA 3.10 are available in the file jira_3_10_2_xss_patch.zip

Patch Zip File	jira_3_10_2_xss_patch.zip
Patch Instructions	jira_3_10_2_xss_patch_instructions.txt
Patch CheckSum	jira_3_10_2_xss_patch.zip.md5

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Security Advisory 2008-02-21

In this advisory:

- Security vulnerabilities
  - XSS vulnerability in Issue Actions
- Available JIRA Patches
  - JIRA 3.12.1
  - JIRA 3.11
  - JIRA 3.10.2

Security vulnerabilities

XSS vulnerability in Issue Actions

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in JIRA's '**Saved Filter**', '**Filter Statistics**', '**Project Statistics**' and '**2D Filter Statistics**' portlets. This potentially allows a malicious user (hacker) to create a shared filter with special JavaScript in the name, and then create a link to run the vulnerable portlets using the shared

filter. If this link was sent to a user and clicked by the user, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen, by using the jelly runner.
- The hacker's text and script might be displayed to other people viewing the JIRA Dashboard. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.12.2, or download the patch for JIRA 3.12.1, 3.11 or 3.10.2, to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

The '**Saved Filter**', '**Filter Statistics**', '**Project Statistics**' and '**2D Filter Statistics**' portlets are affected. The name of a shared filter is not HTML-escaped when the the portlet is viewed.

Fix

The fix is to escape the name of a shared filter when run by the '**Saved Filter**', '**Filter Statistics**', '**Project Statistics**' and '**2D Filter Statistics**' portlets, so that no content in the filter name is interpreted as HTML or CSS.

This issue has been fixed in JIRA 3.12.2. The fix is also provided as a patch for JIRA 3.12.1, 3.11 and 3.10.2. For more information, please see JRA-14277 and JRA-14357.

Available JIRA Patches

JIRA 3.12.1

The patches for JIRA 3.12.1 are available in the file jira_3_12_1_xss_patch.zip

Patch Zip File	jira_3_12_1_xss_patch.zip
Patch Instructions	jira_3_12_1_xss_patch_instructions.txt
Patch CheckSum	jira_3_12_1_xss_patch.zip.md5

JIRA 3.12.1 can also be fixed by upgrading to JIRA 3.12.2

JIRA 3.11

The patches for JIRA 3.11 are available in the file jira_3_11_xss_patch.zip

Patch Zip File	jira_3_11_xss_patch.zip
Patch Instructions	jira_3_11_xss_patch_instructions.txt
Patch CheckSum	jira_3_11_xss_patch.zip.md5

JIRA 3.10.2

The patches for JIRA 3.10 are available in the file jira_3_10_2_xss_patch.zip

Patch Zip File	jira_3_10_2_xss_patch.zip
Patch Instructions	jira_3_10_2_xss_patch_instructions.txt
Patch CheckSum	jira_3_10_2_xss_patch.zip.md5

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Security Advisory 2008-08-26

In this advisory:

- Security vulnerabilities
  - XSS vulnerability in serving HTML attachments with the text/html MIME type
  - MailHandlers may create an infinite loop if the monitored mailbox receives notifications from the same instance of JIRA
  - Directory listings are enabled on Tomcat by default
  - Filters/Search Requests can be modified by URL Hacking
  - 'Manage Project Role Membership for Project' page can be viewed publicly

---

Security vulnerabilities

XSS vulnerability in serving HTML attachments with the text/html MIME type

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and addressed a security vulnerability which may affect JIRA instances in a public environment. This is an XSS (cross-site scripting) vulnerability in JIRA's service of HTML attachments (or other active content, such as Javascript, Flash, etc) with the text/html MIME type, which potentially allows a malicious user (attacker) to insert their own HTML tags or script into an action.

- The attacker could take advantage of this vulnerability to steal other users' session cookies or other credentials, by sending the credentials back to the attacker's own web server.
- The attacker's text and script could be displayed to other people viewing the JIRA issue. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.13 to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable attachments or restrict public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

Any malicious script contained in an HTML attachment of with the text/html MIME type will be run as JIRA serves the attachment, i.e. when an admin or user clicks on the uploaded HTML attachment.

Fix

The fix is to add an administration option to force all attachments in JIRA to be downloaded rather than displayed inline. Administrators can choose from the following:

- force all attachments to be downloaded in JIRA,
- let all attachments be displayed inline, or,
- for Internet Explorer users, force the download of attachments that IE detects to be html files (via mime sniffing). Declared html attachments are also never displayed inline.

Read the documentation for further details on configuring this setting.

This issue has been fixed in JIRA 3.13 only. There are no patches available for previous versions of JIRA, for this fix.

MailHandlers may create an infinite loop if the monitored mailbox receives notifications from the same instance of JIRA

Severity

Atlassian rates this vulnerability as **MEDIUM**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw means that mailhandlers can potentially cause infinite loops if the monitored mailbox receives notifications from the same JIRA instance.

Atlassian recommends that you upgrade to JIRA 3.13 to fix the vulnerability described below.

Risk Mitigation

If you judge it necessary, you can disable disable your mail servers or disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade.

Vulnerability

User sends an email to a JIRA mailbox, where the From and To address are the same, e.g. if an email is sent to a mailbox monitored by JIRA with a 'From' email address identical to the mailbox address it is being sent to, then JIRA will pick up the email again and start an infinite loop for that issue.
This also applies to scenarios where JIRA sends emails to an address which is an alias for a mailbox that it checks.

Fix

The fix is to add a header to the outgoing email that contains a special JIRA "fingerprint" (X-JIRA-FINGERPRINT) that is unique to the JIRA instance.

This issue has been fixed in JIRA 3.13 only. There are no patches available for previous versions of JIRA, for this fix.

Directory listings are enabled on Tomcat by default

Severity

Atlassian rates this vulnerability as **LOW**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and addressed a security flaw which may affect JIRA instances in a public environment. This flaw means that directory listings on the Tomcat application server are public by default.

Atlassian recommends that you upgrade to JIRA 3.13 to fix the vulnerability described below. Alternatively, you can manually disable the directory listing (via the `<TOMCAT_HOME>/conf/web.xml` file in Tomcat directory), which will force JIRA to throw HTTP 404 errors appropriately.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade.

Vulnerability

Users can browse the directory listing on the Tomcat application server, e.g. `/images/`. Please note, the information accessible by the user is already readily available to the user, or can be obtained by downloading JIRA. The webapp directories **do not** contain any user content.

Fix

The fix is to disable directory listings in Tomcat. Please refer to JRA-11634 for details.

The directory listings are disabled by default in Tomcat 5.5.26. This version is bundled with the latest version of JIRA.

This issue has been fixed in JIRA 3.13 for JIRA Standalone and for the sample Tomcat (i.e. versions 4.1, 5.0, 5.5 and 6.0) configuration files shipped with JIRA WAR/EAR. There are no patches available for previous versions of JIRA, for this fix.

Filters/Search Requests can be modified by URL Hacking

Severity

Atlassian rates this vulnerability as **MODERATE**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and addressed a security flaw which may affect JIRA instances in a public environment. This flaw means that issue filters can be modified by hacking the URL, regardless of permissions on the filter.

Atlassian recommends that you upgrade to JIRA 3.13 to fix the vulnerability described below.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade.

Vulnerability

Users can run an issue filter, which they do not have access to, by entering the appropriate URL (although the filter will not return any issues that the user does not have permission to see). By the same means, users can edit a filter, rename a filter and access share and column selection. Filter deletion cannot be actioned purely by the URL, as it requires interaction with the user interface (which enforces permissions).

Fix

The fix is to revise the issue filter functionality as part of the Shareable Filters feature, so that URL hacks are no longer valid.

This issue has been fixed in JIRA 3.13 only. There are no patches available for previous versions of JIRA, for this fix.

'Manage Project Role Membership for Project' page can be viewed publicly

Severity

Atlassian rates this vulnerability as **LOW**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and addressed a security flaw which may affect JIRA instances in a public environment. This flaw means that the 'Manage Project Role Membership for Project' page can be viewed by users who are not logged in. Users cannot view any project role members or modify project roles.

Atlassian recommends that you upgrade to JIRA 3.13 to fix the vulnerability described below.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade.

Vulnerability

Users, who are not logged in, can manually enter the URL for the 'Manage Project Role Membership for Project' to access the page. Project role members will not be visible, nor will the user be able to modify project roles. The only new information available to the user will be the project name.

Fix

The fix is to prompt the user with the appropriate page for unauthorised access, if they are not logged in.

This issue has been fixed in JIRA 3.13 only. There are no patches available for previous versions of JIRA, for this fix.

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Security Advisory 2008-10-29

In this advisory:

- Security vulnerabilities
  - XSS vulnerability on ViewProfile page
  - Return URL is not HTML escaped

Security vulnerabilities

XSS vulnerability on ViewProfile page

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in JIRA's '**ViewProfile**' page. This potentially allows a malicious user (hacker) to create a user with special JavaScript in the fullname of the user. If this user was viewed by another user in the ViewProfile page, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.

Atlassian recommends that you upgrade to JIRA 3.13.1 to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

The '**ViewProfile**' page is affected. The user's 'fullname' is not HTML-escaped when the the page is viewed.

Fix

The fix is to HTML-encode the fullname of the user on the '**ViewProfile**' page, so that it cannot be used to run special scripts.

This issue has been fixed in JIRA 3.13.1 only. There are no patches available for previous versions of JIRA, for this fix. For more information, please see JRA-15733.

Return URL is not HTML escaped

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in the returnURL parameter of the URL of a form (e.g. Add Comment). This potentially allows a malicious user (hacker) to hack the URL to insert special JavaScript in the returnURL parameter. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users click the URL, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on any JIRA page which has a form. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.13.1 to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

All forms in JIRA are affected. The returnURL is not HTML-escaped when the the page is viewed.

Fix

The fix is to HTML-encode the returnURL of form URLs, so that it cannot be used to run special scripts.

This issue has been fixed in JIRA 3.13.1 only. There are no patches available for previous versions of JIRA, for this fix. For more information, please see JRA-15707.

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Security Advisory 2008-12-09

In this advisory:

- Security Vulnerabilities
  - WebWork 1 Parameter Injection Hole
- Available JIRA Patches
  - JIRA 3.13.1
  - JIRA 3.12.3
  - JIRA 3.11
  - JIRA 3.10.2
  - JIRA 3.9.3
  - JIRA 3.8.1
  - JIRA 3.7.4
  - JIRA 3.6.5
  - JIRA 3.5.3
  - JIRA 3.4.x and earlier

Security Vulnerabilities

WebWork 1 Parameter Injection Hole

Severity

Atlassian rates this vulnerability as **CRITICAL**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is a parameter injection vulnerability in the implementation of the WebWork 1 web application framework in JIRA. The Webwork 1 web application framework allows for the dynamic transformation of URL parameters into method calls. This potentially allows a malicious user (hacker) to call exposed public methods in JIRA via specially formatted URLs.

Atlassian recommends that you upgrade to JIRA 3.13.2 to fix the vulnerabilities described below.

Risk Mitigation

We **strongly recommend** that you upgrade or apply the necessary patch as soon as possible. If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

All versions of JIRA are vulnerable to this security flaw.

A number of public JIRA methods are exposed to this vulnerability. These methods can be called via specially formatted URLs. The method names are not listed for security reasons.

Fix

The fix is to process parameters via a trusted implementation of the action factory in the Webwork 1 web application framework, which provides more secure method transformations.

This issue has been fixed in JIRA 3.13.2 or later. The fix is also provided as a patch for JIRA 3.12.3, 3.11, 3.10.2, 3.9.3, 3.8.1, 3.7.4, 3.6.5 and 3.5.3. There are no patches available for JIRA versions 3.4.x or earlier. We recommend that you upgrade to at least JIRA 3.5.x to apply this patch.

Available JIRA Patches

JIRA 3.13.1

The patches for JIRA 3.13.1 are available in the file jra-15664-3.13.1-patch.zip

Patch Zip File	jra-15664-3.13.1-patch.zip
Patch Instructions	jra-15664-3.13.1-patch-instructions.txt
Patch CheckSum	jra-15664-3.13.1-patch.zip.md5

If you are using a version of JIRA 3.13.x **prior to version 3.13.1**, you will need to upgrade to JIRA 3.13.1 before applying this patch.

JIRA 3.12.3

The patches for JIRA 3.12.3 are available in the file jra-15664-3.12.3-patch.zip

Patch Zip File	jra-15664-3.12.3-patch.zip
Patch Instructions	jra-15664-3.12.3-patch-instructions.txt
Patch CheckSum	jra-15664-3.12.3-patch.zip.md5

If you are using a version of JIRA 3.12.x **prior to version 3.12.3**, you will need to upgrade to JIRA 3.12.3 before applying this patch.

JIRA 3.11

The patches for JIRA 3.11 are available in the file jra-15664-3.11-patch.zip

Patch Zip File	jra-15664-3.11-patch.zip
Patch Instructions	jra-15664-3.11-patch-instructions.txt
Patch CheckSum	jra-15664-3.11-patch.zip.md5

JIRA 3.10.2

The patches for JIRA 3.10.2 are available in the file jra-15664-3.10.2-patch.zip

Patch Zip File	jra-15664-3.10.2-patch.zip
Patch Instructions	jra-15664-3.10.2-patch-instructions.txt
Patch CheckSum	jra-15664-3.10.2-patch.zip.md5

If you are using a version of JIRA 3.10.x prior to version 3.10.2, you will need to upgrade to JIRA 3.10.2 before applying this patch.

JIRA 3.9.3

The patches for JIRA 3.9.3 are available in the file jra-15664-3.9.3-patch.zip

Patch Zip File	jra-15664-3.9.3-patch.zip
Patch Instructions	jra-15664-3.9.3-patch-instructions.txt
Patch CheckSum	jra-15664-3.9.3-patch.zip.md5

If you are using a version of JIRA 3.9.x prior to version 3.9.3, you will need to upgrade to JIRA 3.9.3 before applying this patch.

JIRA 3.8.1

The patches for JIRA 3.8.1 are available in the file jra-15664-3.8.1-patch.zip

Patch Zip File	jra-15664-3.8.1-patch.zip
Patch Instructions	jra-15664-3.8.1-patch-instructions.txt
Patch CheckSum	jra-15664-3.8.1-patch.zip.md5

If you are using a version of JIRA 3.8.x prior to version 3.8.1, you will need to upgrade to JIRA 3.8.1 before applying this patch.

JIRA 3.7.4

The patches for JIRA 3.7.4 are available in the file jra-15664-3.7.4-patch.zip

Patch Zip File	jra-15664-3.7.4-patch.zip
Patch Instructions	jra-15664-3.7.4-patch-instructions.txt
Patch CheckSum	jra-15664-3.7.4-patch.zip.md5

If you are using a version of JIRA 3.7.x prior to version 3.7.4, you will need to upgrade to JIRA 3.7.4 before applying this patch.

JIRA 3.6.5

The patches for JIRA 3.6.5 are available in the file jra-15664-3.6.5-patch.zip

Patch Zip File	jra-15664-3.6.5-patch.zip
Patch Instructions	jra-15664-3.6.5-patch-instructions.txt
Patch CheckSum	jra-15664-3.6.5-patch.zip.md5

If you are using a version of JIRA 3.6.x prior to version 3.6.5, you will need to upgrade to JIRA 3.6.5 before applying this patch.

JIRA 3.5.3

The patches for JIRA 3.5.3 are available in the file jra-15664-3.5.3-patch.zip

Patch Zip File	jra-15664-3.5.3-patch.zip
Patch Instructions	jra-15664-3.5.3-patch-instructions.txt
Patch CheckSum	jra-15664-3.5.3-patch.zip.md5

If you are using a version of JIRA 3.5.x prior to version 3.5.3, you will need to upgrade to JIRA 3.5.3 before applying this patch.

JIRA 3.4.x and earlier

There are no patches available for JIRA versions 3.4.x or earlier. We recommend that you upgrade to at least JIRA 3.5.x.

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Security Advisory 2009-04-02

In this advisory:

- Security Vulnerabilities
  - HTTP Header Injection Flaw
  - DWR XSS Security Hole
  - XSS vulnerability in various JIRA parameters
- Security Vulnerabilities - JIRA Plugins
  - JIRA Charting Plugin XSS Security Hole

Security Vulnerabilities

HTTP Header Injection Flaw

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is a HTTP Header injection vulnerability in JIRA. This potentially allows a malicious user (hacker) to hack the header response to insert malicious code. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users click the URL, the malicious code would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker could redirect the user to undesirable web sites. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.13.3 to fix the vulnerabilities described below.

Risk Mitigation

We strongly recommend that you upgrade or apply the necessary patch as soon as possible.

If you are unable to do this, you may wish to consult the vendor of your application server to see whether your application server is immune to header injection vulnerabilities or has configuration options to prevent such attacks. For example, the Coyote (HTTP) connector in Tomcat version 5.5 and later is immune to header injection attacks, as acknowledged in this reference.

Please note, the time required to fix this vulnerability and the extent of its effectiveness will depend on your application server and its configuration.

> **Technical Note**
> In your application server, header injection vulnerabilities can be mitigated if the `setHeader()`, `addHeader()`, and `sendRedirect()` methods in the `HttpServletResponse` class have their parameters properly checked for header termination characters. You may wish to forward this information to the vendor of your application server to help them advise whether they have any countermeasures to protect your application server against header injection attacks.

Vulnerability

All versions of JIRA are vulnerable to this security flaw.

Fix

The fix updates the Seraph framework to a version which correctly encodes and validates redirect URLs before sending them back to the user.

This issue has been fixed in JIRA 3.13.3 or later. The fix is also provided as a patch for JIRA 3.12.3 and 3.11. There are no patches available for JIRA versions 3.10.x and earlier. We recommend that you upgrade to at least JIRA 3.11 to apply this patch.

Available JIRA Patches

JIRA 3.12.3

A replacement seraph jar for JIRA 3.12.3 is available here: atlassian-seraph-0.38.3.jar

Replace JIRA's existing seraph jar with the updated one:

1. Delete the existing seraph jar in WEB-INF/lib/atlassian-seraph-0.37.2.jar
2. Place the replacement atlassian-seraph-0.38.3.jar into WEB-INF/lib

jar file	atlassian-seraph-0.38.3.jar
MD5 sum	atlassian-seraph-0.38.3.jar.md5

JIRA 3.11

A replacement seraph jar for JIRA 3.11 is available here: seraph-0.7.21.1.jar

Replace JIRA's existing seraph jar with the updated one:

1. Delete the existing seraph jar in WEB-INF/lib/seraph-0.7.21.jar
2. Place the replacement seraph-0.7.21.1.jar into WEB-INF/lib

jar file	seraph-0.7.21.1.jar
MD5 sum	seraph-0.7.21.1.jar.md5

JIRA 3.10.x and earlier

There are no patches available for JIRA versions 3.10.x or earlier. We recommend that you upgrade to at least JIRA 3.11.

DWR XSS Security Hole

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a security flaw which may affect JIRA instances in a public environment. This flaw is a XSS vulnerability in the DWR library in JIRA. This potentially allows a malicious user (hacker) to hack the URL to insert special JavaScript. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users click the URL, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on any JIRA page which has a form. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to JIRA 3.13.3 to fix the vulnerabilities described below.

Risk Mitigation

We recommend that you upgrade or apply the necessary patch as soon as possible. If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

All versions of JIRA are vulnerable to this security flaw.

Fix

The fix is to upgrade the DWR library shipped with JIRA to version 2.0.3. This version of the DWR library does not have this security flaw.

This issue has been fixed in JIRA 3.13.3 or later. The fix is also provided as a patch for JIRA 3.12.3 and 3.11. There are no patches available for JIRA versions 3.10.x or earlier. Please see JRA-16072 for further details.

Available JIRA Patches

JIRA 3.12.3

The patches for JIRA 3.12.3 are available in the file jra-16072-3.12.3-patch.zip

Patch Zip File	jra-16072-3.12.3-patch.zip
Patch Instructions	jra-16072-3.12.3-patch-instructions.txt
Patch CheckSum	jra-16072-3.12.3-patch.zip.md5

If you are using a version of JIRA 3.12.x prior to version 3.12.3, you will need to upgrade to JIRA 3.12.3 before applying this patch.

JIRA 3.11

The patches for JIRA 3.11 are available in the file jra-16072-3.11-patch.zip

Patch Zip File	jra-16072-3.11-patch.zip
Patch Instructions	jra-16072-3.11-patch-instructions.txt
Patch CheckSum	jra-16072-3.11-patch.zip.md5

JIRA 3.10.x and earlier

There are no patches available for JIRA versions 3.10.x or earlier. We recommend that you upgrade to at least JIRA 3.11.

XSS vulnerability in various JIRA parameters

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed a number of security flaws which may affect JIRA instances in a public environment. The flaws are all XSS (cross-site scripting) vulnerabilities in various JIRA parameters. Each vulnerability potentially allows a malicious user (hacker) to embed their own JavaScript into a JIRA page.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.

Atlassian recommends that you upgrade to JIRA 3.13.3 to fix the vulnerabilities described below.

You can read more about XSS attacks at cgisecurity, CERT and other places on the web.

Risk Mitigation

If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

A hacker can inject their own JavaScript into various JIRA parameters, described in the table below. If rogue JavaScript is injected into a parameter of a URL, the JavaScript will be executed when a user invokes the URL for the page.

JIRA page	Description
lazyLoader *(portlet loader)*	portletId
`CreateIssueDetails.jspa`	duedate
`EditIssue.jspa`	duedate
`jira.issueviews:searchrequest-fullcontent/temp/SearchRequest.html`	sorter/field, sorter/order
`jira.issueviews:searchrequest-printable/temp/SearchRequest.html`	sorter/order

For more information, please see JRA-16369.

Fix

The fix is to HTML-encode the vulnerable parameters to prevent scripts from being executed from them.

This issue has been fixed in JIRA 3.13.3 only. There are no patches available for previous versions of JIRA, for this fix.

Security Vulnerabilities — JIRA Plugins

JIRA Charting Plugin XSS Security Hole

Severity

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the JIRA Security documentation. This scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed two security flaws in the JIRA Charting plugin which may affect JIRA instances in a public environment that use this plugin. These flaws are XSS vulnerabilities in view actions for the JIRA Charting plugin. This potentially allows a malicious user (hacker) to hack the URL to insert special JavaScript. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users click the URL, the special JavaScript would be executed in the user's session.

- The hacker might take advantage of this flaw to steal other users' session cookies or other credentials, by sending the credentials back to the hacker's own web server.
- The hacker could also gain control over the underlying system, based on the privileges of the user whose session cookie has been stolen.
- The hacker's text and script might be displayed to other people on any JIRA page which has a form. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade your JIRA Charting plugin to version 1.4.1 to fix the vulnerabilities described below.

Risk Mitigation

We recommend that you upgrade your JIRA Charting plugin as soon as possible. If you judge it necessary, you can disable public access (i.e. anonymous access and public signup) to your JIRA system. For even tighter control, you could restrict JIRA access to trusted groups only.

Vulnerability

JIRA instances that use the JIRA Charting plugin (any version) are vulnerable to this security flaw.

Fix

The fix is to HTML encode the appropriate values in the JIRA Charting plugin actions. Please see JCHART-256 and JCHART-257 for further details.

This issue has been fixed in the JIRA Charting plugin 1.4.1 or later. Please see the plugin page to check compatibility with your JIRA version.

Please let us know what you think of the format of this security advisory and the information we have provided.

JIRA Development Hub

Looking for plugins? To find existing plugins and extensions written by the community, visit the Atlassian Plugin Exchange and the JIRA Extensions space.

This documentation applies to JIRA 4.0. If you are developing a plugin for JIRA 3.x, please see the previous documentation.

Getting started developing with JIRA

So you're thinking of customizing JIRA? Maybe you want to change the way JIRA looks? Or create a new custom field type? Or add some new Workflow transitions? Well, you've come to the right place. Here are all the resources to get started developing for JIRA.

1. Getting started with Atlassian plugins

This tutorial will show you how to set up your development environment, create an empty plugin template, and the basic principles of building, debugging, and testing a plugin. It will take you through the prerequisites and introduce you to some of the resources that Atlassian provides for plugin developers.

2. The JIRA Plugin Guide

These documents are specifically about plugins for JIRA. There's a page for each plugin module type that JIRA supports. You can combine multiple plugin modules inside a single plugin to accomplish complex tasks.

3. Understanding how JIRA works

These documents go some of the way to explaining what's really going on inside JIRA. Some of this information is useful to plugin developers. Other pieces are more relevant to the JIRA development team, but we've published them here in the spirit of open documentation.

Want to modify the JIRA source code?

You'll want to download a copy of the source distribution and then follow the instructions in Building JIRA from Source.

1. Building JIRA from Source
2. Developer Tutorials

Help and documentation

- The JIRA Documentation
- The JIRA API
- The JIRA Developer FAQ
- The JIRA Developer Forums
- The JIRA Community Space
- Workflow Function plugin tutorial — Vincent Massol's excellent tutorial
- Atlassian Partner listing

JIRA Plugin Guide

JIRA includes a plugin system that enables developers to write plugins which enhance JIRA's functionality in various ways.

Setting up a Plugin Project

Please refer to How to build an Atlassian Plugin to set up your development environment and create a plugin template.

A Plugin Overview

A JIRA plugin is a single JAR containing code, a plugin descriptor (XML) and usually some Velocity template files to render HTML.

The plugin descriptor is the only mandatory part of the plugin. It must be called **atlassian-plugin.xml** and be located in the root of your JAR file.

Each plugin consists of one or more **plugin modules**. These are of different types (for example a report, or a portlet) and each has an individual XML element describing it. Each module is described below together with the XML element required for it.

Here is a sample of the descriptor with highlighted elements:

```
<!-- the plugin key must be unique, think of it as the 'package' of the plugin -->
<atlassian-plugin key="com.atlassian.plugin.sample" name="Sample Plugin">
    <!-- a short block describing the plugin itself -->
    <plugin-info>
        <description>This is a brief textual description of the plugin</description>
        <!-- the version of the plugin -->
        <version>1.1</version>
        <!-- the versions of the application this plugin is for -->
        <application-version min="3.0" max="3.0"/>
        <!-- details of the plugin vendor -->
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
    </plugin-info>

    . . . 1 or more plugin modules . . .
</atlassian-plugin>
```

Each plugin has a **plugin key** which is unique among all plugins (eg "com.atlassian.plugin.sample"). Semantically this equates to the package of a Java class. Each module within the plugin also has a **module key** which is unique within the plugin (eg "myreport"). Semantically this equates to the class name of a Java class.

The plugin key + module key are combined to make the **complete key** of the plugin module (combining the examples above, the complete key would be "com.atlassian.plugin.sample:myreport"). Note: a : is used to separate the plugin key from the module key.

JIRA Plugin Module Types

The following types of plugin modules are supported by JIRA

Module Type	Since version...	Documentation	Description
portlet	3.0	Portlet Plugin Module	Add new portlets to JIRA
report	3.0	Report Plugin Module	Add new reports to JIRA
customfield-type	3.0	Custom Field Plugin Module	Add new types of fields to JIRA
customfield-searcher	3.0	Custom Field Plugin Module	Add new types of field searchers to JIRA
project-tabpanel	3.0	Project Tab Panel Plugin Module	Add new tabs to the Browse 'Project' screen
component-tabpanel	3.10	Component Tab Panel Plugin Module	Add new tabs to the Browse 'Component' screen
version-tabpanel	3.10	Version Tab Panel Plugin Module	Add new tabs to the Browse 'Version' screen
issue-tabpanel	3.0	Issue Tab Panel Plugin Module	Add new tabs to the View Issue screen
issue-operation	3.4	Issue Operations Plugin Module	Add new operations to the View Issue screen
resource	3.5	Downloadable Plugin Resources	Downloadable resources from within any plugin
servlet	3.5	Servlet Plugin Module	A standard Java servlet deployed within a JIRA plugin
webwork	3.1	Webwork plugin module	XWork/Webwork actions and views bunded with a plugin, enabling user interaction

component	3.0	Component Plugin Module	Adds components to JIRA's component system
rpc-soap	3.0	RPC Endpoint Plugin Module	Deploys a SOAP service within JIRA
rpc-xmlrpc	3.0	RPC Endpoint Plugin Module	Deploys an XML-RPC service within JIRA
search-request-view	3.7	Search Request View Plugin Module	Add a new view in the Issue Navigator
workflow-condition	3.0	Workflow Plugin Modules	Add new conditions to the JIRA workflow
workflow-validator	3.0	Workflow Plugin Modules	Add new validations to the JIRA workflow
workflow-function	3.0	Workflow Plugin Modules	Add new post functions to the JIRA workflow
web-item	3.7	Web UI Plugin Module	Add new links into the Jira web interface
web-section	3.7	Web UI Plugin Module	Add new tabs/sections into the Jira web interface

Built-in JIRA system plugins

A number of functions and areas within JIRA 3 are shipped as built in plugins. These can also be useful for plugin developers who want to know more about how to create their own plugins, as they showcase the functionality that can be built.

The system plugins are referenced from the following files (located in `/WEB-INF/classes`:

- `system-workflow-plugin.xml` - the built in workflow conditions, validators and functions.
- `system-customfieldtypes-plugin.xml` - the built in custom field types.
- `system-project-plugin.xml` - the built in project tab panels (ie roadmap, change log and popular issues).
- `system-reports-plugin.xml` - the built in system reports (ie time tracking and developer workload reports).
- `system-portlets-plugin.xml` - all of the built in system portlets.

Deploy a JIRA Plugin

Installing plugins in JIRA is easy.

Once you have downloaded or created your plugin jar, follow these steps:

1. Shut down JIRA
2. copy '$MY_COOL_PLUGIN.jar' into '.../atlassian-jira/WEB-INF/lib/'
3. Start up JIRA.
4. Go to 'Administration > Plugins' and confirm that $MY_COOL_PLUGIN is listed and enabled.
5. Enjoy!

Component Plugin Module

A component plugin module defines a Java component which will be injected into JIRA's internal component system.

> ⚠ JIRA's internal components are managed by PicoContainer.

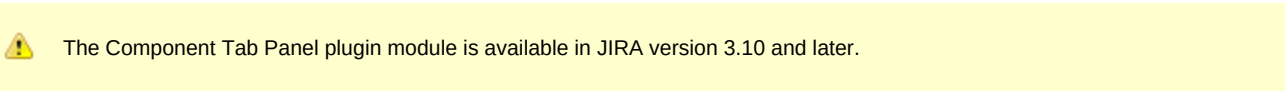
A new component is simple to define as follows:

```
<component key="userService" name="User Service"
    class="com.atlassian.jira.rpc.soap.UserServiceImpl">
    <interface>com.atlassian.jira.rpc.soap.UserService</interface>
</component>
```

This example here defines a component implementing UserService that is put into the PicoContainer to inject into any other plugin modules. If you include this component module in your plugin, then your other plugin modules can define a constructor with a UserService parameter and this implementation will be provided to your plugin module automatically.

These components allow you to simplify the creation and management of your plugin modules quite a lot.

Component Tab Panel Plugin Module



The Component Tab Panel plugin module is available in JIRA version 3.10 and later.

The Component Tab Panel plugin module allows you to add new tabs to the 'Browse Component' page.

All these tab panels ('Open Issues', 'Road Map', 'Change Log' and 'Popular Issues') are implemented as plugins. New component tab panels can be implemented to display component-specific info.

Here is an example panel module descriptor:

```
<!--
The class defined should implement
com.atlassian.jira.plugin.componentpanel.ComponentTabPanel
and it may be useful to use the functionality provided by
com.atlassian.jira.plugin.componentpanel.impl.GenericTabPanel.
-->
    <component-tabpanel key="component-openissues-panel" i18n-name-key=
"componentpanels.openissues.name"
                        name="Open Issues Panel" class=
"com.atlassian.jira.plugin.componentpanel.impl.GenericTabPanel">
        <description key="componentpanels.openissues.description">Show the open issues for this
component.</description>
        <label key="common.concepts.openissues"/>
        <!-- this is a number defining the order of all panels.
             The system panels are 10, 20, 30 and 40. -->
        <order>10</order>
        <!-- this template produces the HTML for the panel -->
        <resource type="velocity" name="view"
                  location="templates/plugins/jira/projectentitypanels/openissues-component-panel.vm"
/>
        <!-- this properties files contains i18n keys for the panel -->
        <resource type="i18n" name="i18n" location=
"com.atlassian.jira.plugins.componentpanels.openissues"/>
    </component-tabpanel>
```

Configuring Plugins with Object Configurable Parameters

JIRA Report and Portlet plugins use 'Object Configurable' atlassian module to request configuration parameters form users. These parameters are described in the atlassian-plugin.xml as part of the plugin entry. For example, for the Time Tracking Report report that ships with JIRA the input parameters are:

```
<properties>
      <property>
          <key>versionId</key>
          <name>common.concepts.version</name>
          <description>report.timetracking.version.description</description>
          <type>select</type>
          <values class="com.atlassian.jira.portal.VersionOptionalValuesGenerator"/>
      </property>
      <property>
          <key>sortingOrder</key>
          <name>report.timetracking.sortingorder</name>
          <description>report.timetracking.sortingorder.description</description>
          <type>select</type>
          <values class="com.atlassian.jira.portal.SortingValuesGenerator"/>
      </property>
      <property>
          <key>completedFilter</key>
          <name>report.timetracking.filter</name>
          <description>report.timetracking.filter.description</description>
          <type>select</type>
          <values class="com.atlassian.jira.portal.FilterValuesGenerator"/>
      </property>
   </properties>
```

Types

The various `<types>` that can be used are:

Type	Input HTML Type
string	text box
long	text box
select	select box
multiselect	multi-select box
hidden	hidden field
date	text box with calendar pop-up
user	text box with user picker pop-up
text	text area

(for an example of how to add new types, see the Adding checkbox as a type example)

Values

Values can be provided by a value provider class that must subclass ValuesGenerator.

Or.

You can hard code the values into the descriptor in the form:

```
<values>
  <value>
    <key>KEY1</key>
    <value>somevalue</value>
  </value>
  <value>
    <key>KEY2</key>
    <value>someothervalue</value>
  </value>
</values>
```

Defaults

You can specify a default for the following types: User, Long and Date.

The format for this is:

```
<default>5</default>
```

Adding checkbox as a type

This page provides an example of how to extend the available types listed on the Configuring Plugins with Object Configurable Parameters reference page.

JIRA Source Note:

You must own a commercial license of JIRA to have access to JIRA's source code-- a necessary part of this modification.

In the Atlassian JIRA Source modify the following:

ObjectConfigurationTypes.java

```
public class ObjectConfigurationTypes
{
   public static final int CHECKBOX = 9;
      ...

   public static int getType(String typeStr)
   {
      ...

      else if (typeStr.toLowerCase().startsWith("checkbox"))
      {
          return CHECKBOX;
      }

      ...
   }

}
```

objectconfiguration_form.jsp

```
<webwork:elseIf test="objectConfiguration/fieldType(.)==9"><%-- CHECKBOX --%>
    <webwork:property value="." />
    <webwork:property value="paramValue(.)" />
    <ui:component name="." value="paramValue(.)" label="text(objectConfiguration/fieldName(.))"
template="checkbox.jsp">
        <ui:param name="'description'"><webwork:text name="objectConfiguration/fieldDescription(.)"
/></ui:param>
        <ui:param name="'fieldValue'"><webwork:property value="paramValue(.)" /></ui:param>
    </ui:component>
</webwork:elseIf>
```

Example Plugin Code

This example uses the Issue Creation Report example (com.atlassian.jira.plugin.report.example) provided in the JIRA Plugin Development Kit.

In your plugin source (e.g. a report plugin) add the following property:

atlassian-plugin.xml

```
<property>
   <key>checkbox</key>
   <name>report.issuecreation.checkbox</name>
   <description>report.issuecreation.checkbox.description</description>
   <type>checkbox</type>
   <default>true</default>  //If you would like the default to be checked
</property>
```

Generally, your atlassian-plugin.xml will refer to a properties file (e.g. issuecreation_report.properties) for configuration of field labels and descriptions. Add the following attributes:

issuecreation_report.properties

```
report.issuecreation.checkbox = CheckBox Label
report.issuecreation.checkbox.description = A simple checkbox for test purposes.
```

Modify your plugin java file to extract/work with the boolean checkbox value:

CreationReport.java

```
...

public String generateReportHtml(ProjectActionSupport action, Map params) throws Exception
{
        User remoteUser = action.getRemoteUser();
        I18nHelper i18nBean = new I18nBean(remoteUser);

        // Retrieve the project parameter
Long projectId = ParameterUtils.getLongParam(params, "projectid");
        // Retrieve the start and end dates and the time interval specified by the user
Date startDate = ParameterUtils.getDateParam(params, "startDate", i18nBean.getLocale());
        Date endDate = ParameterUtils.getDateParam(params, "endDate", i18nBean.getLocale());
        Long interval = ParameterUtils.getLongParam(params, "interval");
        boolean checkbox = (Boolean.valueOf(ParameterUtils.getStringParam(params, "checkbox"
))).booleanValue();
     ...

        velocityParams.put("checkbox", new Boolean(checkbox));
 }
   ...
```

Custom Field Plugin Module

The custom field plugin module allows you to add new custom field types and searchers to JIRA.

1. Custom field types - these define the type of a custom field
2. Custom field seachers - these define the ways a given type can be searched

Types and searchers can be combined in different ways to produce new custom fields, for example a "user" custom field could take a simple text searcher (to enter the username as text) or a more complex "user picker searcher" (where the user is picked from a popup window).

Here is an example of their respective descriptor elements:

```
<!--
   The module class of a type must implement:
   com.atlassian.jira.issue.customfields.CustomFieldType
   and there are a number of abstract classes that can be extended,
   for details see the bundled system custom field types
-->
<customfield-type key="userpicker" name="User Picker"
class="com.atlassian.jira.issue.customfields.impl.UserCFType">
   <description>
      Choose a user from the user base via a popup picker window.
   </description>

   <!-- this template is used on the view issue page -->
   <resource type="velocity" name="view"
   location="templates/plugins/fields/view-user.vm" />
   <!-- this template is used on the create/edit issue pages -->
   <resource type="velocity" name="edit"
   location="templates/plugins/fields/edit-userpicker.vm" />
   <!-- this template is used when viewing an issue as XML -->
   <resource type="velocity" name="xml"
   location="templates/plugins/fields/xml-user.vm" />
</customfield-type>

<!--
   The module class of a searcher must implement:
   com.atlassian.jira.issue.customfields.CustomFieldSearcher
   and again the are a number of useful abstract classes bundled.
-->
<customfield-searcher key="userpickersearcher" name="User Picker Searcher"
class="com.atlassian.jira.issue.customfields.searchers.UserPickerSearcher">
   <description>
      Allow to search for a user using a userpicker.
   </description>

   <!-- this template is used on the issue navigator search form -->
   <resource type="velocity" name="search"
   location="templates/plugins/fields/search-userpicker.vm" />
   <!-- this element defines the valid custom field types for this searcher -->
   <valid-customfield-type
   package="com.atlassian.jira.plugin.system.customfieldtypes" key="userpicker" />
</customfield-searcher>
```

For more details, see the How to create a new Custom Field Type.

To learn more about the custom field Velocity templates, see Custom field Velocity context unwrapped

Custom field Velocity context unwrapped

Below is a list of available objects in the Velocity context for custom fields.

Object	Description	edit	view	col-view
customField	`com.atlassian.jira.issue.fields.CustomField` object. Information on the current field.	✓	✓	✓
issue	`com.atlassian.jira.issue.Issue` object. Pretty much everything about the issue.	✓	✓	✓

fieldLayoutItem	`com.atlassian.jira.issue.fields.layout.field.FieldLayoutItem`. Is field required? hidden?	✓	✗	✗
action	Calling action.	✓	✗	✗
displayParameters	Custom paramaters to the template, such as whether to display headers or not	✓	✗	✗
value	String value of the custom field. Thus this is a String for Date and other single valued fields, List of Strings for Multi selects and CustomFieldParams full of Strings for Cascading selects	✓	✓	✓
customFieldParams	This is where the value is pulled from, for convenience	✓	✓	✓
config	`com.atlassian.jira.issue.customfields.config.CustomFieldConfig` object.	✓	✗	✗
configs	The various configuration items for that context. This include things like, default values, select list options and other configurable options	✓	✗	✗
i18n	`com.atlassian.jira.web.bean.I18nBean` for your internationalisation needs	✓	✓	✓
descriptor	The module descriptor of the current field	✓	✓	✓
textutils	`com.opensymphony.util.TextUtils` for text manipulation needs	✓	✓	✓
outlookdate	`com.atlassian.jira.web.util.OutlookDate` for formatting dates, JIRA style	✓	✓	✓
authcontext	`com.atlassian.jira.security.JiraAuthenticationContext` for authentication information	✓	✓	✓
dateutils	`com.atlassian.core.util.DateUtils` more date functions	✓	✓	✓
req	`HttpServletRequest` object.	✓	✓	✓
baseurl	The `getContextPath` of the req object	✓	✓	✓
constantsManager	The `com.atlassian.jira.config.ConstantsManager` object for managing "constants" (issue types, resolutions etc.)	✓	✓	✓
projectManager	`com.atlassian.jira.project.ProjectManager`	✓	✓	✓
applicationProperties	`com.atlassian.jira.config.properties.ApplicationProperties`	✓	✓	✓
jirautils	`com.atlassian.jira.util.JiraUtils` a random set of methods. has isPublic method	✓	✓	✓
jirakeyutils	`com.atlassian.jira.util.JiraKeyUtils` an object used for parsing keys	✓	✓	✓
buildutils	`com.atlassian.jira.util.BuildUtils` has information on build numbers, editions etc.	✓	✓	✓
velocityhelper	`com.atlassian.jira.util.JiraVelocityHelper` random set of utility methods	✓	✓	✓
userutils	`com.atlassian.core.user.UserUtils` untility for getting users	✓	✓	✓

+ Other velocity parameters from the custom field type

How to add searching to a Custom Field?

Custom fields need a **custom field searcher** before you can search them in the Issue Navigator.

If a custom field type is a going to behave somewhat like an existing type, then you can reuse an existing searcher. For example, if you're creating a custom field type that's like a multi select list and want to reuse the default multi-select searcher, you could add the block below to your atlassian-plugin.xml (taken from `system-customfieldtypes-plugin.xml`)

```
<customfield-searcher key="multiselectsearcher" name="Multi Select Searcher"
    i18n-name-key="admin.customfield.searcher.multiselectsearcher.name"
    class="com.atlassian.jira.issue.customfields.searchers.MultiSelectSearcher">
    <description key="admin.customfield.searcher.multiselectsearcher.desc">Search for multiple values
using a single select list.</description>

    <resource type="velocity" name="search" location=
"templates/plugins/fields/edit-searcher/search-multiselect.vm"/>
    <resource type="velocity" name="view" location=
"templates/plugins/fields/view-searcher/view-searcher-multioption.vm"/>
    <resource type="velocity" name="label" location=
"templates/plugins/fields/view-searcher/label-searcher-basictext.vm"/>
    <valid-customfield-type package="YOUR CUSTOM PACKAGE" key="YOUR CUSTOM FIELD KEY"/>
</customfield-searcher>
```

which will enable searching for any text based, multi-select custom field type. When creating the custom field, you will now be able to select the multi-select searcher and your custom field should then be displayed in the Issue Navigator.

Note that "YOUR CUSTOM PACKAGE" refers to the package (ie. the module key) of the custom field that the searcher applies to.

Downloadable Plugin Resources

JIRA plugins may define downloadable resources. If your plugin requires JIRA to serve additional static files such as images, Javascript or CSS, you will need to use downloadable plugin resources to make them available.

This is only available as of JIRA 3.5 and above.

Defining a Single Downloadable Resource

Downloadable resources are configured to map a `name` of some downloadable file to its `location` within the plugin jar-file.

```
<atlassian-plugin name='IM Presence Macros' key='confluence.extra.impresence'>
    <plugin-info>
        <description>Macros to show online status for popular Instant Messaging services.
</description>
        <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
        <version>0.1</version>
    </plugin-info>
    <macro name='aim' class='com.atlassian.confluence.extra.impresence.AIMPresenceMacro' key='aim'>
        <description>Displays an AIM status graphic.</description>
        <resource type="download" name="aimon.gif" location="templates/extra/impresence/aimon.gif">
            <param name="content-type" value="image/gif"/>
        </resource>
    </macro>

    <resource type="download" name="aimoff.gif" location="templates/extra/impresence/aimoff.gif"/>
</atlassian-plugin>
```

- Resources can be downloaded either within a plugin module, or as a resource of the entire plugin.
    - Resources are always looked up relative to a plugin module (see below). If a resource can not be found in the plugin module, it will then be looked for in the plugin itself.
- Each resource must be of `type="download"`
- The `name` of the resource is how it will be referenced from within the application
- The `location` of the resource is where it appears within the plugin itself
- An optional `content-type` parameter can be used to supply the file's MIME type
    - In the absence of a `content-type`, the application will attempt to guess the file's type from its file extension. For common file extensions, an explicit content-type is not necessary.

Defining a Directory of Downloadable Resources

If your plugin requires a lot of resources, you may wish to expose a directory of files as resources, rather than writing definitions for each individual file.

```
<resource type="download" name="icons/" location="templates/extra/impresence/icons/"/>
```

- The `name` and `location` must both have trailing slashes
- Subdirectories are also exposed, so in the example above, `icons/small/mail.gif` will be mapped to the resource `templates/extra/impresence/icons/small/mail.gif`

Referring to Downloadable Resources

The URL for a downloadable resource is as follows:

```
{server root}/download/resources/{plugin key}:{module key}/{resource name}
```

For example:

```
http://confluence.example.com/download/resources/confluence.extra.impresence:aim/aimon.gif
```

In a velocity template, you should use the `$req.contextPath` property to ensure that your resources are always relative to the URL of the Confluence server:

```
$req.contextPath/download/resources/confluence.extra.impresence:aim/aimon.gif
```

Installing and Configuring Plugins

Installing plugins in JIRA is easy.

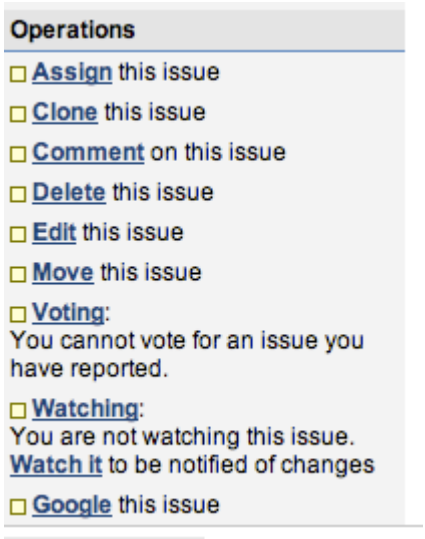
Once you have downloaded or created your plugin jar, follow these steps:

1. Shut down JIRA
2. copy '$MY_COOL_PLUGIN.jar' into '.../atlassian-jira/WEB-INF/lib/'
3. Start up JIRA.
4. Go to 'Administration > Plugins' and confirm that $MY_COOL_PLUGIN is listed and enabled.
5. Enjoy!

Issue Operations Plugin Module

The Issue Operations plugin module allows you to add new operations to the 'View Issue' screen.

Issue Operation plugin modules are available in JIRA 3.4 and later.



You can add new operations with a plugin, linking to information about a single issue (most likely pulled from an external source).

Simple Example

Here is an example descriptor that adds a link to Google a given issue's summary:

```
<!--
The module class should implement
com.atlassian.jira.plugin.issueoperation.PluggableIssueOperation
-->
<issue-operation key="google-summary" name="Google this issue"
class="com.atlassian.jira.plugin.issueoperation.DefaultPluggableIssueOperation">
<resource type="velocity" name="view">
&lt;img src="$req.contextPath/images/icons/bullet_creme.gif" height=8 width=8 border=0
align=absmiddle&gt;
&lt;b&gt;&lt;a href="http://www.google.com/search?q=${issue.summary}"&gt;Google&lt;/a&gt;&lt;/b&gt;
issue summary
</resource>
<!-- the relative order of operations -->
<order>10</order>
</issue-operation>
```

Issue operations are very useful as a 'hook' to link to your other plugin components - such as Webwork actions, project tab panels, reports etc.

Notes

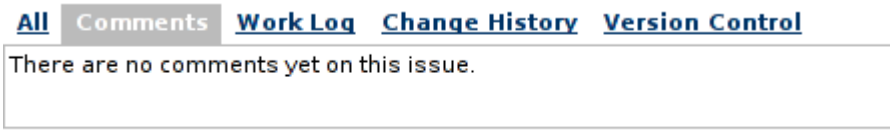
- All issue operation classes implements `PluggableIssueOperation`.
- The `showOperation(Issue issue)` method allows you to turn show or hide operations for certain users, permissions etc.
- `DefaultPluggableIssueOperation` is a useful base that should work for most simple HTML operations. See for example the Canned Response issue operation plugin
- The view velocity resource can point to files like other plugins e.g.

  ```
  <resource type="velocity" name="view" location="templates/more-operations.vm"/>
  ```

- All pluggable issue operations occur after system issue operations (i.e. order is among all pluggable operations, not all operations)

Issue Tab Panel Plugin Module

The Issue Tab Panel plugin module allows you to add new tab panels to the View Issue screen.



You can add a new tab with a plugin, displaying information about a single issue (most likely pulled from an external source). This is how the Subversion and Perforce integration works.

Here is an example descriptor (from the Subversion plugin - source available):

```
<!--
The module class should implement
com.atlassian.jira.plugin.issuetabpanel.IssueTabPanel
-->
    <issue-tabpanel key="subversion-commits-tabpanel" name="Subversion Changes Tab Panel"
        class=
"com.atlassian.jira.plugin.ext.subversion.issuetabpanels.changes.SubversionRevisionsTabPanel">
        <resource type="i18n" name="i18n" location=
"templates.plugins.subversion.issuetabpanels.subversion_jira_plugin"/>
        <description>Show Subversion commits related to an issue in an issue tab panel.</description>
        <label>Subversion Commits</label>
        <resource type="velocity" name="view" location=
"templates/plugins/subversion/issuetabpanels/subversion-commits-view.vm"/>
    </issue-tabpanel>
```

Portlet Plugin Module

In JIRA 4, portlets have been superceded by gadgets.
For details, please see Gadgets and JIRA Portlets.

A portlet plugin module defines a portlet that users can add to their Dashboards.
The properties that it can take are the same as the report plugin module above.

Here is a sample portlet module descriptor element:

```
<!--
The module class must implement
com.atlassian.jira.portal.Portlet
and will usually extend:
com.atlassian.jira.portal.PortletImpl
-->
<portlet key="assignedtome" name="Assigned Issues"
    class="com.atlassian.jira.portal.portlets.AssignedToMePortlet">
    <description key="portlet.assignedtome.description">i18n description</description>
    <!-- this template produces the eventual HTML of the portlet -->
    <resource type="velocity" name="view"
    location="templates/plugins/jira/portlets/assignedtome.vm" />
    <label key="portlet.assignedtome.name" />
    <!-- an optional thumbmail image used to preview the portlet for users -->
    <thumbnail>portlets/dashboard/thumbnails/assigned.gif</thumbnail>
    <!-- the permissions required to add this portlet
    (optional - not often required) -->
    <permission>assignable</permission>
    <objectdescriptor key="portlet.assignedtome.display.name" />
    <!-- same as the properties of the report plugin module -->
    <properties>
        <property>
            <key>numofentries</key>
            <name>portlet.assignedtome.field.numofentries.name</name>
            <description>
            portlet.assignedtome.field.numofentries.description
            </description>
            <type>long</type>
            <default>10</default>
        </property>
    </properties>
</portlet>
```

For more details, see the How to create a JIRA Portlet.

Quick start guide for creating a portlet plugin

In JIRA 4, portlets have been superceded by gadgets.
For details, please see Gadget Development.

Goal

This page is intented to get you up and running with writing your own JIRA portlet within no time. It actually is a combination of several examples included in the plugin development kit which were broken. The portlet which is explained below lets you configure 3 parameters and will result in table of books that match the searchcriterium:

1. subscriptionId (Needed to use an Amazon webservice)
2. searchParameter (What book property will we be looking for ? E.g. Title, Author, ...)
3. searchArgument (The argument which will be searched for. E.g. "Java" or "Ruby")

Screenshot 1: Configuration of portlet

Results of Book Search by Author 'Sierra'	
Head First Design Patterns (Head First)	Elisabeth Freeman
Head First Java, 2nd Edition	Kathy Sierra
SCJP Sun Certified Programmer for Java 5 Study Guide (Exam 310-055) (Certification Press Study Guides)	Katherine Sierra
Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam (Head First)	Bryan Basham
WOLF	Dakota Cassidy
Head First EJB (Brain-Friendly Study Guides; Enterprise JavaBeans)	Kathy Sierra
Wild About Books (Irma S and James H Black Honor for Excellence in Children's Literature (Awards))	Judy Sierra
Miami Vice (1984-89 Television Series)	
Dark Side	Angela Knight
Sierra Club 2008 Engagement Calendar	Sierra Club

Screenshot 2: result of search (parameter = 'Author' & argument = 'Sierra')

Results of Book Search by Title 'java'	
Head First Java, 2nd Edition	Kathy Sierra
SCJP Sun Certified Programmer for Java 5 Study Guide (Exam 310-055) (Certification Press Study Guides)	Katherine Sierra
Regular Expression Pocket Reference: Regular Expressions for Perl, Ruby, PHP, Python, C, Java and .NET (Pocket Reference (O'Reilly))	Tony Stubblebine
Java Persistence with Hibernate	Christian Bauer
Java Concurrency in Practice	Brian Goetz
Java How to Program (7th Edition) (How to Program)	Harvey & Paul) Deitel & Associates Inc.
Core Java(TM), Volume I--Fundamentals (8th Edition)	Cay S. Horstmann
Groovy Recipes: Greasing the Wheels of Java (Pragmatic Programmers)	Scott Davis
Murach's Java Servlets and JSP, 2nd Edition	Joel Murach
Java In A Nutshell, 5th Edition	David Flanagan

Screenshot 3: result of search (parameter = 'Title' & argument = 'Java')

Prequisites

My installed software:

1. JDK 1.6.0_03
2. Eclipse 3.3.0
3. Maven 2.0.7
4. Jira enterprise standalone edition 3.12.2

Configuration steps

You will need to make some changes to your $maven_home/conf/settings.xml.

```
<profiles>
    <profile>
      <id>Ciber</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>

      <repositories>
        ...
        <!-- JIRA -->
        <repository>
          <id>atlassian-public</id>
          <url>https://maven.atlassian.com/repository/public</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
        <repository>
          <id>atlassian-contrib</id>
          <url>https://maven.atlassian.com/contrib</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
        <repository>
          <id>atlassian-m1-repository</id>
          <url>https://maven.atlassian.com/maven1</url>
          <layout>legacy</layout>
        </repository>
      </repositories>

      <pluginRepositories>
        ...
        <!-- JIRA -->
        <pluginRepository>
          <id>atlassian-public</id>
          <url>https://maven.atlassian.com/repository/public</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
        <pluginRepository>
          <id>atlassian-contrib</id>
          <url>https://maven.atlassian.com/contrib</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
       </pluginRepository>
      </pluginRepositories>

      <properties>
        <downloadSources>false</downloadSources>
        <downloadJavadocs>false</downloadJavadocs>
        <atlassian.pdk.server.url>http://localhost:8080</atlassian.pdk.server.url>
```

```
  <atlassian.pdk.server.username>********</atlassian.pdk.server.username>
  <atlassian.pdk.server.password>********</atlassian.pdk.server.password>
</properties>
```

```
    </profile>
  </profiles>
```

See also "how to build an atlassian plugin".

Next we will create a new plugin project based on a plugin archetype. In a DOS-box, run following command:

```
mvn org.apache.maven.plugins:maven-archetype-plugin:1.0-alpha-7:create \
  -DarchetypeGroupId=com.atlassian.maven.archetypes \
  -DarchetypeArtifactId=jira-plugin-archetype \
  -DarchetypeVersion=10 \
  -DremoteRepositories=https://maven.atlassian.com/repository/public/ \
  -DgroupId=$MY_PACKAGE$ -DartifactId=$MY_PLUGIN$
```

and replace $MY_PACKAGE$ and MY_PLUGIN according to your project. Make sure you remove the backslashes "\" so the complete command is one 1 line before you execute it.

Maven will create a new folder MY_PLUGIN and provide some default files. Check the attachment for how the plugin works.

Cheers,

Robby

Project Tab Panel Plugin Module

The Project Tab Panel plugin module allows you to add new tabs to the 'Browse Projects' page.

All these tab panels ('Road Map', 'Change Log', 'Popular Issues' and 'Open Issues') are implemented as plugins. New project tab panels can be implemented to display project-specific info.

Here is an example panel module descriptor:

```
<!--
The class defined should implement
com.atlassian.jira.plugin.projectpanel.ProjectTabPanel
and it may be useful to use the functionality provided by
com.atlassian.jira.plugin.projectpanel.impl.GenericProjectTabPanel.
-->
<project-tabpanel key="roadmap-panel" name="Road Map Panel"
    class="com.atlassian.jira.plugin.projectpanel.impl.VersionsProjectTabPanel">
    <description key="projectpanels.roadmap.description">
    A roadmap of the upcoming versions in this project.
    </description>
    <label key="common.concepts.roadmap" />
    <!-- this is a number defining the order of all panels.
    The system panels are 10, 20, 30 and 40. -->
    <order>20</order>
    <!-- this template produces the HTML for the panel -->
    <resource type="velocity" name="view"
    location="templates/plugins/jira/projectpanels/roadmap-panel.vm" />
    <!-- this properties files contains i18n keys for the panel -->
    <resource type="i18n" name="i18n"
    location="com.atlassian.jira.plugins.projectpanels.roadmap" />
</project-tabpanel>
```

Report Plugin Module

A report plugin module defines a report within JIRA. Reports have HTML results, and optionally Excel results as well. These results are rendererd by Velocity templates included with the plugin. A report can also take in parameters selected by the user before running.

Here is a sample report plugin module definition:

```
<!--
The "class" attribute here defines the actual report object.
This is an implementation of com.atlassian.jira.plugin.report.Report.
There is also an abstract implementation which may be useful at
com.atlassian.jira.plugin.report.impl.AbstractReport.
-->
<report key="time-tracking" name="Time Tracking Report"
    class="com.atlassian.jira.plugin.report.impl.TimeTrackingReport">
    <description key="report.timetracking.description">
        This report shows the time tracking details for a specific project.
    </description>

    <!-- the label of this report, which the user will use to select it -->
    <label key="report.timetracking.label" />

    <!-- the 'view' template is used to render the HTML result -->
    <resource type="velocity" name="view"
        location="templates/plugins/jira/reports/time-tracking-report.vm" />
    <!--
        The 'excel' template is used to render an Excel result.
        The 'Excel view' of the report will only be visible if
        this template exists for the plugin module
    -->
    <resource type="velocity" name="excel"
        location="templates/plugins/jira/reports/time-tracking-report-excel.vm" />

    <!-- this is a .properties file containin the i18n keys for this report -->
    <resource type="i18n" name="i18n"
        location="com.atlassian.jira.plugins.reports.timetracking" />

    <!-- the properties of this report which the user must select before running it -->
    <properties>
        <property>
            <key>versionId</key>
            <name>common.concepts.version</name>
            <description>report.timetracking.version.description</description>
            <!-- valid types are string, text, long, select, date -->
            <type>select</type>
            <!-- the values generator is a class which will
            generate values for this select list -->
            <values class="com.atlassian.jira.portal.VersionOptionalValuesGenerator"/>
        </property>
        <property>
            <key>sortingOrder</key>
            <name>report.timetracking.sortingorder</name>
            <description>report.timetracking.sortingorder.description</description>
            <type>select</type>
            <values class="com.atlassian.jira.portal.SortingValuesGenerator"/>
        </property>
        <property>
            <key>completedFilter</key>
            <name>report.timetracking.filter</name>
            <description>report.timetracking.filter.description</description>
            <type>select</type>
            <values class="com.atlassian.jira.portal.FilterValuesGenerator"/>
        </property>
    </properties>
</report>
```

For more details, see the How to create a JIRA Report.

RPC Endpoint Plugin Module

The RPC endpoint plugin modules enable you to publish new SOAP and XML-RPC endpoints within JIRA.

These endpoints allow you to expose your own remote web services to the outside world.

Here are two example RPC endpoint descriptors:

```
<rpc-soap key="soap" name="System SOAP Services"
    class="com.atlassian.jira.rpc.soap.JiraSoapServiceImpl">
    <description>The standard JIRA SOAP services.</description>
    <service-path>jiraservice-v1</service-path>
    <published-interface>
        com.atlassian.jira.rpc.soap.JiraSoapService
    </published-interface>
</rpc-soap>

<rpc-xmlrpc key="xmlrpc" name="System XML-RPC Services"
    class="com.atlassian.jira.rpc.xmlrpc.JiraXmlRpcService">
    <description>The standard JIRA XML-RPC services.</description>
    <service-path>jira1</service-path>
</rpc-xmlrpc>
```

The first defines a SOAP service, the second an XML-RPC service. These objects (class="x") will be published via Glue or Apache XML-RPC respectively. The **`service-path`** element defines where in the URL namespace the services will be published. The **`published-interface`** element for the SOAP module defines which interface will be published (XML-RPC publishes all methods of the object).

Please take note that you also have to create a Component Plugin Module to avoid the client getting `NullPointerException`, e.g.

```
<component key="component-FoobarSoapService" name="Foobar SOAP Service Component" class=
"com.atlassian.jira.rpc.FoobarSoapServiceImpl">
    <interface>com.atlassian.jira.rpc.FoobarSoapService</interface>
</component>

<rpc-soap key="soap-foobar" name="Foobar SOAP Service" class=
"com.atlassian.jira.rpc.FoobarSoapServiceImpl">
    <description>Foobar SOAP service.</description>
    <service-path>foobarservice</service-path>
    <published-interface>com.atlassian.jira.rpc.FoobarSoapService</published-interface>
</rpc-soap>
```

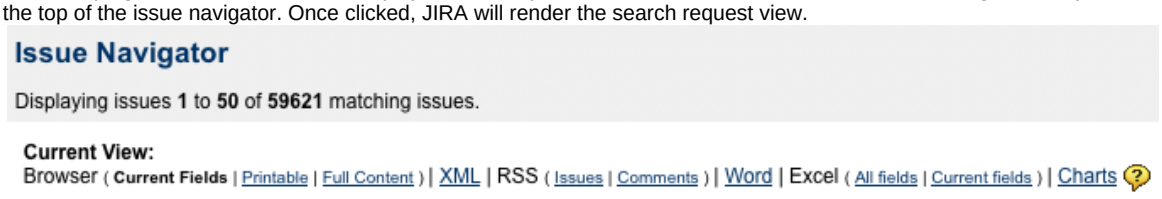
You can learn more about RPC plugins from looking at JIRA's system RPC plugin. The source to this plugin is also freely available, and serves as an excellent learning resource.

You may also wish to see the Creating a XML-RPC Client and Creating a SOAP Client if you're interested in creating a JIRA remote client.

Search Request View Plugin Module

Search request view plugin modules were implemented in the JIRA 3.7 release. They are not available in previous releases.

Search request view plugin modules are used to display different representations of search results in the issue navigator. They will be displayed as a link at the top of the issue navigator. Once clicked, JIRA will render the search request view.



Here's what the bundled 'Printable' view looks like:

Search request views can be used to render the search results in any possible representation. Some popuplar choices are for example RSS or XML.

The Search Request View Plugin Module

A search request view plugin consists of 3 components:

- The plugin module definition in `atlassian-plugin.xml`
- The view implementation class defining any logic needed to render a view
- Any number of view templates defining how to render the view.

Let's attempt a sample implementation, to render the results of a search request in this format:

```
<issues filtername="My filter">
   <issue>
      <key>HSP-1</key>
      <summary>Sample issue</summary>
   </issue>
   <issue>
      <key>MKY-1<key>
      <summary>Another sample issue</summary>
   </issue>
</issues>
```

Plugin Module Definition

First a definition of the plugin is needed in your plugin's atlassian-plugin.xml:

```
...
    <search-request-view key="simple-searchrequest-xml" name="Simple XML"
                        class="com.atlassian.jira.sample.searchrequest.SimpleSearchRequestXmlView"
state='enabled'
                        fileExtension="xml" contentType="text/xml">
        <resource type="velocity" name="header" location="templates/searchrequest-xml-header.vm"/>
        <resource type="velocity" name="singleissue" location=
"templates/searchrequest-xml-singleissue.vm"/>
        <resource type="velocity" name="footer" location="templates/searchrequest-xml-footer.vm"/>
        <order>100</order>
    </search-request-view>
...
```

The search-request-view module is fairly straight-forward. It defines a unique **key** for the module, a **name** which will be used for the link in the issue navigator and the view implementation **class**. Further it also defines what **contentType** to return when displaying this view and a number of view resources. Finally the **order** can be used to control in which order the links appear in the IssueNavigator.

Search Request View Implementation Class

The view implementation class below extends a convenience class available in JIRA to make writing search request views easier, namely AbstractSearchRequestView. All that's left to do then is to implement the `writeSearchResults()` method.

If a search request view needs more control (e.g. control the HTTP headers used in the returned view), then an implementation class can implement the SearchRequestView interface directly.

SimpleSearchRequestXmlView.java

```
...

/**
 * Sample implementation of a simple XML search request view.
```

```
 * <p/>
 * Note that this class extends {@link
com.atlassian.jira.plugin.searchrequestview.AbstractSearchRequestView}. This
 * isn't necessary but makes things a lot simpler.  It is also possible to implement the {@link
 * com.atlassian.jira.plugin.searchrequestview.SearchRequestView} interface directly.
 */
public class SimpleSearchRequestXmlView extends AbstractSearchRequestView
{
    private final JiraAuthenticationContext authenticationContext;
    private final SearchProviderFactory searchProviderFactory;
    private final IssueFactory issueFactory;
    private final SearchProvider searchProvider;

    public SimpleSearchRequestXmlView(JiraAuthenticationContext authenticationContext,
SearchProviderFactory searchProviderFactory,
            IssueFactory issueFactory, SearchProvider searchProvider)
    {
        this.authenticationContext = authenticationContext;
        this.searchProviderFactory = searchProviderFactory;
        this.issueFactory = issueFactory;
        this.searchProvider = searchProvider;
    }

    public void writeSearchResults(final SearchRequest searchRequest, final SearchRequestParams
searchRequestParams, final Writer writer)
    {
        final Map defaultParams = JiraVelocityUtils.getDefaultVelocityParams(authenticationContext);

        //Need to put the filtername into the velocity context.  This may be null if this is an
anoymous filter.
final Map headerParams = new HashMap(defaultParams);
        headerParams.put("filtername", searchRequest.getName());
        try
        {
            //First we need to write the header
writer.write(descriptor.getHtml("header", headerParams));

            //now lets write the search results.  This basically iterates over each issue in the
search results and writes
//it to the writer using the format defined by this plugin.  To ensure that this doesn't result in
huge
//memory consumption only one issue should be loaded into memory at a time.  This can be guaranteed by
using a
//Hitcollector.
final Searcher searcher = searchProviderFactory.getSearcher(SearchProviderFactory.ISSUE_INDEX);
            final Map issueParams = new HashMap(defaultParams);
            //This hit collector is responsible for writing out each issue as it is encountered in the
search results.
//It will be called for each search result by the underlying Lucene search code.
final DocumentHitCollector hitCollector = new IssueWriterHitCollector(searcher, writer, issueFactory)
            {
                protected void writeIssue(Issue issue, Writer writer) throws IOException
                {
                    //put the current issue into the velocity context and render the single issue view
issueParams.put("issue", issue);
                    writer.write(descriptor.getHtml("singleissue", issueParams));
                }
            };
            //now run the search that's defined in the issue navigator and pass in the hitcollector
from above which will
//write out each issue in the format specified in this plugin.
searchProvider.searchAndSort(searchRequest, authenticationContext.getUser(), hitCollector,
searchRequestParams.getPagerFilter());

            //finally lets write the footer.
writer.write(descriptor.getHtml("footer", Collections.emptyMap()));
        }
        catch (IOException e)
        {
            throw new RuntimeException(e);
```

```
        }
        catch (SearchException e)
        {
            throw new RuntimeException(e);
        }
```

```
    }
}
```

View templates

Finally, a number of view templates are needed to display our new view.

First, the header needs to display the **filtername** as an attribute of the `<issues>` tag or simply 'Anonymous' if the view isn't displaying a saved filter:

searchrequest-xml-header.vm

```
#set($displayName = 'Anonymous')
#if($filtername)
    #set($displayName = $textutils.htmlEncode($filtername))
#end
<issues filtername="$displayName">
```

Then to render each issue, we need to construct an `<issue>` tag for each individual issue:

searchrequest-xml-singleissue.vm

```
<issue>
        <key>$!issue.key</key>
        <summary>$!issue.summary</summary>
    </issue>
```

Finally the footer should close the `<issues>` tag:

searchrequest-xml-footer.vm

```
</issues>
```

Example

The full source for the sample plugin above can be downloaded here. If you wish to just try the plugin out feel free to download the plugin jar.

Here's what the sample plugin looks like in action:

And this is a sample search result rendered by the plugin:

```
<issues filtername="All issues">
    <issue>
        <key>HSP-2</key>
        <summary>New issue</summary>
    </issue>
    <issue>
        <key>HSP-1</key>
        <summary>dude</summary>
    </issue>
</issues>
```

Servlet Plugin Module

Servlet plugin modules are implemented in the 3.5 Release. They are not available in previous releases.

Servlet plugin modules enable you to deploy Java servlets as a part of your plugins.

- For more information about plugins in general, read JIRA Plugin Guide.
- To learn how to install and configure plugins (including macros), read Installing and Configuring Plugins.

The Servlet Plugin Module

Each servlet is deployed as a plugin module of type "servlet". Here is an example `atlassian-plugin.xml` file containing a single servlet:

```
<atlassian-plugin name='Hello World Servlet' key='jira.plugin.helloworld'>
    <plugin-info>
        <description>A basic Servlet module test - says "Hello World!"</description>
        <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
        <version>1.0</version>
    </plugin-info>

    <servlet name='Hello World Servlet' key='helloWorld'
        class='com.atlassian.jira.plugin.helloworld.HelloWorldServlet'>
        <description>Says Hello World, Australia or your name.</description>
        <url-pattern>/helloworld</url-pattern>
        <init-param>
            <param-name>defaultName</param-name>
            <param-value>Australia</param-value>
        </init-param>
    </servlet>
</atlassian-plugin>
```

- the **class** attribute of `servlet` is an subclass of `javax.servlet.http.HttpServlet`.
- the **url-pattern** elements (one or more) define the locations this servlet will be accessed.
- the **init-param** elements allow you to define initial parameters for your servlet, using the same method as you would normally in `web.xml`.

Accessing Your Servlet

You servlet will be accessed within the JIRA web application via each `url-pattern` you specify, beneath the `/plugins/servlet` parent path.

For example, if you specify a `url-pattern` of `/helloworld` as above, and your JIRA application was deployed at http://yourserver/jira - then your servlet would be accessed at http://yourserver/jira/plugins/servlet/helloworld .

Notes

Some information to be aware of when developing or configuring a servlet plugin module:

- Your servlet's `init()` method will not be called on web application startup, as for a normal servlet. Instead, this method will be called the first time your servlet is accessed after each time it is enabled. This means that if you disable a plugin containing a servlet, or a single servlet module, and re-enable it again `init()` will be called again.
- Because all servlet modules are deployed beneath a common `/plugins/servlet` root, be careful choosing each `url-pattern` under which your servlet is deployed. It is recommended to use a value that will always be unique to the world!

User Format Plugin Guide

User Format plugin modules are implemented in the JIRA 3.13 release. They are not available in previous releases.

User Format plugin modules are used to display user details in JIRA. JIRA ships with a number of default user format implementations that are used to render the full names for users system wide. You can use User Format plugin modules to implement custom behaviours for these user details. Here are some examples:

- Display a profile picture next to the user.
- Link to an external profile page.
- Display special avatars for users in certain groups.

For more information about plugins in general, read JIRA Plugin Guide. To learn how to install and configure plugins (including macros), read Installing and Configuring Plugins.

The User Format Plugin Module

Here is an example `atlassian-plugin.xml` file containing a single user format plugin module:

```
<atlassian-plugin name='User Format' key='jira.user.format' i18n-name-key="user.format.plugin.name">
    <plugin-info>
        <description key="user.format.plugin.desc">This plugin renders a user in JIRA.</description>
        <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
        <application-version min="4.0" max="4.0"/>
        <version>1.0</version>
    </plugin-info>

    <user-format key="profile-link-user-format" i18n-name-key="user.format.plugin.profile.link.name"
name="Profile Link User Format"
          class="com.atlassian.jira.plugin.profile.ProfileLinkUserFormat" system="true">
        <description key="user.format.plugin.profile.link.desc">Simple link to a user's profile page
displaying the user's full name.</description>
        <type i18n-name-key="user.format.type.profile.link">profileLink</type>
        <resource type="velocity" name="view" location="templates/plugins/userformat/profileLink.vm"/>
    </user-format>
</atlassian-plugin>
```

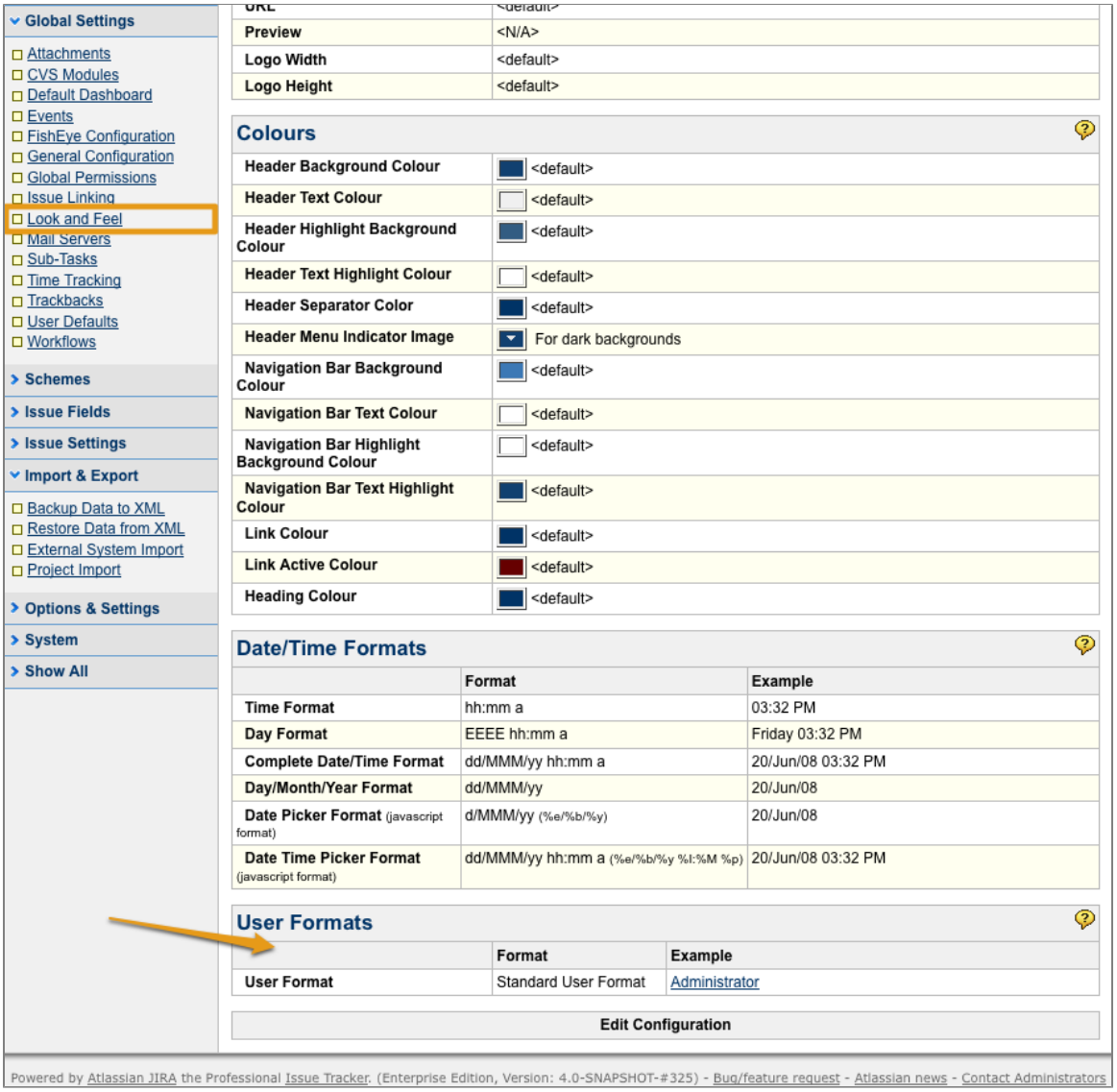
* the **class** attribute of `user-format` needs to implement `com.atlassian.jira.plugin.profile.UserFormat`.

- the **resource** elements (one or more) can be used to implement complicated rendering using velocity.
- The **type** attribute defines where the user format will be used. The system types that are pre-defined include:

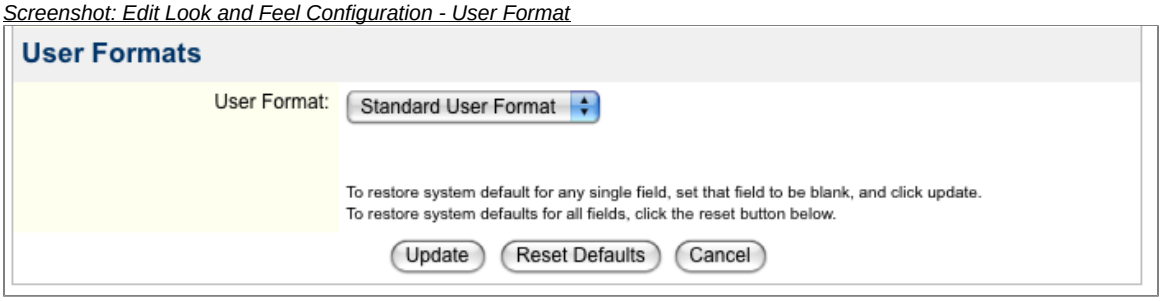
Type	Description
profileLink	Simple link to a user's profile page displaying the user's full name.
fullName	Safely displays the user's full name.
profileLinkSearcher	Simple link to a user's profile page displaying the user's full name from the issue navigator.
profileLinkExternal	Simple link to a user's profile used in emails, word documents, excel downloads etc.
profileLinkActionHeader	Simple link to a user's profile in issue action headers such as comments.
fullProfile	Full user description including user operation links and report links.

Choosing your User Format

Once you've added your own user format, JIRA's 'Look and Feel' administration allows you to select the User Format that will be used across JIRA:
Screenshot: View Look and Feel Configuration



Screenshot: Edit Look and Feel Configuration - User Format



Sample Implementation

The following example demonstrates how to implement a user format that prints a user's full name with a link to the user's profile page in JIRA.

1. First, you will need an implementation of the `UserFormat` interface:

ProfileLinkUserFormat.java

```
package com.atlassian.jira.plugin.profile;

import com.atlassian.core.util.map.EasyMap;
import com.atlassian.jira.user.util.UserUtil;
import com.opensymphony.user.User;

import java.util.Map;

/**
 * Very simple implementation that only renders the users full name with a link to the user's profile
page. If the
 * username is null, it will display 'Anonymous'.  If no user matching the username can be found, ony
the username
 * will be printed.
 *
 * @since v4.0
 */
public class ProfileLinkUserFormat implements UserFormat
{
    private UserFormatModuleDescriptor moduleDescriptor;
    private UserUtil userUtil;
    private final UserFormatManager userFormatManager;

    public ProfileLinkUserFormat(UserFormatModuleDescriptor moduleDescriptor, UserUtil userUtil,
UserFormatManager userFormatManager)
    {
        this.moduleDescriptor = moduleDescriptor;
        this.userUtil = userUtil;
        this.userFormatManager = userFormatManager;
    }

    public String format(String username, String id)
    {
        final Map params = getInitialParams(username, id);
        return moduleDescriptor.getHtml(VIEW_TEMPLATE, params);
    }

    public String format(String username, String id, Map params)
    {
        final Map velocityParams = getInitialParams(username, id);
        velocityParams.putAll(params);

        return moduleDescriptor.getHtml(VIEW_TEMPLATE, velocityParams);
    }

    private Map getInitialParams(final String username, final String id)
    {
        final User user = userUtil.getUser(username);
        final String fullName = userFormatManager.formatUser(username, FullNameUserFormat.TYPE, id);
        return EasyMap.build("username", username, "user", user, "fullname", fullName, "id", id);
    }
}
```

2. You will then need to implement the view velocity template that used to display the user:

profileLink.vm

```
#if ($username)
    #set ($quote = '"')
    #if($user)
        #set($author = "<a id=${quote}$!{id}_${textutils.htmlEncode($username)}${quote}
href=${quote}${baseurl}/secure/ViewProfile.jspa?name=${velocityhelper.urlencode($username)}${quote}>$text
)
    #else
        #set($author = $textutils.htmlEncode($username))
    #end
#else
    #set($author = $i18n.getText('common.words.anonymous'))
#end
${author}
```

3. Finally, you can use this plugin to print the user's details, as shown below:

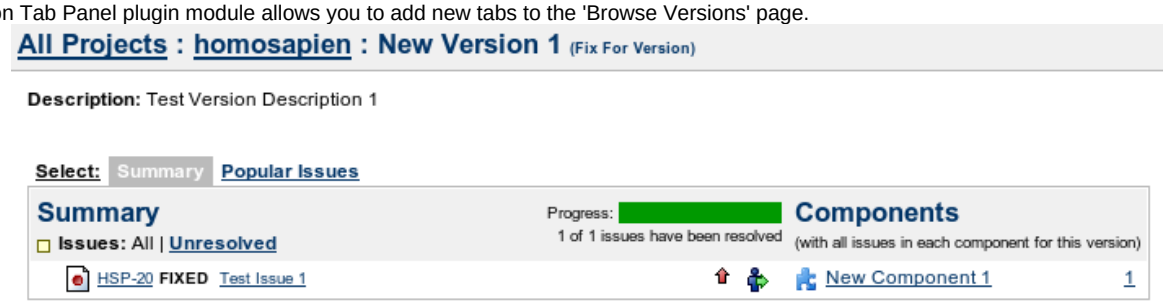
```
$userformat.formatUser($worklog.author, 'profileLink', "worklog_${worklog.id}_header")
```

In this case, `profileLink` the **type** and `worklog_${worklog.id}_header` is the **id** that is passed to the UserFormat.format method for rendering.

Version Tab Panel Plugin Module

The Version Tab Panel plugin module is available in JIRA version 3.10 and later.

The Version Tab Panel plugin module allows you to add new tabs to the 'Browse Versions' page.



These tab panels ('Summary' and 'Popular Issues') are implemented as plugins. New version tab panels can be implemented to display version-specific info.

Here is an example panel module descriptor:

```
<!--
The class defined should implement
com.atlassian.jira.plugin.versionpanel.VersionTabPanel
and it may be useful to use the functionality provided by
com.atlassian.jira.plugin.versionpanel.impl.GenericTabPanel.
-->
<version-tabpanel key="version-openissues-panel" i18n-name-key="versionpanels.openissues.name"
                  name="Open Issues Panel" class=
"com.atlassian.jira.plugin.versionpanel.impl.GenericTabPanel">
        <description key="versionpanels.openissues.description">Show the open issues for this version.
</description>
        <label key="common.concepts.openissues"/>
        <!-- this is a number defining the order of all panels.
             The system panels are 10, 20, 30 and 40. -->
        <order>10</order>
        <!-- this template produces the HTML for the panel -->
        <resource type="velocity" name="view"
                  location="templates/plugins/jira/projectentitypanels/openissues-version-panel.vm"/>
        <!-- this properties files contains i18n keys for the panel -->
        <resource type="i18n" name="i18n" location=
"com.atlassian.jira.plugins.versionpanels.openissues"/>
    </version-tabpanel>
```

Web Resources

JIRA plugins may define downloadable resources. If your plugin requires JIRA to serve additional static Javascript or CSS files, you will need to use downloadable web resources to make them available. Web resources differ from Downloadable Plugin Resources in that web resources are added at the top of the page in the header.

This is only available as of JIRA 3.7 and above.

Defining a Single Web Resource

Downloadable resources are configured to map a `name` of some downloadable file to its `location` within the plugin jar-file.

```
<atlassian-plugin name='IM Presence Macros' key='jira.extra.impresence'>
    <plugin-info>
        <description>Macros to show online status for popular Instant Messaging services.
</description>
        <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
        <version>0.1</version>
    </plugin-info>

    <web-resource key="scriptaculous" name="Scriptaculous" >
        <resource type="download" name="scriptaculous.js" location=
"/includes/js/effects/scriptaculous.js" />
        <resource type="download" name="effects.js" location="/includes/js/effects/effects.js" />
    </web-resource>

</atlassian-plugin>
```

- Resources must be contained within a <webresource> tag. Loading other resources may be implemented in the future.
- The `key` of the webresource is how it will be referenced from within the application
- Each resource must be of `type="download"`
- The `name` of the resource will be suffixed to the URL
- The `location` of the resource is where it appears within the plugin itself

Referring to Web Resources

In your plugin, you need to refer to a WebResourceManager, and call the `requireResource()` method. Getting a reference to a

WebResourceManager is application specific, but in JIRA it would be:

```
WebResourceManager webResourceManager = new WebResourceManagerImpl();
webResourceManager.requireResource("jira.extra.impresence:scriptaculous"); //should be the full module
key for the <webreference> module.
```

The URL for a downloadable resource is application specific. For JIRA it would be follows:

```
{server root}/s/{build num}/{plugin version}/{system counter}/c/download/resources/{plugin key}:{module
key}/{resource name}
```

For the above example:

```
<script type="text/javascript"
  src=
"http://jira.example.com/s/170/1.0/1/_/download/resources/jira.extra.impresence:scriptaculous/scriptaculo
></script>
<script type="text/javascript"
  src=
"http://jira.example.com/s/170/1.0/1/_/download/resources/jira.extra.impresence:scriptaculous/effects.js"
></script>
```

will be inserted in the header of the page.

Web UI Plugin Module

Web UI Plugin module

Web UI plugin modules are available in Jira 3.7 and later.

Web UI plugin modules allow you to insert links, tabs and sections of links into the Jira web interface

Sections and Items

Web UI plugins can consist of two kinds of plugin modules:

- web-item modules define links that will be displayed in the UI at a particular location
- web-section modules define a collection of links that will be displayed together

Web items or web sections (referred to collectively as 'web fragments') may be displayed in a number of different ways, depending on the location of the fragment.

Locations

In a number of places in the Jira UI, there are lists of links representing operations relevant to the content being viewed. These are the *locations* and *sections* that you can customise:

Location key	Sections	Dynamic Sections?	Description
system.admin	project usersgroups globalsettings schemes issuefields issuesettings exportimport options system		The administrative menu links on the left-hand side of the Administration page

system.preset.filters		✗	The filter links on the right hand side of Browse Project page and on the project portlets of the dashboard. Note that the filter links are defined only once and is reused in both places. Links are displayed in two columns where the right column contains the every second link, and the rest in the left column
system.top.navigation.bar		✗	System links on the top navigation bar The web-item can optionally specify up to 2 URL prefix that will be used to determine whether the link is selected or not. This is done by adding a `param` element with names: `selected` and `selected2`
system.user.navigation.bar	links views	✗	Links on the top-right user navigation bar. web-item's must be added to either the `links` or `views` section. • `links` section allows its links to be plain or popup links. To make a link to be a popup link, set a `param` element with name `isPopupLink` as true. Additional parameters can be set to dynamically set the height (`windowHeight`), width (`windowWidth`) and whether to have scrollbars (`scrollbars`) for the popup • `views` section displays all its links as icons and require the `icon` element
system.browse.project.operations		✗	Project operation links on the 'Browse Project' Page
system.view.project.operations		✗	Project operation links on the 'View Project' Page

- Locations marked as being 'Dynamic Sections' allows new sections to be added. Whereas unmarked locations will *not* display new sections added to that location and require the items to be added to the existing sections. If there are no sections, the items should be added to the location key itself.

Web Item Definition

Here's a sample atlassian-plugin.xml fragment for a web item:

```
<web-item key="popup_recent_history" name="Recent History Popup Link" section=
"system.user.navigation.bar/popups" weight="10">
    <label key="bodytop.history" />
    <tooltip key="tooltip.history" />
    <link linkId="user_history">/secure/popups/recenthistory.jsp</link>
    <param name="windowHeight">$historyWindowHeight</param>
    <context-provider class="com.atlassian.jira.plugin.web.contextproviders.HeightContextProvider" />
    <condition class="com.atlassian.jira.plugin.web.conditions.UserHasIssueHistoryCondition" />
</web-item>
```

The web-item has the following attributes:

Attribute	Description	Required?
key	Plugin module key, which must be unique within the plugin. It is also used in the UI for determining which module is currently being displayed.	✓
name	Displayable name for the module, only used in the plugin administrative UI	✓
section	Location into which this web item should be placed. For non-sectioned locations, this is just the location key. For sectioned locations it is the location key, followed by a slash ('/'), and the name of the web section in which it should appear	✓
weight	Determines the order in which web items appear. Items are displayed top to bottom or left to right in order of ascending weight. (i.e. the 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom). Jira's system sections and links weight start from 100 and 10 respectively. The weight is incremented by 10 for each in sequence so that there is ample space to insert your own sections and links	✓

The web-item has the following elements

Element	Description	Required?

label	Is the i18n key that will be used to look up the textual representation of the link.	✓
tooltip	Is the i18n key that will be used to look up the textual mouse-over text of the link.	✗
link	Defines where the web item should link to. The contents of the link element will be rendered using Velocity, allowing you to put dynamic content in links. For more complex examples of links, see below.	✓
icon	Defines an icon to display with or as the link.	✗
param	Defines a key/value pair available from the web-item. This is handy if you want to use additional custom values from the UI.	✗
context-provider	Allows dynamic addition to the velocity context available for various web-item elements (in XML descriptors only). Currently only one context-provider can be specified per web item and section.	✗
condition	Defines a condition that must be satisfied for the web item to be displayed. If you want to 'invert' a condition, add an attribute 'invert="true"' to it. The web item will then be displayed if the condition returns false (not true).	✗
conditions	Defines the logical operator type to evaluate its condition elements. By default 'AND' will be used.	✗

Label elements

Label elements may contain optional parameters, as shown below:

```
<label key="common.concepts.create.new.issue">
    <param name="param0">$helper.project.name</param>
</label>
```

- The parameters allow you to insert values into the label using Java's MessageFormat syntax.
- Parameter names must start with `param` and will be mapped in *alphabetical order* to the substitutions in the format string. ie. param0 is {0}, param1 is {1}, param2 is {2} etc
- Parameter values are rendered using Velocity, allowing you to include dynamic content

Tooltip elements

Tooltip elements have the same attributes and parameters as the label elements.

Link elements

Link elements may contain additional information:

```
<link linkId="create_link" accessKey=
"$authcontext.i18nBean.getText('alt.text.createnewissue.accessKey')">/secure/CreateIssue!default.jspa
</link>
```

- The **linkId** is **optional**, and provides an XML id for the link being generated.
- The **accessKey** is **optional** and provides an access key for the link being generated. The contents of this element will be rendered using Velocity, allowing you to generate an access key dynamically.

Icon elements

Icon elements have a `height` and `width` attribute. The location of the icon is specified within a link element:

```
<icon height="16" width="16">
    <link>/images/icons/print.gif</link>
</icon>
```

Param elements

Param elements represent a Map of key/value pairs, where each entry corresponds to the param elements attribute: `name` and `value` respectively.

```
<param name="key" value="value" />
```

its value can be retrieved from within the velocity view by (where $item is an `WebItemModuleDescriptor`):

```
$item.webParams.get("key") <!-- retrieve the value -->
$item.webParams.getRenderedParam("key", $user, $helper) <!-- retrieve the velocity rendered value -->
```

If the `value` attribute is not specified the value will be set to the body of the element. ie. the following two param elements are equivalent

```
<param name="isPopupLink" value="true" />
<param name="isPopupLink">true</param>
```

Context-provider element

Adds to the velocity context available to the `web-section` and `web-item` modules. This means you can add what you need to the context to build more flexible section and item elements. Currently only one context-provider can be specified per module, additional context-providers are ignored.

`context-provider` element must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement `com.atlassian.plugin.web.ContextProvider`
- will be auto-wired by Pico before any additions to the velocity context

For example, the following context-provider will add `historyWindowHeight` and `filtersWindowHeight` to the context.

The following example: `HeightContextProvider` extends `AbstractJiraContextProvider`, which implements `ContextProvider`. The `AbstractJiraContextProvider` conveniently extracts the `User` and `JiraHelper` from the context map you would otherwise have to do manually.

```
public class HeightContextProvider extends AbstractJiraContextProvider
{
    private final ApplicationProperties applicationProperties;

    public HeightContextProvider(ApplicationProperties applicationProperties)
    {
        this.applicationProperties = applicationProperties;
    }

    public Map getContextMap(User user, JiraHelper jiraHelper)
    {
        int historyIssues = 0;
        if (jiraHelper != null && jiraHelper.getRequest() != null)
        {
            UserHistory history = (UserHistory)
jiraHelper.getRequest().getSession().getAttribute(SessionKeys.USER_ISSUE_HISTORY);
            if (history != null)
            {
                historyIssues = history.getIssues().size();
            }
        }
        int logoHeight =
TextUtils.parseInt(applicationProperties.getDefaultBackedString(APKeys.JIRA_LF_LOGO_HEIGHT));
        String historyHeight = String.valueOf(80 + logoHeight + (25 * historyIssues));
        String filterHeight = String.valueOf(205 + logoHeight);
        return EasyMap.build("historyWindowHeight", historyHeight,
                             "filtersWindowHeight", filterHeight);
    }
}
```

The above `HeightContextProvider` can be used by nesting the following element in a web-item module.

```
<context-provider class="com.atlassian.jira.plugin.web.contextproviders.HeightContextProvider" />
```

The newly added context entries historyWindowHeight and filtersWindowHeight can be used in the XML module descriptors just like normal velocity context variables by prefixing it with the dollar symbol ($)

```
<!-- pass the value of historyWindowHeight as a parameter called windowHeight (see param element above
for its usage) -->
<param name="windowHeight">$historyWindowHeight</param>

<!-- set the link's label to print the value of filtersWindowHeight -->
<label>filter window height is: $filtersWindowHeight</label>
```

Condition and Conditions elements

Conditions can be added to the `web-section` and `web-item` modules to display them only when **all** its conditions are true.

Condition elements must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement `com.atlassian.plugin.web.Condition`
- will be auto-wired by Pico before any condition checks are performed

Condition elements can take optional parameters. These parameters will be passed in to the Condition's `init()` method as a Map of String key/value pairs after autowiring, but before any condition checks are performed. For example:

```
<condition class="com.atlassian.jira.plugin.web.conditions.JiraGlobalPermissionCondition">
    <param name="permission">admin</param>
</condition>
```

To invert a condition, add the attribute 'invert="true"' to the condition element. This is useful where you want to show the section if a certain condition is *not* satisfied.
Conditions elements are composed of a collection of condition/conditions elements and a type attribute. The type attribute defines what logical operator is used to evaluate its collection of condition elements. The type can be one of **AND** or **OR**.
For example: The following conditions is true if the current user is a system admin OR a project admin

```
<conditions type="OR">
    <condition class="com.atlassian.jira.plugin.web.conditions.JiraGlobalPermissionCondition">
        <param name="permission">admin</param>
    </condition>
    <condition class="com.atlassian.jira.plugin.web.conditions.UserHasProjectsCondition">
        <param name="permission">project</param>
    </condition>
</conditions>
```

Web Section Definition

Here's a sample atlassian-plugin.xml for a web section:

```
<web-section key="usersgroups" name="Users and Groups Section" location="system.admin" weight="110">
    <label key="admin.menu.usersandgroups.users.and.groups" />
    <condition class="com.atlassian.jira.plugin.web.conditions.UserIsAdminCondition" />
</web-section>
```

- **key**, **name**, **weight**, **label**, **tooltip**, **condition** and **conditions** are all the same as `web-item`
- **location** is **required**, and serves the same purpose as **section** in `web-item`, except that a section **can not** be contained within another section.
- Web sections do not have links or icons.

Velocity Context

The following table lists out the velocity context available for use in the XML descriptor and velocity views.

Name	Type	Description
user	User	Currently logged in user
helper	JiraHelper	Convenient class which holds information such as the current request and selected project
xmlutils	XMLUtils	Utilities for basic XML reading
textutils	TextUtils	Utilities for common String manipulations
urlcodec	JiraUrlCodec	Utility for encoding a string
outlookdate	OutlookDate	Class to give a nice String representation of a date
authcontext	JiraAuthenticationContext	User locale dependant utility class. Can get the current user, locale, I18nBean (for internationalisation) and OutlookDate
dateutils	DateUtils	Utilities for displaying date/time
externalLinkUtil	ExternalLinkUtil	A simple utility class that lets you resolve external links that may need to change, because of partner sites and such
requestContext	VelocityRequestContext	A context that allows for different implementations depending on whether it is running in the scope of a web request, or via email.
req	HttpServletRequest	current request
baseurl	String	The base URL for this instance (velocityRequestContext.getBaseUrl())

Examples

Heres a simple example that uses both the web UI module and the webwork plugin module.
A webwork plugin module defines a URL-addressable 'action', allowing JIRA's user-visible functionality to be extended or partially overridden. In this example, the action simply prints "Hello World" or greets a given name. However the action can only be executed by entering a specific URL that is not linked from Jira. This is where the web UI plugin comes in play, by adding the specific URL as a link from the Jira web interface.

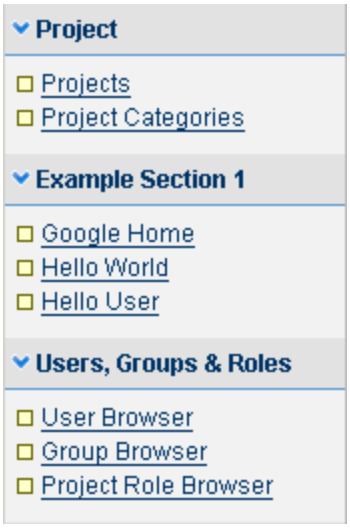
The following plugin modules will:

- Register a new webwork action that prints "Hello World" or Hello to a specific name
- Adds a new section to the administration menu called Example 1 Section
- Adds 3 links under the new section:
    - Google Home - static link to http://google.com
    - Hello World - static link to the HelloWorld action with no name argument
    - Hello User - dynamic link to the HelloWorld action with current user's login name

```
<webwork1 key="HelloWorld" name="Hello World Example Action">
    <description key="action.hello.world.desc">Webwork plugin example that prints hello world. Can
also specify a name to say hello to.</description>
    <actions>
        <action name="HelloWorldAction" alias="Hello">
            <view name="success">/templates/example/helloworld.vm</view>
        </action>
    </actions>
</webwork1>

<web-section key="example1" name="Example 1 Section" location="system.admin" weight="105">
    <label key="section.example.one.label" />
</web-section>

    <web-item key="google_home" name="Google Home" section="system.admin/example1" weight="10">
        <description key="item.google.home.desc">Simple link to google.com.</description>
        <label key="item.google.home.label" />
        <link linkId="google_home">http://google.com</link>
    </web-item>

    <web-item key="hello_world" name="Webwork Hello World Example" section="system.admin/example1"
weight="20">
        <description key="item.hello.world.desc">Link to the Hello World action. No name parameter
specified.</description>
        <label key="item.hello.world.label" />
        <link linkId="hello_world">/secure/Hello.jspa</link>
    </web-item>

    <web-item key="hello_user" name="Webwork Hello User Example" section="system.admin/example1"
weight="30">
        <description key="item.hello.user.desc">Link to the Hello World action with name set to the
current user's full name.</description>
        <label key="item.hello.user.label" />
        <link linkId="hello_user">/secure/Hello.jspa?name=${user.fullName}</link>
        <condition class="com.atlassian.jira.plugin.web.conditions.UserLoggedInCondition" />
    </web-item>
```

Here is the screenshot of the new administration menu:

The new section "Example 1 Section" appears in between the "Project" and "Users, Groups & Roles" as its weight is in between the two.



Get the full sample plugin from the Jira Plugin Development Kit. The sample shows how to combine a simple webwork module with a web UI module to provide you with an interface to your plugin from Jira. You can find a real life application of this in the JIRA Subversion plugin (for Jira 3.7 compatible versions only).

How to create a custom preset filter

How to create a custom preset filter

This 'how to' guide will demonstrate how to create your very own custom preset filter using the Web UI plugin module.

Firstly a preset filter is a link to the JIRA issue navigator with pre-defined filter attributes. These links appear on Browse Project page and on the project portlets.

Creating a new preset-filter is easy and simple to add to your JIRA plugins. All you need to do is define a new web-item module in your `atlassian-plugin.xml` with the section attribute set to '`system.preset.filters`'. JIRA will automatically add a new filter link to both the `browse project page` and the `project portlets`. The following `web-item` template is the minimal required to create a preset filter. (Please refer to the Web UI Plugin Module for further details on defining a web-item such as restricting when the filter is available)

```
<web-item key="[FILTER_KEY]" name="[FILTER_NAME]" section="system.preset.filters">
    <label key="[FILTER_LABEL]" />
    <link>[FILTER_URL]</link>
</web-item>
```

The above attribute values (enclosed in '[' and ']') are explained in Web UI Plugin Module. The main challenge of making a preset filter is generating the [FILTER_URL]. For example the following relative URL repesents the standard JIRA preset filter `Outstanding`

```
/secure/IssueNavigator.jspa?reset=true
&mode=hide&pid=10240&resolution=-1&sorter/field=updated&sorter/order=DESC
```

This may seem cryptic or cumbersome to create/modify manually and require you to know exactly what the field and its values are. The best approach to generating these links is to use the Issue Navigator to filter the issues as normal. Once you have finished filtering the issues, you simply copy the link provided by the permlink (by right clicking on it and copying the links target, this may depend on the browser you are using).

Step by Step Example: 'All closed issues in current project'

This example will demonstrate how to create a new preset filter called `Closed` which finds all issues in Closed status for a given project.

1. Start off with the minimal web-item module defined above

```
<web-item key="[FILTER_KEY]" name="[FILTER_NAME]" section="system.preset.filters">
    <label key="[FILTER_LABEL]" />
    <link>[FILTER_URL]</link>
</web-item>
```

2. Change the attribute values to desired values (excluding the [FILTER_URL] for now)

```
<web-item key="filter_closed" name="Closed Issues" section="system.preset.filters">
    <label key="Closed" />
    <link>[FILTER_URL]</link>
</web-item>
```

3. Now to generate the filter url
    a. Goto the Issue Navigator and set the following search fields as follows:
        - Project: select any one single project (this is to get the fields id)
        - Status: Closed

        then click on `View` (or `View & Hide`) button
    b. Right click on the permlink and copy its target link (You can also click on the `permlink` and copy from the address bar) to get the absolute URL. You should get something like the following:

```
http://company.com:8080/contextpath/secure/IssueNavigator.jspa?reset=true
&pid=10000&status=6&sorter/field=issuekey&sorter/order=DESC
```

    c. We will need to escape reserved characters such as the ampersand '&' with '&'

```
http://company.com:8080/contextpath/secure/IssueNavigator.jspa?reset=true
&amp;pid=10000&amp;status=6&amp;sorter/field=issuekey&amp;sorter/order=DESC
```

    d. (Optional) To make your preset-filter work on any JIRA instance (different host/context path), it is better to make the absolute URL from the previous step into a relative one. This step involves simply removing everything before `/secure/IssueNavigator.jspa`.
    The relative URL would be:

```
/secure/IssueNavigator.jspa?reset=true
&amp;pid=10000&amp;status=6&amp;sorter/field=issuekey&amp;sorter/order=DESC
```

    e. This URL will give us the closed issues for the project with the id `10000`. To make this dynamic we replace `10000` with `$helper.project.id` which will get the current project's id (please refer to Web UI Plugin Module for further details). So our final [FILTER_URL] is given by:

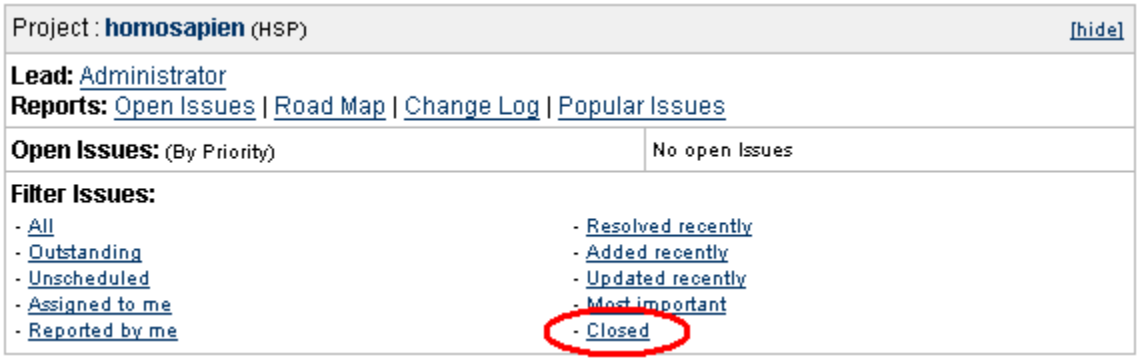
```
/secure/IssueNavigator.jspa?reset=true
&amp;pid=$helper.project.id&amp;status=6&amp;sorter/field=issuekey&amp;sorter/order=DESC
```

4. The final `web-item` would look something like this:

```
<web-item key="filter_closed" name="Closed Issues" section="system.preset.filters">
    <label key="Closed" />
    <link>
/secure/IssueNavigator.jspa?reset=true&amp;pid=$helper.project.id&amp;status=6&amp;sorter/field=is
</link>
</web-item>
```

There are more attributes and elements that can be defined for the web-item module to provide additional features such as internationalisation and ordering of the filters. Please refer to the Web UI plugin module for details

You can now just add that to your `atlassian-plugin.xml` and it will appear on both the Browse Project page and on the project portlets as follows:



Compiling and installing the preset filter as a plugin in JIRA

The above steps showed how to make a single `plugin module`, this however is not the complete plugin. A JIRA plugin is a single JAR (Java ARchive) file with atleast a plugin descriptor (an XML file called `atlassian-plugin.xml`). This XML file includes one or more of the `plugin module`'s and contains additional information specifying the overall plugin's details such as its name, description, version, etc'.

1. The following sample can be used as a template to create the `atlassian-plugin.xml` file.

```
<!-- the plugin key must be unique, think of it as the 'package' of the plugin -->
<atlassian-plugin key="com.atlassian.plugin.sample" name="Sample Plugin">
    <!-- a short block describing the plugin itself -->
    <plugin-info>
        <description>This is a brief textual description of the plugin</description>
        <!-- the version of the plugin -->
        <version>1.0</version>
        <!-- the versions of the application this plugin is for -->
        <application-version min="3.7" max="3.7"/>
        <!-- details of the plugin vendor -->
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
    </plugin-info>

    <!-- ... Add your plugin modules here ... -->
</atlassian-plugin>
```

   Modifying the attributes of the template and adding the preset filter plugin module where marked, the final file will look like the attached: atlassian-plugin.xml.

2. Now the `atlassian-plugin.xml` needs to be packaged as a single JAR file. There are various ways of doing this, two simple ways are:
   - In a command line with `jar` installed, run the following command (with atlassian-plugin.xml in the current directory):

     ```
     jar -cvf plugin-name.jar atlassian-plugin.xml
     ```

     OR

   - Add the `atlassian-plugin.xml` file to a ZIP file (eg. using WinZip/WinRAR), then rename the ZIP file with a `.jar` extension (ie. rename `file.zip` to `file.jar`)
     The final jar file should look something like preset-filter-example-plugin.jar
3. To install the JAR file:

a. Shut down JIRA
b. copy the jar file into '.../atlassian-jira/WEB-INF/lib/'
c. Start up JIRA.
d. Go to 'Administration > Plugins' and confirm that the plugin is listed and enabled (look for the name specified in atlassian-plugin.xml - eg. Sample Plugin).
e. Enjoy!

Please note that the steps outlined above are only valid for packaging basic plugins such as the preset filters. For a more comprehensive plugin, you may need to include additional resources. Please refer to the JIRA Plugin Guide and Setting up your plugin project for more details.

You can get a full sample plugin from the Jira Plugin Development Kit which extends the `web-item` defined above by internationalising it and placing the 'Closed' link under the 'All' filter.

Webwork plugin module

A webwork plugin module defines a URL-addressible 'action', allowing JIRA's user-visible functionality to be extended or partially overridden.

Here is a sample webwork plugin module:

```
<webwork1 key="HelloWorld_key" name="My Hello World action" class="java.lang.Object">
    <actions>
        <action name="HelloWorldAction" alias="Hello">
            <view name="success">/templates/helloworld.vm</view>
        </action>
    </actions>
</webwork1>
```

Where:

- **HelloWorld_key** is any uniquely identifying key
- **My Hello World action** is a user-friendly module description (displays in the plugins admin section)
- Action name **HelloWorldAction** is a class name, relative to a package specified in `WEB-INF/classes/webwork.properties`:

  ```
  webwork.action.packages=webwork.action.standard, com.atlassian.jira.web.action
  ```

  For example, the class could be `com/atlassian/jira/web/action/HelloWorldAction.java`.

- Action alias **Hello** is the URL which this action can be invoked from, relative to `/secure`, eg. http://localhost:8080/jira/secure/Hello.jspa
- **/templates/helloworld.vm** is the template (here, velocity) which will render the HTML which the user sees, displaying information fetched by running methods on the action class (**HelloWorldAction**). It is **not** possible ot include a `*.jsp` file within the plugin jar, only Velocity files. If you wish to use a jsp, you must add it into the `webapps` directory or with the JIRA itself.

Webwork plugins effectively extend the actions defined in JIRA's main **WEB-INF/classes/actions.xml** file. You should look there for examples on what is possible.

Examples

By specifying as 'alias' the name of an existing action (in actions.xml), you can override default JIRA behaviour. For example, to override the Administrators.jspa action (the 'Contact Administrators' link at the bottom of every page):

```
<webwork1 key="ASFAdministrators" name="Lists project leads on administrators page" class=
"java.lang.Object">
        <actions>
            <action name="ASFAdministrators" alias="Administrators">
                <view name="success">/templates/asf_administrators.vm</view>
            </action>
        </actions>
    </webwork1>
```

See the full sample plugin

See also the [Copyright information for attachments] page for another example of overriding a default JIRA action.

Workflow Plugin Modules

The workflow plugin modules allow you to add new capabilities to JIRA's workflow engine.

JIRA uses OSWorkflow as it's workflow engine. The web-based workflow editor has a number of plugin modules which allow you to build workflows more easily.

The modules are:

- Conditions - check whether or not a given workflow transition can be executed by a given user
- Functions - perform actions after a workflow transition has been executed.
- Validators - check that the data given to a workflow transition is valid.

Here are some sample descriptors for these plugin modules:

```
<workflow-condition key="isuseringroup-condition" name="User Is In Group"
    class="com.atlassian.jira.plugin.workflow.WorkflowIsUserInGroupConditionFactoryImpl">
    <description>
    Condition to allow only users in a given group to execute a transition.
    </description>

    <condition-class>
    com.opensymphony.workflow.util.OSUserGroupCondition
    </condition-class>

    <resource type="velocity" name="view"
    location="templates/jira/.../isuseringroup-condition-view.vm"/>
    <resource type="velocity" name="input-parameters"
    location="templates/jira/.../isuseringroup-condition-input-params.vm"/>
</workflow-condition>

<workflow-function key="update-issue-field-function" name="Update Issue Field"
    class="com.atlassian.jira.plugin.workflow.UpdateIssueFieldFunctionPluginFactory">
    <description>Updates a simple issue field to a given value.</description>

    <function-class>
    com.atlassian.jira.workflow.function.issue.UpdateIssueFieldFunction
    </function-class>

    <orderable>true</orderable>
    <unique>false</unique>
    <deletable>true</deletable>

    <resource type="velocity" name="view"
    location="templates/jira/.../update-issue-field-function-view.vm"/>
    <resource type="velocity" name="input-parameters"
    location="templates/jira/.../update-issue-field-function-input-params.vm"/>
</workflow-function>


<workflow-validator key="permission-validator" name="Permission Validator"
    class="com.atlassian.jira.plugin.workflow.WorkflowPermissionValidatorPluginFactory">
    <description>Validates that the user has a permission.</description>

    <validator-class>
    com.atlassian.jira.workflow.validator.PermissionValidator
    </validator-class>

    <resource type="velocity" name="view"
    location="templates/jira/.../permission-validator-view.vm"/>
    <resource type="velocity" name="input-parameters"
    location="templates/jira/.../permission-validator-input-params.vm"/>
</workflow-validator>
```

For more details, see the How to create Custom Workflow Elements for JIRA 3 page.

Understanding how JIRA works

Understanding how JIRA works

Once you start building your own plugins, it is likely that you'll need to call on JIRA code to accomplish certain tasks; for example, to retreive a list of users, make workflow changes or add new data to issues. We've compiled some resources about how JIRA works under the hood to help you know how to access that functionality.

PicoContainer and Dependency Injection

JIRA uses PicoContainer to manage object creation throughout the system. It is important to understand how dependency injection works before trying to call JIRA functionality from your plugin. Here is a brief introduction to PicoContainer and JIRA.

Database

It may be useful to understand how JIRA interacts with the database.

> You should try not to manipulate JIRA's database directly, if you can avoid doing so.

Working with Issues and Fields

There are several docs that explain how to retreive and manipulate issues and fields.

- Creating and Editing an Issue
- How to retrieve an issue with an ID or an IssueKey
- How to search for Issues from within a Plugin
- Retrieving issue's links
- Working with Custom Fields
- JiraCreateSearch
- Issue Properties
- JiraIssueChangeHistory

API Docs

You can always looking things up in the API documentation..

Customising JIRA code

When customising JIRA, it is sometimes necessary to make code modifications. Most classes in JIRA conform to an interface (eg. the Webwork action classes, and *Manager classes), so it is possible to write your own implementation of JIRA interfaces and use yours instead of the default. This page describes the basics of how plug modified classes into JIRA with minimal pain.

Picocontainer

JIRA uses Picocontainer as a central object factory. Picocontainer is responsible for instantiating objects and resolving their constructor dependencies. This greatly simplifies code, in that any Picocontainer-instantiated object (eg. a Webwork action) can obtain an instance of another (eg. a Manager class) simply by requesting one in its constructor. PicoContainer will ensure each object required in the constructor is passed in (aka dependency injection). Eg. the ViewIssue action:

ViewIssue.java

```
public class ViewIssue extends AbstractViewIssue
{
    ....
    public ViewIssue(RepositoryManager repositoryManager, PermissionManager permissionManager,
TrackbackManager trackbackManager,
                     ThumbnailManager thumbnailManager, SubTaskManager subTaskManager,
IssueLinkManager issueLinkManager,
                     IssueLinkTypeManager issueLinkTypeManager, VoteManager voteManager,
WatcherManager watcherManager,
                     PluginManager pluginManager)
    {
        super(issueLinkManager, subTaskManager);
        this.trackbackManager = trackbackManager;
        this.thumbnailManager = thumbnailManager;
        this.issueLinkTypeManager = issueLinkTypeManager;
        this.pluginManager = pluginManager;
        this.pagerManager = new PagerManager(ActionContext.getSession());
        this.repositoryManager = repositoryManager;
        this.permissionManager = permissionManager;
        this.voteManager = voteManager;
        this.watcherManager = watcherManager;
    }
    ....
}
```

Non-managed classes

Classes not managed by Picocontainer (eg. workflow conditions / functions, Services and Listeners, or JSP scriptlets) can still get pico-instantiated objects statically using static methods of ManagerFactory or ComponentManager.getInstance(). For example:

```
ComponentManager.getInstance().getProjectManager();
ComponentManager.getInstance().getIssueFactory();
ManagerFactory.getCustomFieldManager();
//or
ManagerFactory.getApplicationProperties();
```

Using ComponentManager.getInstance() is prefered as ManagerFactory will become obsolete. However, if a method that you are after does not exist on the ComponentManager, use ManagerFactory.

Register new Picocontainer-managed classes

Picocontainer-managed classes need to be registered with Picocontainer. This happens automatically for Webwork actions, but other classes need to be registered manually. This is done in ComponentManager's registerComponents() method:

ComponentManager.java

```
    private void registerComponents()
    {
         ....
        internalContainer.registerComponentInstance(CoreFactory.getActionDispatcher());
        internalContainer.registerComponentInstance(CoreFactory.getGenericDelegator());
        internalContainer.registerComponentInstance(CoreFactory.getAssociationManager());
        internalContainer.registerComponentInstance(MailFactory.getServerManager());
        internalContainer.registerComponentInstance(UserManager.getInstance());
        internalContainer.registerComponentInstance(SearchProviderFactory.getDefaultProvider());
        .....
        internalContainer.registerComponentImplementation(IssueUpdater.class,
DefaultIssueUpdater.class);
        internalContainer.registerComponentImplementation(IssueEventDispatcher.class);
        internalContainer.registerComponentImplementation(ChangeLogUtils.class);
        internalContainer.registerComponentImplementation(QueryCreator.class);
        internalContainer.registerComponentImplementation(ReleaseNoteManager.class);
        internalContainer.registerComponentImplementation(ApplicationProperties.class,
ApplicationPropertiesImpl.class);
        internalContainer.registerComponentImplementation(VelocityManager.class,
JiraVelocityManager.class);
        internalContainer.registerComponentImplementation(ConstantsManager.class,
DefaultConstantsManager.class);
        internalContainer.registerComponentImplementation(PermissionSchemeManager.class,
DefaultPermissionSchemeManager.class);
        internalContainer.registerComponentImplementation(CacheManager.class,
DefaultCacheManager.class);
        internalContainer.registerComponentImplementation(TrackbackManager.class,
TrackbackManagerImpl.class);
        internalContainer.registerComponentImplementation(WatcherManager.class,
DefaultWatcherManager.class);
        internalContainer.registerComponentImplementation(EntityUtils.class);
        internalContainer.registerComponentImplementation(AttachmentManager.class,
DefaultAttachmentManager.class);
        internalContainer.registerComponentImplementation(FieldManager.class,
DefaultFieldManager.class);
      .....
     }
```

If you wanted to register your overridden version of a pico-registered class, you could just register yours instead of the default in ComponentManager above.

Overriding components in JIRA 3.0

Mike tells me there is an even better way of registering your own Pico components, using plugins, than the technique described here. See JIRA Plugin Guide

In 3.0 there is a nicer way of plugging in your customizations that avoids modifying JIRA code. In jira-application.properties, register an *extension container provider*:

```
jira.extension.container.provider = com.mycompany.jira.MyContainerProvider
```

In this class, you can register your own implementations of interfaces, which will be used in preference to the defaults in ComponentManager:

MyContainerProvider.java

```
package com.mycompany.jira;

import org.picocontainer.PicoContainer;
import org.picocontainer.defaults.DefaultPicoContainer;
import com.atlassian.jira.config.component.ProfilingComponentAdapterFactory;
import com.atlassian.jira.web.action.issue.BugAssociatorPrefs;
import com.atlassian.jira.security.PermissionManager;
import com.atlassian.jira.permission.PermissionSchemeManager;

import com.mycompany.jira.MyBugAssociatorPrefs;
import com.mycompany.jira.MyPermissionManager;
import com.mycompany.jira.MyPermissionSchemeManager;

public class MyContainerProvider implements ContainerProvider
{
    private DefaultPicoContainer container;

    public PicoContainer getContainer(PicoContainer parent)
    {
        if (container == null)
            buildContainer(parent);
        return container;
    }

    private void buildContainer(PicoContainer parent)
    {
        this.container = new DefaultPicoContainer(new ProfilingComponentAdapterFactory(), parent);
        container.registerComponentImplementation(BugAssociatorPrefs.class,
MyBugAssociatorPrefs.class);
        container.registerComponentImplementation(PermissionManager.class, MyPermissionManager.class);
        container.registerComponentImplementation(PermissionSchemeManager.class,
MyPermissionSchemeManager.class);
    }
}
```

Here we have registered our own implementations of three classes, after delegating to the default (so ours will take precedence). You can now keep MyContainerProvider and your modified com.mycompany.jira.* classes in their own jar, which can be dropped into any JIRA instance to customize it to your needs.

Database Schema

Please note that JIRA 3.7 has introduced some changes to the database schema.

JIRA uses Entity Engine module of the OfBiz suite to communicate with the database. You can learn more about the Entity Engine by reading its online documentation.

The database schema is described in the `entitymodel.xml` found in the `WEB-INF/classes/entitydefs` directory under the JIRA web application. The `entitymodel.xml` file has an XML definition of all JIRA's database tables, table columns and their data type. Some of the relationships between tables also appear in the file.

If you are using JIRA's API you will notice that a lot of code deals with `GenericValue` objects. The `GenericValue` is an OfBiz entity engine object. Each GenericValue object represents a record in the database.

To get a value of a field from a GenericValue you will need to use the relevant getter method for the field's type. For example:

```
GenericValue project = ...
String name = project.getString("name");
Long id = project.getLong("id");
```

The list of valid fields for each entity can be obtained by looking the entity's definition in the `WEB-INF/classes/entitydefs/entitymodel.xml` file. For the above example, one needs to look at the "Project" entity.

Relationships between tables

Some of the relationships between JIRA's tables in the database are documented below:

- Issue Fields
  - Simple fields
  - User details
  - Components and versions
  - Issue links
- Custom fields
  - Custom field configuration options
  - Custom field configuration default value
  - Custom field configuration schemes
  - Custom field configuration scopes
- Change History
- Users and Groups
- Issue status and workflow
- Issue status
- Issue workflow step
- How status and step relate
- Summary
  - Example SQLs

Issue Fields

This page shows how to examine each of a JIRA issue's fields via SQL. We will use JRA-3166 as a sample issue in our queries.

Simple fields

Most fields in JIRA are kept in the `jiraissue` table:

```
mysql> desc jiraissue;
+----------------------+---------------+------+-----+---------+-------+
| Field                | Type          | Null | Key | Default | Extra |
+----------------------+---------------+------+-----+---------+-------+
| ID                   | decimal(18,0) | NO   | PRI |         |       |
| pkey                 | varchar(255)  | YES  | MUL | NULL    |       |
| PROJECT              | decimal(18,0) | YES  | MUL | NULL    |       |
| REPORTER             | varchar(255)  | YES  |     | NULL    |       |
| ASSIGNEE             | varchar(255)  | YES  | MUL | NULL    |       |
| issuetype            | varchar(255)  | YES  |     | NULL    |       |
| SUMMARY              | varchar(255)  | YES  |     | NULL    |       |
| DESCRIPTION          | longtext      | YES  |     | NULL    |       |
| ENVIRONMENT          | longtext      | YES  |     | NULL    |       |
| PRIORITY             | varchar(255)  | YES  |     | NULL    |       |
| RESOLUTION           | varchar(255)  | YES  |     | NULL    |       |
| issuestatus          | varchar(255)  | YES  |     | NULL    |       |
| CREATED              | datetime      | YES  |     | NULL    |       |
| UPDATED              | datetime      | YES  |     | NULL    |       |
| DUEDATE              | datetime      | YES  |     | NULL    |       |
| VOTES                | decimal(18,0) | YES  |     | NULL    |       |
| TIMEORIGINALESTIMATE | decimal(18,0) | YES  |     | NULL    |       |
| TIMEESTIMATE         | decimal(18,0) | YES  |     | NULL    |       |
| TIMESPENT            | decimal(18,0) | YES  |     | NULL    |       |
| WORKFLOW_ID          | decimal(18,0) | YES  |     | NULL    |       |
| SECURITY             | decimal(18,0) | YES  |     | NULL    |       |
| FIXFOR               | decimal(18,0) | YES  |     | NULL    |       |
| COMPONENT            | decimal(18,0) | YES  |     | NULL    |       |
+----------------------+---------------+------+-----+---------+-------+
```

They can be retrieved with a regular select:

```
mysql> select id, pkey, project, reporter, assignee, issuetype, summary from jiraissue where
pkey='JRA-3166';
+-------+----------+---------+-----------+----------+-----------+--------------------------------+
| id    | pkey     | project | reporter  | assignee | issuetype | summary                        |
+-------+----------+---------+-----------+----------+-----------+--------------------------------+
| 16550 | JRA-3166 |   10240 | mvleeuwen | NULL     | 2         | Database consistency check tool |
+-------+----------+---------+-----------+----------+-----------+--------------------------------+
```

User details

Say we wish to find out the email address and other details about our reporter, `mvleeuwen`. First we find this user's ID:

```
mysql> select id from userbase where username='mvleeuwen';
+-------+
| id    |
+-------+
| 13841 |
+-------+
```

Then use it to look up 'properties' of this userbase record (stored in propertysets. Each property has a record in the `propertyentry` table specifying its name and type, and a record in one of `propertystring`, `propertydecimal`, `propertydate`, `propertytext`, `propertydata` or `propertynumber`, depending on the type.

```
mysql> desc propertyentry;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| ID           | decimal(18,0) | NO   | PRI |         |       |
| ENTITY_NAME  | varchar(255)  | YES  |     | NULL    |       |
| ENTITY_ID    | decimal(18,0) | YES  | MUL | NULL    |       |
| PROPERTY_KEY | varchar(255)  | YES  |     | NULL    |       |
| propertytype | decimal(9,0)  | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+

mysql> select * from propertyentry where ENTITY_NAME='OSUser' and ENTITY_ID=(select id from userbase
where username='mvleeuwen');
+-------+-------------+-----------+--------------+--------------+
| ID    | ENTITY_NAME | ENTITY_ID | PROPERTY_KEY | propertytype |
+-------+-------------+-----------+--------------+--------------+
| 18352 | OSUser      |     13841 | email        |            5 |
| 18353 | OSUser      |     13841 | fullName     |            5 |
+-------+-------------+-----------+--------------+--------------+
```

So `email` and `fullName` are of type 5. which means the `propertystring` table. Here is the list of `propertytype` to table mappings:

propertyentry.propertytype value	Table value is stored in	Used for
1	propertynumber	Boolean values, eg. user preferences
5	propertystring	Most fields, eg. full names, email addresses
6	propertytext	Large blocks of text, eg. the introduction text, HTML portletconfigurations
2/3	propertydecimal	Unused in JIRA
7	propertydate	Unused in JIRA
10	propertydata	Unused in JIRA

So the `email` and `fullName` properties are strings, and so can be found in the `propertystring` table:

```
mysql> select * from propertystring where id in (18352, 18353);
+-------+--------------------+
| ID    | propertyvalue      |
+-------+--------------------+
| 18352 | lemval@zonnet.nl   |
| 18353 | Michael van Leeuwen |
+-------+--------------------+
```

Components and versions

Since each issue can have multiple components/versions, there is a join table between `jiraissue` and `version`/`component` tables called `nodeassociation`:

```
mysql> desc nodeassociation;
+--------------------+---------------+------+-----+---------+-------+
| Field              | Type          | Null | Key | Default | Extra |
+--------------------+---------------+------+-----+---------+-------+
| SOURCE_NODE_ID     | decimal(18,0) | NO   | PRI |         |       |
| SOURCE_NODE_ENTITY | varchar(60)   | NO   | PRI |         |       |
| SINK_NODE_ID       | decimal(18,0) | NO   | PRI |         |       |
| SINK_NODE_ENTITY   | varchar(60)   | NO   | PRI |         |       |
| ASSOCIATION_TYPE   | varchar(60)   | NO   | PRI |         |       |
| SEQUENCE           | decimal(9,0)  | YES  |     | NULL    |       |
+--------------------+---------------+------+-----+---------+-------+

mysql> select distinct SOURCE_NODE_ENTITY from nodeassociation;
+--------------------+
| SOURCE_NODE_ENTITY |
+--------------------+
| Issue              |
| Project            |
+--------------------+

mysql> select distinct SINK_NODE_ENTITY from nodeassociation;
+-----------------------+
| SINK_NODE_ENTITY      |
+-----------------------+
| IssueSecurityScheme   |
| PermissionScheme      |
| IssueTypeScreenScheme |
| NotificationScheme    |
| ProjectCategory       |
| FieldLayoutScheme     |
| Component             |
| Version               |
+-----------------------+

mysql> select distinct ASSOCIATION_TYPE from nodeassociation;
+------------------+
| ASSOCIATION_TYPE |
+------------------+
| IssueVersion     |
| IssueFixVersion  |
| IssueComponent   |
| ProjectScheme    |
| ProjectCategory  |
+------------------+
```

So to get fix-for versions of an issue, run:

```
mysql> select * from projectversion where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueFixVersion' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
| ID    | PROJECT | vname | DESCRIPTION | SEQUENCE | RELEASED | ARCHIVED | URL  | RELEASEDATE |
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
| 11614 |   10240 | 3.6   | NULL        |      131 | NULL     | NULL     | NULL | NULL        |
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
```

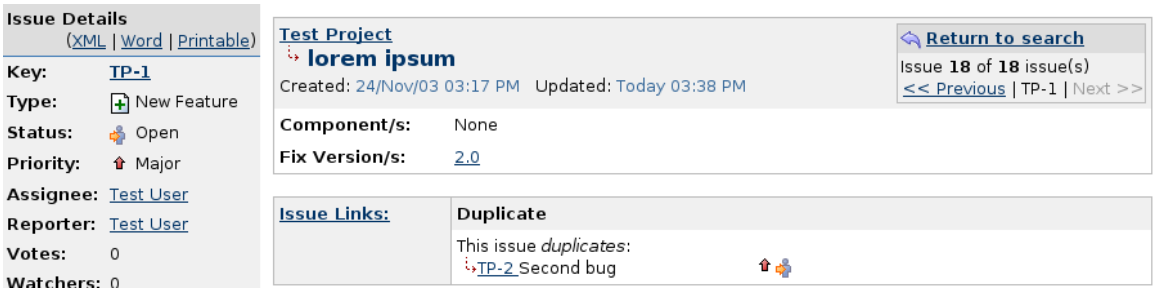
Similarly with affects versions:

```
mysql> select * from projectversion where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueVersion' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
ID    | PROJECT | vname              | DESCRIPTION | SEQUENCE | RELEASED | ARCHIVED | URL  |
RELEASEDATE         |
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
10931 |   10240 |  3.0.3 Professional | NULL        |       73 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
| 10930 |   10240 |  3.0.3 Standard     | NULL        |       72 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
| 10932 |   10240 |  3.0.3 Enterprise   | NULL        |       74 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
```

and components:

```
mysql> select * from component where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueComponent' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+---------------+-------------+------+------+-------------+
| ID    | PROJECT | cname         | description | URL  | LEAD | ASSIGNEETYPE |
+-------+---------+---------------+-------------+------+------+-------------+
| 10126 |   10240 | Web interface | NULL        | NULL | NULL |         NULL |
+-------+---------+---------------+-------------+------+------+-------------+
```

Issue links



JIRA issue links are stored in the `issuelink` table, which simply links the IDs of two issues together, and records the link type:

```
mysql> desc issuelink;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ID          | decimal(18,0) | NO   | PRI |         |       |
| LINKTYPE    | decimal(18,0) | YES  | MUL | NULL    |       |
| SOURCE      | decimal(18,0) | YES  | MUL | NULL    |       |
| DESTINATION | decimal(18,0) | YES  | MUL | NULL    |       |
| SEQUENCE    | decimal(18,0) | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

For instance, to list all links between TP-1 and TP-2:

```
mysql> select * from issuelink where SOURCE=(select id from jiraissue where pkey='TP-1') and
DESTINATION=(select id from jiraissue where pkey='TP-2');
+-------+----------+--------+-------------+----------+
| ID    | LINKTYPE | SOURCE | DESTINATION | SEQUENCE |
+-------+----------+--------+-------------+----------+
| 10020 |    10000 |  10000 |       10010 |     NULL |
+-------+----------+--------+-------------+----------+
1 row in set (0.00 sec)
```

Link types are defined in `issuelinktype`. This query prints all links in the system with their type:

```
mysql> select j1.pkey, issuelinktype.INWARD, j2.pkey from jiraissue j1, issuelink, issuelinktype,
jiraissue j2 where j1.id=issuelink.SOURCE and j2.id=issuelink.DESTINATION and
issuelinktype.id=issuelink.linktype;
+-------+---------------------+-------+
| pkey  | INWARD              | pkey  |
+-------+---------------------+-------+
| TP-4  | jira_subtask_inward | TP-5  |
| TP-4  | jira_subtask_inward | TP-7  |
| TP-4  | jira_subtask_inward | TP-8  |
| TP-11 | jira_subtask_inward | TP-12 |
| TP-4  | jira_subtask_inward | TP-6  |
| TP-1  | is duplicated by    | TP-2  |
+-------+---------------------+-------+
6 rows in set (0.00 sec)
```

Subtasks

As shown in the last query, JIRA records the issue-subtask relation as a link. The "subtask" link type is hidden in the user interface (indicated by the 'pstyle' value below), but visible in the database:

```
mysql> select * from issuelinktype;
+-------+------------------+---------------------+----------------------+--------------+
| ID    | LINKNAME         | INWARD              | OUTWARD              | pstyle       |
+-------+------------------+---------------------+----------------------+--------------+
| 10000 | Duplicate        | is duplicated by    | duplicates           | NULL         |
| 10001 | jira_subtask_link | jira_subtask_inward | jira_subtask_outward | jira_subtask |
+-------+------------------+---------------------+----------------------+--------------+
2 rows in set (0.00 sec)
```

This means it is possible to convert an issue to a subtask, or vice-versa, by tweaking `issuelink` records.

Custom fields have their own set of tables. For details, see Custom fields

Custom fields

Custom fields defined in the system are stored in the `customfield` table, and instances of custom fields are stored in `customfieldvalue`:

```
mysql> desc customfieldvalue;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ID          | decimal(18,0) | NO   | PRI |         |       |
| ISSUE       | decimal(18,0) | YES  | MUL | NULL    |       |
| CUSTOMFIELD | decimal(18,0) | YES  |     | NULL    |       |
| PARENTKEY   | varchar(255)  | YES  |     | NULL    |       |
| STRINGVALUE | varchar(255)  | YES  |     | NULL    |       |
| NUMBERVALUE | decimal(18,6) | YES  |     | NULL    |       |
| TEXTVALUE   | longtext      | YES  |     | NULL    |       |
| DATEVALUE   | datetime      | YES  |     | NULL    |       |
| VALUETYPE   | varchar(255)  | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
```

We can print all custom field values for an issue with:

```
mysql> select * from customfieldvalue where issue=(select id from jiraissue where pkey='JRA-5448');
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
ID    | ISSUE | CUSTOMFIELD | PARENTKEY | STRINGVALUE | NUMBERVALUE | TEXTVALUE | DATEVALUE
| VALUETYPE |
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
23276 | 22160 |       10190 | NULL      | NULL        |        NULL | NULL      | 2004-12-07 17:25:58
| NULL      |
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
```

and we can see what type of custom field this (10190) is with:

```
mysql> select * from customfield where id=10190;
+-------+-------------------------------------------------+-------------------------------------------------
ID    | CUSTOMFIELDTYPEKEY                              | CUSTOMFIELDSEARCHERKEY
| cfname          | DESCRIPTION | defaultvalue | FIELDTYPE | PROJECT | ISSUETYPE |
+-------+-------------------------------------------------+-------------------------------------------------
10190 | com.atlassian.jira.ext.charting:resolutiondate |
com.atlassian.jira.ext.charting:resolutiondatesearcher | Resolution Date | NULL        | NULL
|      NULL |    NULL | NULL      |
+-------+-------------------------------------------------+-------------------------------------------------
```

(ie. it's a "Resolution Date").

This query identifies a particular custom field value in a particular issue:

```
mysql> select stringvalue from customfieldvalue where customfield=(select id from customfield where
cfname='Urgency') and issue=(select id from jiraissue where pkey='FOR-845');
+-------------+
| stringvalue |
+-------------+
| Low         |
+-------------+
1 row in set (0.33 sec)
```

If the custom field has multiple values (multi-select or multi-user picker), each issue can have multiple `customfieldvalue` rows:

```
mysql> select * from customfieldvalue where customfield=(select ID from customfield where
cfname='MultiUser');
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
ID    | ISSUE | CUSTOMFIELD | PARENTKEY | STRINGVALUE | NUMBERVALUE | TEXTVALUE | DATEVALUE |
VALUETYPE |
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
10002 | 10060 |       10000 | NULL      | bob         |        NULL | NULL      | NULL      | NULL
|
| 10003 | 10060 |       10000 | NULL      | jeff        |        NULL | NULL      | NULL      | NULL
|
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
rows in set (0.00 sec)
```

Here issue 10060 has two users, `bob` and `jeff` in its MultiUser custom field.

Custom field configuration options

The option sets (1, 2, 3 and A, B, C) are stored in the **customfieldoption** table:

mysql> select * from customfieldoption where customfieldconfig=10031;

ID	CUSTOMFIELD	CUSTOMFIELDCONFIG	PARENTOPTIONID	SEQUENCE	customvalue	optiontype
10000	10001	10031	NULL	0	1	NULL
10001	10001	10031	NULL	1	2	NULL
10002	10001	10031	NULL	2	3	NULL

mysql> select * from customfieldoption where customfieldconfig=10032;

ID	CUSTOMFIELD	CUSTOMFIELDCONFIG	PARENTOPTIONID	SEQUENCE	customvalue	optiontype
10003	10001	10032	NULL	0	A	NULL
10004	10001	10032	NULL	1	B	NULL
10005	10001	10032	NULL	2	C	NULL

Custom field configuration default value

The custom field default value is stored in the **genericconfiguration** table. Since this table must store a value for any custom field type (cascading selects, multi-selects, etc) the value is encoded as XML.

If we were to set a default value of "2" for our "Default Configuration Scheme for SelectCF", it would be recorded as:

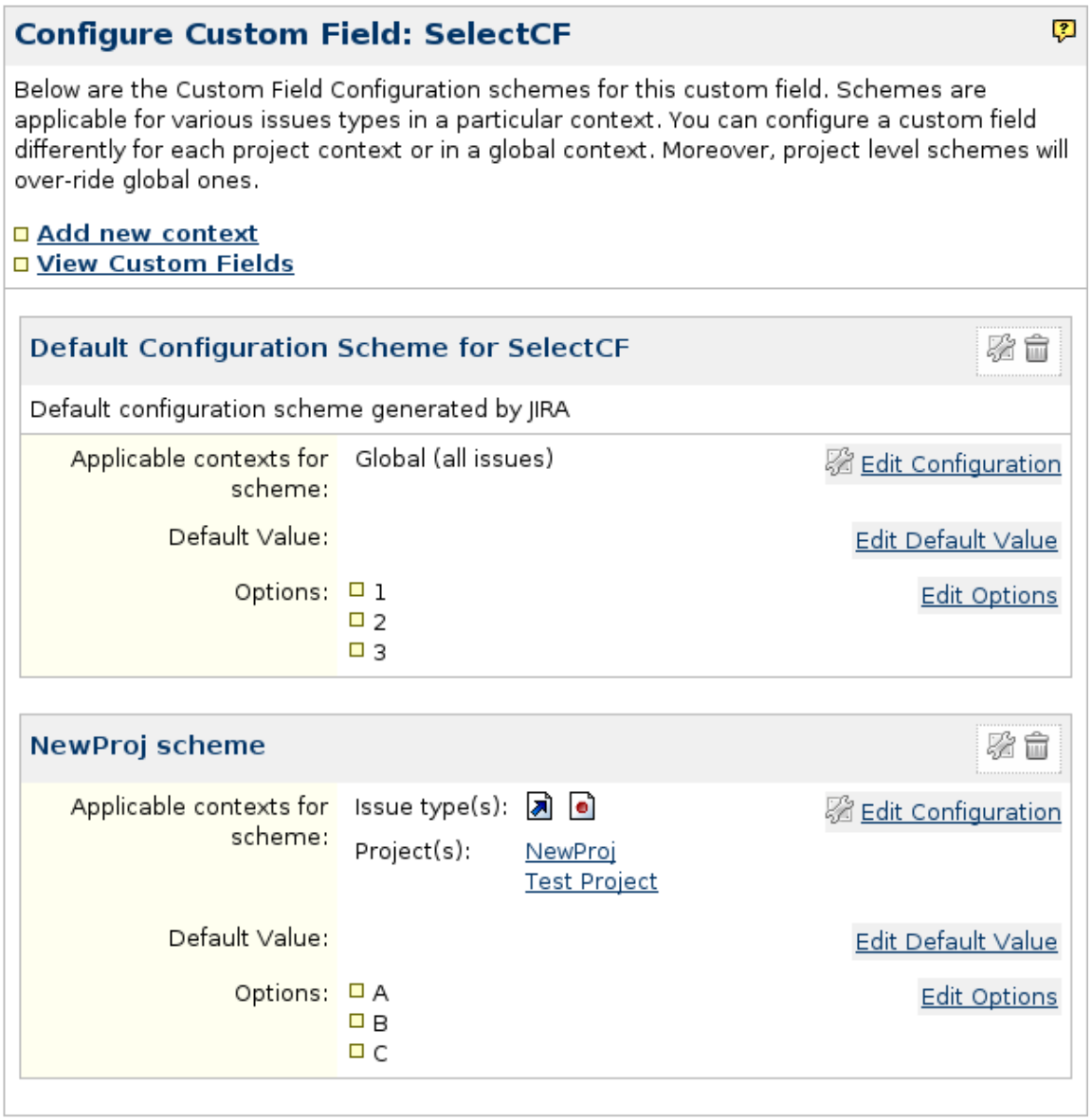
mysql> select * from genericconfiguration where ID=10031;

ID	DATATYPE	DATAKEY	XMLVALUE
10031	DefaultValue	10030	<string>2</string>

Custom field configuration schemes

JIRA custom fields can have different default values and possible values for each project and/or issue type. This is set up by clicking 'Configure' in the custom field definition.

For instance, in this screenshot the "SelectCF" select-list field will have values 1, 2, 3 for all projects except bugs and improvements in "NewProj" and"Test Project", which will have values A, B and C:



Custom field configuration scopes

In the database, these custom field configuration schemes are stored in the **fieldconfigscheme** table

mysql> select * from fieldconfigscheme where id in (10031,10032);

ID	confignam e	DESCRIPTION	FIELDID	CUSTOMFIELD
10031	Default Configuration Scheme for SelectCF	Default configuration scheme generated by JIRA	customfield_10001	NULL
10032	NewProj scheme		customfield_10001	NULL

The **projects in scope** for each of these schemes is listed as records (one per project) in the **configurationcontext** table:

mysql> select * from configurationcontext where fieldconfigscheme=10031;

ID	PROJECTCATEGORY	PROJECT	customfield	FIELDCONFIGSCHEME
10053	NULL	NULL	customfield_10001	10031

(Here showing that that the "Default Configuration Scheme for SelectCF" applies to all projects)

mysql> select * from configurationcontext where fieldconfigscheme=10032;

ID	PROJECTCATEGORY	PROJECT	customfield	FIELDCONFIGSCHEME
10054	NULL	10000	customfield_10001	10032
10055	NULL	10010	customfield_10001	10032

(Here showing that "NewProj scheme" is restricted to projects with ids 10000 and 10010 ("Test Project" and "NewProj")).

Finally, the **issue types in scope** for each scheme is listed as records (one per issue type) in the **fieldconfigschemeissuetype** table:

mysql> select * from fieldconfigschemeissuetype where fieldconfigscheme = 10031;

ID	ISSUETYPE	FIELDCONFIGSCHEME	FIELDCONFIGURATION
10051	NULL	10031	10031

(Here showing that "Default Configuration Scheme for SelectCF" is not limited to any issue types)

mysql> select * from fieldconfigschemeissuetype where fieldconfigscheme = 10032;

ID	ISSUETYPE	FIELDCONFIGSCHEME	FIELDCONFIGURATION
10052	1	10032	10032
10053	4	10032	10032

(Here showing that "Newproj scheme" is limited to issue types with IDs 1 and 4).

Note that there should *always be a record* in `configurationcontext` and `fieldconfigschemeissuetype` for each issue type configuration scheme. If the scheme isn't restricted to any projects or issue types, the `project` and `issuetype` columns of the respective tables should be `NULL`. Incidentally JIRA has/had a bug where it didn't leave an entry when deleting an issue type (JRA-10461), so if you are making changes manually, don't make the same mistake.

Change History

Change History Database Tables

JIRA stores the Change History records of each issue in the `changegroup` and `changeitem` tables.

Each change to the issue triggered by a user inserts one record into the `changegroup` table. Each changegroup table record describes which issue it refers to, the time of the change and the user who has performed the change (null for a non-logged in user).

```
mysql> select * from changegroup;
+-------+---------+--------+---------------------+
| ID    | issueid | AUTHOR | CREATED             |
+-------+---------+--------+---------------------+
| 10000 |   10000 | admin  | 2005-06-09 15:16:39 |
| 10751 |   10000 | admin  | 2005-06-10 00:00:00 |
+-------+---------+--------+---------------------+
```

Each changegroup record refers to one or many `changeitem` records. Each `changeitem` record describes the issue field that has been updated and its old and new values. The `OLDVALUE` column records the id of the changed enity (e.g. status) while `OLDSTRING` records the name fo the entity, so that if the entity is removed from the system the change history for an issue can still be displayed. The `NEWVALUE` and `NEWSTRING` columns are similar in nature.

```
mysql> select * from changeitem;
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
| ID    | groupid | FIELDTYPE | FIELD      | OLDVALUE | OLDSTRING | NEWVALUE | NEWSTRING |
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
| 10000 |   10000 | jira      | status     | 1        | Open      | 6        | Closed    |
| 10001 |   10000 | jira      | resolution | NULL     | NULL      | 1        | Fixed     |
| 11404 |   10751 | jira      | status     | 1        | Open      | 6        | Closed    |
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
```

Inserting change history records

When writing tools that import data into JIRA, it is sometimes required to import change history. To do this please first insert a record into the `changegroup` table with a valid issue id:

```
insert into changegroup values (20000,10000,'admin','2005-06-12');
```

The issues are stored in the `jiraissue` table:

```
mysql> select id, pkey from jiraissue;
+-------+-------+
| id    | pkey  |
+-------+-------+
| 10000 | TST-1 |
+-------+-------+
```

And then insert the required number of `changeitem` records referencing the inserted `changegroup` record:

```
insert into changeitem values (11000, 20000, 'jira','status','1','Open','6','Closed');
```

The SEQUENCE_VALUE_ITEM table

The SEQUENCE_VALUE_ITEM table is used to record, in a database independent way, the maximum ID used in each of JIRA's database tables:

```
mysql> select * from SEQUENCE_VALUE_ITEM;
+---------------------------+--------+
| SEQ_NAME                  | SEQ_ID |
+---------------------------+--------+
| Action                    |  10310 |
| ChangeGroup               |  11050 |
| ChangeItem                |  11320 |
| ColumnLayout              |  10040 |
| ColumnLayoutItem          |  10120 |
| Component                 |  10110 |
| ConfigurationContext      |  10170 |
| SchemeIssueSecurities     |  10040 |
...
```

Actually, Ofbiz allocates IDs in batches of 10, so the SEQ_ID is the next available ID rounded up to the nearest 10. So you might have:

```
mysql> select max(ID) from jiraaction;
+---------+
| max(ID) |
+---------+
|   10303 |
+---------+
1 row in set (0.04 sec)

mysql> select * from SEQUENCE_VALUE_ITEM where SEQ_NAME='Action';
+----------+--------+
| SEQ_NAME | SEQ_ID |
+----------+--------+
| Action   |  10310 |
+----------+--------+
1 row in set (0.01 sec)
```

Where 10310 is the nearest 10 above 10303.

The `SEQ_NAME` column refers to the database table name defined in `WEB-INF/classes/entitydefs/entitymodel.xml` (eg. "Action" is `jiraaction`).

Manually inserting records

The implication of this is that if you want to manually insert records into JIRA database tables, you **must update SEQUENCE_VALUE_ITEM yourself**. Set the relevant rows' SEQ_ID values to a value greater than the actual maximum ID in the table. You will then need to restart JIRA to ensure all database caches are reset.

Retrieving Change History using JIRA's API

The best way to retrieve change history entries is:

```
actionManager.getChangeHistory(getIssue(), authenticationContext.getUser());
```

You can declare dependency on JiraAuthenticationContext and ActionManager in the constructor of your plugin as described in PicoContainer and JIRA.

The getChangeHistory method returns `ChangeHistory` objects on which you can call the getChangeItems() method. This returns a List if GenericValue objects each one representing an issue field update. To check the field that was updated do:

String fieldName = changeItem.getString("field")

GenericValues are described in Database Schema.

Users and Groups

User and Group Tables

User Tables

JIRA uses an open source framework called OSUser to manage its users. The user records are stored in the `userbase` database table. The table holds little information:

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| ID            | decimal(18,0) |      | PRI | 0       |       |
| username      | varchar(255)  | YES  | MUL | NULL    |       |
| PASSWORD_HASH | varchar(255)  | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

OSUser also stores some properties for the user. The properties are: *full name* and *e-mail*. These properties are stored in the `propertyentry` and `propertystring` tables.

For each user the `propertyentry` table:

```
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| ID           | decimal(18,0) |      | PRI | 0       |       |
| ENTITY_NAME  | varchar(255)  | YES  | MUL | NULL    |       |
| ENTITY_ID    | decimal(18,0) | YES  |     | NULL    |       |
| PROPERTY_KEY | varchar(255)  | YES  |     | NULL    |       |
| propertytype | decimal(9,0)  | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
```

stores 'OSUser' in the ENTITY_NAME column, the id of the `userbase` record in the ENTITY_ID column, and 'fullName' or 'email' in the PROPERTY_KEY column.

The `propertystring` table:

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| ID            | decimal(18,0) |      | PRI | 0       |       |
| propertyvalue | text          | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

stores the actual values of the properties. The ID column is the same as the ID of the `propertyentry` record and the propertyvalue column would store the full name or e-mail of the user.

Group Tables

The groups are stored in the `groupbase` table:

```
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| ID        | decimal(18,0) |      | PRI | 0       |       |
| groupname | varchar(255)  | YES  | MUL | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
```

Group Membership

The `membershipbase` table records which users belong to which groups:

```
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| ID         | decimal(18,0) |      | PRI | 0       |       |
| USER_NAME  | varchar(255)  | YES  | MUL | NULL    |       |
| GROUP_NAME | varchar(255)  | YES  | MUL | NULL    |       |
+------------+---------------+------+-----+---------+-------+
```

The USER_NAME column is set to the username column in the `userbase` table. The GROUP_NAME is set to the groupname record in the `groupbase` table.

Watches and Votes

Watches and votes are recorded in the userassociation table:

```
+------------------+---------------+------+-----+---------+-------+
| Field            | Type          | Null | Key | Default | Extra |
+------------------+---------------+------+-----+---------+-------+
| SOURCE_NAME      | varchar(60)   | NO   | PRI |         |       |
| SINK_NODE_ID     | decimal(18,0) | NO   | PRI |         |       |
| SINK_NODE_ENTITY | varchar(60)   | NO   | PRI |         |       |
| ASSOCIATION_TYPE | varchar(60)   | NO   | PRI |         |       |
| SEQUENCE         | decimal(9,0)  | YES  |     | NULL    |       |
+------------------+---------------+------+-----+---------+-------+
```

For example:

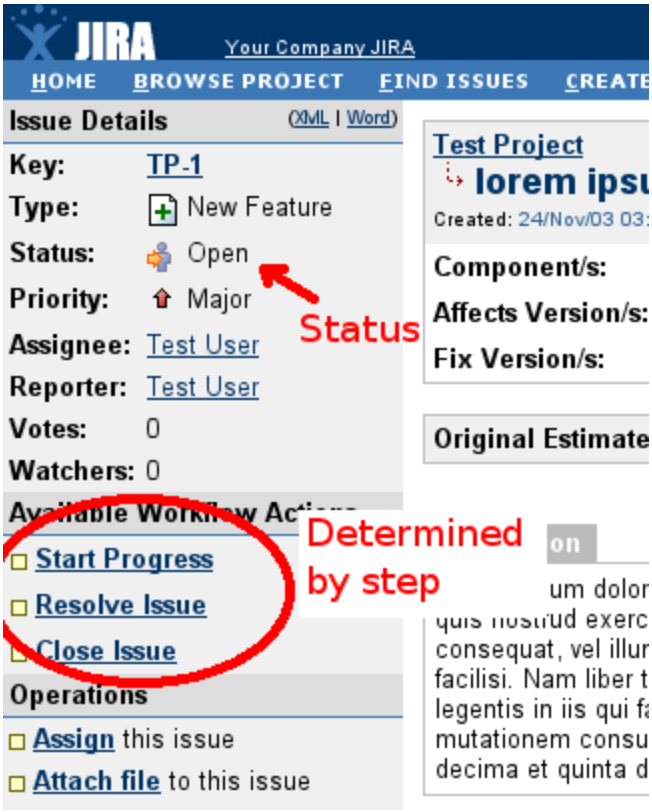
```
mysql> select * from userassociation;
+--------------+-------------+------------------+------------------+----------+
| SOURCE_NAME  | SINK_NODE_ID | SINK_NODE_ENTITY | ASSOCIATION_TYPE | SEQUENCE |
+--------------+-------------+------------------+------------------+----------+
| asmith       |     108433  | Issue            | WatchIssue       |     NULL |
| droberts     |     100915  | Issue            | WatchIssue       |     NULL |
| dfernandez   |     106387  | Issue            | VoteIssue        |     NULL |
...
```

For example, here user 'asmith' is watching issue with id 108433.

Issue status and workflow

This page describes the database tables involved in issue workflow. It will be useful for people who wish to insert issues into the database manually, or diagnose/fix corrupted databases.
JIRA issues have both:

- a **status** (Open, Closed, In Progress etc).
- a **workflow step**, which governs which transitions are available



Issue status

In the database, the **status** (Open, Closed etc) is stored on the `jiraissue` table:

```
mysql> select issuestatus from jiraissue where pkey='TP-1';
+-------------+
| issuestatus |
+-------------+
| 1           |
+-------------+
1 row in set (0.00 sec)

mysql> select pname from issuestatus, jiraissue where issuestatus.id=jiraissue.issuestatus and
pkey='TP-1';
+-------+
| pname |
+-------+
| Open  |
+-------+
1 row in set (0.00 sec)
```

Issue workflow step

Originally JIRA issues only had a status. Then in version 2.0, *workflow* was added, so that transitions between statuses could be customized. An issue's workflow step is stored in new tables, referenced from `jiraissue` by the `workflow_id`:

```
mysql> select * from OS_WFENTRY where ID=(select workflow_id from jiraissue where pkey='TP-1');
+-------+------+-------------+-------+
| ID    | NAME | INITIALIZED | STATE |
+-------+------+-------------+-------+
| 10000 | jira | 0           | 1     |
+-------+------+-------------+-------+
1 row in set (0.02 sec)
```

The TP-1 issue's OS_WFENTRY row indicates that the issue uses the 'jira' (default, built-in) workflow.

```
mysql> select * from OS_CURRENTSTEP where ENTRY_ID=(select workflow_id from jiraissue where
pkey='TP-1');
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
ID    | ENTRY_ID | STEP_ID | ACTION_ID | OWNER | START_DATE          | DUE_DATE | FINISH_DATE | STATUS
| CALLER |
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
10000 | 10000    | 1       | 0         |       | 2003-11-24 15:17:50 |          |             | Open
|        |
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
row in set (0.13 sec)
```

The issue's OS_CURRENTSTEP row specifies the issue's *current step*. The only field really used is `STEP_ID`. This references a step definition in the workflow:

```
<step id="1" name="Open">
```

The workflow definition for the built-in 'jira' workflow can be seen in `atlassian-jira/WEB-INF/classes/jira-workflow.xml`

How status and step relate

An issue's status and workflow step are kept in synch:

```
mysql> select issuestatus.pname status, issuestatus, OS_CURRENTSTEP.STEP_ID, OS_CURRENTSTEP.STATUS
from
   issuestatus, jiraissue, OS_CURRENTSTEP where
   issuestatus.id=jiraissue.issuestatus and jiraissue.workflow_id=OS_CURRENTSTEP.ENTRY_ID;
+-------------+-------------+---------+----------+
| status      | issuestatus | STEP_ID | STATUS   |
+-------------+-------------+---------+----------+
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
...
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| In Progress | 3           | 3       | Underway |
| Closed      | 6           | 6       | Closed   |
+-------------+-------------+---------+----------+
32 rows in set (0.00 sec)

mysql>
```

Status and step are kept in synch is with a workflow post-function (UpdateIssueStatusFunction), which updates the status whenever the step changes.

If the step gets out of synch with the status, then incorrect (or no) workflow operations appear on the issue page. Eg. if OS_CURRENTSTEP.STEP_ID was 6 ("Closed") when jiraissue.issuestatus was 1 ("Open"), then the issue would have only one transition ("Reopen issue") which would break if anyone clicked on it.

Summary

- For each `jiraissue` row, there is a `OS_CURRENTSTEP` and `OS_WFENTRY` row.
- `OS_WFENTRY` specifies the applicable workflow. `OS_CURRENTSTEP` specifies the step in that workflow.
- The relations are:
  - jiraissue.WORKFLOW_ID == OS_WFENTRY.ID
  - jiraissue.WORKFLOW_ID == OS_CURRENTSTEP.ENTRY_ID

Example SQLs

Some examples of SQLs that can be run against the JIRA schema:
Example SQL queries for JIRA

Change History

Change History Database Tables

JIRA stores the Change History records of each issue in the `changegroup` and `changeitem` tables.

Each change to the issue triggered by a user inserts one record into the `changegroup` table. Each changegroup table record describes which issue it refers to, the time of the change and the user who has performed the change (null for a non-logged in user).

```
mysql> select * from changegroup;
+-------+---------+--------+---------------------+
| ID    | issueid | AUTHOR | CREATED             |
+-------+---------+--------+---------------------+
| 10000 |   10000 | admin  | 2005-06-09 15:16:39 |
| 10751 |   10000 | admin  | 2005-06-10 00:00:00 |
+-------+---------+--------+---------------------+
```

Each changegroup record refers to one or many `changeitem` records. Each `changeitem` record describes the issue field that has been updated and its old and new values. The `OLDVALUE` column records the id of the changed enity (e.g. status) while `OLDSTRING` records the name fo the entity, so that if the entity is removed from the system the change history for an issue can still be displayed. The `NEWVALUE` and `NEWSTRING` columns are similar in nature.

```
mysql> select * from changeitem;
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
| ID    | groupid | FIELDTYPE | FIELD      | OLDVALUE | OLDSTRING | NEWVALUE | NEWSTRING |
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
| 10000 |   10000 | jira      | status     | 1        | Open      | 6        | Closed    |
| 10001 |   10000 | jira      | resolution | NULL     | NULL      | 1        | Fixed     |
| 11404 |   10751 | jira      | status     | 1        | Open      | 6        | Closed    |
+-------+---------+-----------+------------+----------+-----------+----------+-----------+
```

Inserting change history records

When writing tools that import data into JIRA, it is sometimes required to import change history. To do this please first insert a record into the `changegroup` table with a valid issue id:

```
insert into changegroup values (20000,10000,'admin','2005-06-12');
```

The issues are stored in the `jiraissue` table:

```
mysql> select id, pkey from jiraissue;
+-------+-------+
| id    | pkey  |
+-------+-------+
| 10000 | TST-1 |
+-------+-------+
```

And then insert the required number of `changeitem` records referencing the inserted `changegroup` record:

```
insert into changeitem values (11000, 20000, 'jira','status','1','Open','6','Closed');
```

The SEQUENCE_VALUE_ITEM table

The SEQUENCE_VALUE_ITEM table is used to record, in a database independent way, the maximum ID used in each of JIRA's database tables:

```
mysql> select * from SEQUENCE_VALUE_ITEM;
+----------------------------+--------+
| SEQ_NAME                   | SEQ_ID |
+----------------------------+--------+
| Action                     |  10310 |
| ChangeGroup                |  11050 |
| ChangeItem                 |  11320 |
| ColumnLayout               |  10040 |
| ColumnLayoutItem           |  10120 |
| Component                  |  10110 |
| ConfigurationContext       |  10170 |
| SchemeIssueSecurities      |  10040 |
...
```

Actually, Ofbiz allocates IDs in batches of 10, so the SEQ_ID is the next available ID rounded up to the nearest 10. So you might have:

```
mysql> select max(ID) from jiraaction;
+---------+
| max(ID) |
+---------+
|   10303 |
+---------+
1 row in set (0.04 sec)

mysql> select * from SEQUENCE_VALUE_ITEM where SEQ_NAME='Action';
+----------+--------+
| SEQ_NAME | SEQ_ID |
+----------+--------+
| Action   |  10310 |
+----------+--------+
1 row in set (0.01 sec)
```

Where 10310 is the nearest 10 above 10303.

The `SEQ_NAME` column refers to the database table name defined in `WEB-INF/classes/entitydefs/entitymodel.xml` (eg. "Action" is `jiraaction`).

Manually inserting records

The implication of this is that if you want to manually insert records into JIRA database tables, you **must update SEQUENCE_VALUE_ITEM yourself**. Set the relevant rows' SEQ_ID values to a value greater than the actual maximum ID in the table. You will then need to restart JIRA to ensure all database caches are reset.

Retrieving Change History using JIRA's API

The best way to retrieve change history entries is:

```
actionManager.getChangeHistory(getIssue(), authenticationContext.getUser());
```

You can declare dependency on JiraAuthenticationContext and ActionManager in the constructor of your plugin as described in PicoContainer and JIRA.

The getChangeHistory method returns `ChangeHistory` objects on which you can call the getChangeItems() method. This returns a List if

GenericValue objects each one representing an issue field update. To check the field that was updated do:

String fieldName = changeItem.getString("field")

GenericValues are described in Database Schema.

Configuration properties

There are some global JIRA configuration settings, such as:

- Search index path
- Attachments path
- Base URL
- Settings configured in Administration -> General Configuration
- License info

These are all stored in a propertyset on a virtual 'jira.properties' entity:

```
mysql> select * from propertyentry where ENTITY_NAME='jira.properties';
+-------+-----------------+-----------+-------------------------------------+--------------+
| ID    | ENTITY_NAME     | ENTITY_ID | PROPERTY_KEY                        | propertytype |
+-------+-----------------+-----------+-------------------------------------+--------------+
| 10001 | jira.properties |         1 | webwork.i18n.encoding               |            5 |
| 10000 | jira.properties |         1 | jira.i18n.language.index            |            5 |
| 10002 | jira.properties |         1 | jira.title                          |            5 |
| 10008 | jira.properties |         1 | jira.option.allowattachments        |            1 |
| 10003 | jira.properties |         1 | jira.baseurl                        |            5 |
| 10005 | jira.properties |         1 | jira.path.index                     |            5 |
| 10006 | jira.properties |         1 | jira.option.indexing                |            1 |
| 10007 | jira.properties |         1 | jira.path.attachments               |            5 |
| 10004 | jira.properties |         1 | jira.mode                           |            5 |
| 10011 | jira.properties |         1 | jira.path.backup                    |            5 |
| 10012 | jira.properties |         1 | License Message                     |            5 |
| 10013 | jira.properties |         1 | License Hash 1                      |            5 |
| 10018 | jira.properties |         1 | jira.option.user.externalmanagement |            1 |
| 10019 | jira.properties |         1 | jira.option.voting                  |            1 |
| 10016 | jira.properties |         1 | jira.setup                          |            5 |
| 10022 | jira.properties |         1 | jira.version.patched                |            5 |
| 10017 | jira.properties |         1 | jira.option.allowunassigned         |            1 |
| 10020 | jira.properties |         1 | jira.option.watching                |            1 |
| 10021 | jira.properties |         1 | jira.option.issuelinking            |            1 |
| 10023 | jira.properties |         1 | jira.option.cache.issues            |            1 |
| 10024 | jira.properties |         1 | jira.issue.desc.environment         |            5 |
| 10025 | jira.properties |         1 | jira.issue.desc.timetrack           |            5 |
| 10027 | jira.properties |         1 | jira.timetracking.hours.per.day     |            5 |
| 10028 | jira.properties |         1 | jira.issue.desc.original.timetrack  |            5 |
| 10050 | jira.properties |         1 | jira.option.allowsubtasks           |            1 |
| 10080 | jira.properties |         1 | jira.option.allowthumbnails         |            1 |
| 10101 | jira.properties |         1 | jira.constant.default.resolution    |            5 |
| 10026 | jira.properties |         1 | jira.timetracking.days.per.week     |            5 |
| 10100 | jira.properties |         1 | jira.scheme.default.issue.type      |            5 |
| 10120 | jira.properties |         1 | jira.option.emailvisible            |            5 |
| 10150 | jira.properties |         1 | jira.sid.key                        |            5 |
| 10161 | jira.properties |         1 | jira.trackback.exclude.pattern      |            5 |
| 10151 | jira.properties |         1 | jira.lf.edit.version                |            5 |
| 10160 | jira.properties |         1 | jira.comment.level.visibility.groups |            1 |
+-------+-----------------+-----------+-------------------------------------+--------------+
34 rows in set (0.02 sec)
```

The 'propertytype' column indicates which table stores the actual value for this property. 1 means 'propertynumber' and 5 means 'propertystring':

```
mysql> select * from propertyentry pe, propertynumber ps where pe.id=ps.id and
pe.ENTITY_NAME='jira.properties' and propertytype='1' ;
+-------+-----------------+-----------+-------------------------------------+-------------+-------+----
ID    | ENTITY_NAME     | ENTITY_ID | PROPERTY_KEY                        | propertytype | ID    |
propertyvalue |
+-------+-----------------+-----------+-------------------------------------+-------------+-------+----
10006 | jira.properties |         1 | jira.option.indexing                |            1 | 10006 |
1 |
| 10008 | jira.properties |         1 | jira.option.allowattachments        |            1 | 10008 |
1 |
| 10018 | jira.properties |         1 | jira.option.user.externalmanagement |            1 | 10018 |
0 |
| 10017 | jira.properties |         1 | jira.option.allowunassigned         |            1 | 10017 |
0 |
| 10019 | jira.properties |         1 | jira.option.voting                  |            1 | 10019 |
1 |
| 10020 | jira.properties |         1 | jira.option.watching                |            1 | 10020 |
1 |
| 10021 | jira.properties |         1 | jira.option.issuelinking            |            1 | 10021 |
1 |
| 10023 | jira.properties |         1 | jira.option.cache.issues            |            1 | 10023 |
0 |
| 10050 | jira.properties |         1 | jira.option.allowsubtasks           |            1 | 10050 |
1 |
| 10080 | jira.properties |         1 | jira.option.allowthumbnails         |            1 | 10080 |
1 |
| 10160 | jira.properties |         1 | jira.comment.level.visibility.groups |            1 | 10160 |
1 |
+-------+-----------------+-----------+-------------------------------------+-------------+-------+----
rows in set (0.01 sec)
```

Here you can see that indexing is enabled (1), external user management off (0), subtasks enabled (1), etc.

```
mysql> select PROPERTY_KEY, propertyvalue
         from propertyentry pe, propertystring ps
         where pe.id=ps.id and pe.ENTITY_NAME='jira.properties' and propertytype='5';
+-------+-----------------+-----------+--------------------------+-------------+-------+---------------
ID    | ENTITY_NAME     | ENTITY_ID | PROPERTY_KEY             | propertytype | ID    | propertyvalue
|
+-------+-----------------+-----------+--------------------------+-------------+-------+---------------
10004 | jira.properties |         1 | jira.mode                |            5 | 10004 | public
|
| 10000 | jira.properties |         1 | jira.i18n.language.index |            5 | 10000 | english
|
| 10001 | jira.properties |         1 | webwork.i18n.encoding    |            5 | 10001 | UTF-8
|
| 10002 | jira.properties |         1 | jira.title               |            5 | 10002 | Your Company
JIRA                                |
| 10003 | jira.properties |         1 | jira.baseurl             |            5 | 10003 |
http://localhost:8080/jira                        |
| 10005 | jira.properties |         1 | jira.path.index          |            5 | 10005 |
/home/jturner/jira/cleandb/ent/3.7.2/index       |
| 10007 | jira.properties |         1 | jira.path.attachments    |            5 | 10007 |
/home/jturner/jira/cleandb/ent/3.7.2/attachments |
| 10011 | jira.properties |         1 | jira.path.backup         |            5 | 10011 |
/home/jturner/jirabackups
...
| 10022 | jira.properties |         1 | jira.version.patched             |            5 | 10022 |
186
...
mysql>
```

JIRA build versions and upgrading

One important property stored is the build number:

```
mysql> select * from propertyentry pe, propertystring ps where pe.id=ps.id and
pe.ENTITY_NAME='jira.properties' and pe.PROPERTY_KEY='jira.version.patched';
+-------+-----------------+-----------+----------------------+--------------+-------+---------------+
| ID    | ENTITY_NAME     | ENTITY_ID | PROPERTY_KEY         | propertytype | ID    | propertyvalue |
+-------+-----------------+-----------+----------------------+--------------+-------+---------------+
| 10022 | jira.properties |         1 | jira.version.patched |            5 | 10022 | 186           |
+-------+-----------------+-----------+----------------------+--------------+-------+---------------+
1 row in set (0.00 sec)
```

The build number corresponds to a JIRA version. In the footer of JIRA pages you'll see this build number after the version, eg. "Version: 3.7.2-#186".

The build number is mostly relevant when upgrading. JIRA will run "upgrade tasks" when it detects that the JIRA's data is from an older version, and it does this by comparing the build number in the database with its own.

When an upgrade task is successfully run, it is recorded in the upgradehistory table:

```
mysql> select * from upgradehistory;
+-------+-----------------------------------------------------------------------+
| ID    | UPGRADECLASS                                                          |
+-------+-----------------------------------------------------------------------+
| 10000 | com.atlassian.jira.upgrade.tasks.UpgradeTask1_2                       |
| 10001 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build10                  |
| 10080 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build100                 |
| 10081 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build101                 |
| 10082 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build102                 |
| 10085 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build105                 |
| 10002 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build11                  |
| 10083 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build103                 |
| 10084 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build104                 |
| 10101 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build133                 |
....
| 10074 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build96                  |
| 10010 | com.atlassian.jira.upgrade.tasks.enterprise.UpgradeTask_Build47       |
| 10040 | com.atlassian.jira.upgrade.tasks.enterprise.UpgradeTask_Build69       |
| 10071 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build93                  |
| 10073 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build95                  |
| 10075 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build98                  |
| 10061 | com.atlassian.jira.upgrade.tasks.enterprise.UpgradeTask_Build84       |
| 10065 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build89                  |
| 10140 | com.atlassian.jira.upgrade.tasks.UpgradeTask_Build186                 |
+-------+-----------------------------------------------------------------------+
52 rows in set (0.02 sec)
```

Custom fields

Custom fields defined in the system are stored in the `customfield` table, and instances of custom fields are stored in `customfieldvalue`:

```
mysql> desc customfieldvalue;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ID          | decimal(18,0) | NO   | PRI |         |       |
| ISSUE       | decimal(18,0) | YES  | MUL | NULL    |       |
| CUSTOMFIELD | decimal(18,0) | YES  |     | NULL    |       |
| PARENTKEY   | varchar(255)  | YES  |     | NULL    |       |
| STRINGVALUE | varchar(255)  | YES  |     | NULL    |       |
| NUMBERVALUE | decimal(18,6) | YES  |     | NULL    |       |
| TEXTVALUE   | longtext      | YES  |     | NULL    |       |
| DATEVALUE   | datetime      | YES  |     | NULL    |       |
| VALUETYPE   | varchar(255)  | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
```

We can print all custom field values for an issue with:

```
mysql> select * from customfieldvalue where issue=(select id from jiraissue where pkey='JRA-5448');
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
ID    | ISSUE | CUSTOMFIELD | PARENTKEY | STRINGVALUE | NUMBERVALUE | TEXTVALUE | DATEVALUE
| VALUETYPE |
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
23276 | 22160 |       10190 | NULL      | NULL        |        NULL | NULL      | 2004-12-07 17:25:58
| NULL      |
+-------+-------+-------------+-----------+-------------+-------------+-----------+---------------------+
```

and we can see what type of custom field this (10190) is with:

```
mysql> select * from customfield where id=10190;
+-------+-----------------------------------------------+-----------------------------------------------
ID    | CUSTOMFIELDTYPEKEY                              | CUSTOMFIELDSEARCHERKEY
| cfname          | DESCRIPTION | defaultvalue | FIELDTYPE | PROJECT | ISSUETYPE |
+-------+-----------------------------------------------+-----------------------------------------------
10190 | com.atlassian.jira.ext.charting:resolutiondate |
com.atlassian.jira.ext.charting:resolutiondatesearcher | Resolution Date | NULL        | NULL
|      NULL |    NULL | NULL      |
+-------+-----------------------------------------------+-----------------------------------------------
```

(ie. it's a "Resolution Date").

This query identifies a particular custom field value in a particular issue:

```
mysql> select stringvalue from customfieldvalue where customfield=(select id from customfield where
cfname='Urgency') and issue=(select id from jiraissue where pkey='FOR-845');
+-------------+
| stringvalue |
+-------------+
| Low         |
+-------------+
1 row in set (0.33 sec)
```

If the custom field has multiple values (multi-select or multi-user picker), each issue can have multiple `customfieldvalue` rows:

```
mysql> select * from customfieldvalue where customfield=(select ID from customfield where
cfname='MultiUser');
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
ID    | ISSUE | CUSTOMFIELD | PARENTKEY | STRINGVALUE | NUMBERVALUE | TEXTVALUE | DATEVALUE |
VALUETYPE |
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
10002 | 10060 |       10000 | NULL      | bob         |        NULL | NULL      | NULL      | NULL
|
| 10003 | 10060 |       10000 | NULL      | jeff        |        NULL | NULL      | NULL      | NULL
|
+-------+-------+-------------+-----------+-------------+-------------+-----------+-----------+----------
rows in set (0.00 sec)
```

Here issue 10060 has two users, `bob` and `jeff` in its MultiUser custom field.

Custom field configuration options

The option sets (1, 2, 3 and A, B, C) are stored in the **customfieldoption** table:

mysql> select * from customfieldoption where customfieldconfig=10031;

ID	CUSTOMFIELD	CUSTOMFIELDCONFIG	PARENTOPTIONID	SEQUENCE	customvalue	optiontype
10000	10001	10031	NULL	0	1	NULL
10001	10001	10031	NULL	1	2	NULL

10002	10001	10031	NULL	2	3	NULL

mysql> select * from customfieldoption where customfieldconfig=10032;

ID	CUSTOMFIELD	CUSTOMFIELDCONFIG	PARENTOPTIONID	SEQUENCE	customvalue	optiontype
10003	10001	10032	NULL	0	A	NULL
10004	10001	10032	NULL	1	B	NULL
10005	10001	10032	NULL	2	C	NULL

Custom field configuration default value

The custom field default value is stored in the **genericconfiguration** table. Since this table must store a value for any custom field type (cascading selects, multi-selects, etc) the value is encoded as XML.

If we were to set a default value of "2" for our "Default Configuration Scheme for SelectCF", it would be recorded as:

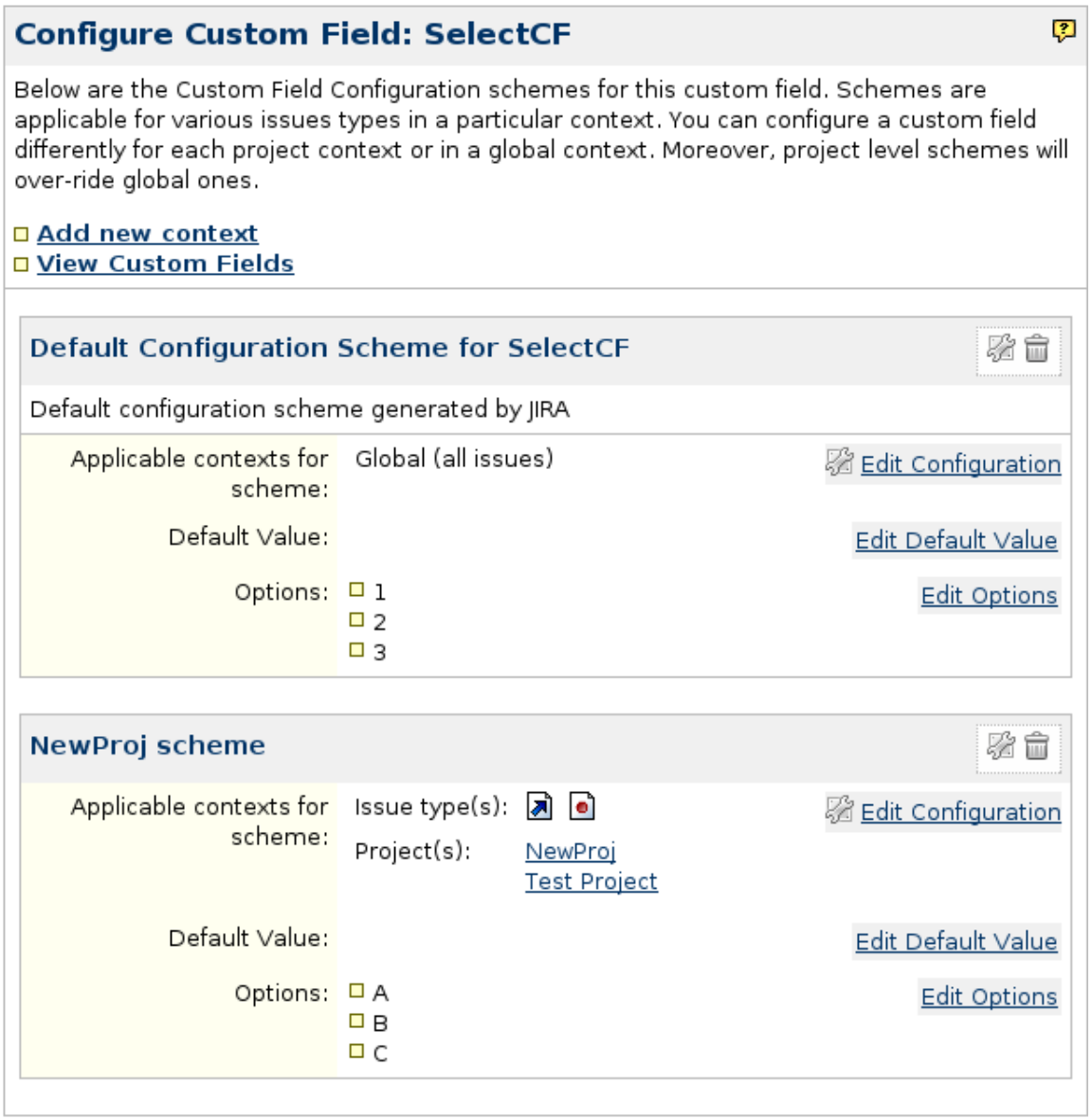
mysql> select * from genericconfiguration where ID=10031;

ID	DATATYPE	DATAKEY	XMLVALUE
10031	DefaultValue	10030	<string>2</string>

Custom field configuration schemes

JIRA custom fields can have different default values and possible values for each project and/or issue type. This is set up by clicking 'Configure' in the custom field definition.

For instance, in this screenshot the "SelectCF" select-list field will have values 1, 2, 3 for all projects except bugs and improvements in "NewProj" and"Test Project", which will have values A, B and C:



Custom field configuration scopes

In the database, these custom field configuration schemes are stored in the **fieldconfigscheme** table

mysql> select * from fieldconfigscheme where id in (10031,10032);

ID	configname	DESCRIPTION	FIELDID	CUSTOMFIELD
10031	Default Configuration Scheme for SelectCF	Default configuration scheme generated by JIRA	customfield_10001	NULL
10032	NewProj scheme		customfield_10001	NULL

The **projects in scope** for each of these schemes is listed as records (one per project) in the **configurationcontext** table:

mysql> select * from configurationcontext where fieldconfigscheme=10031;

ID	PROJECTCATEGORY	PROJECT	customfield	FIELDCONFIGSCHEME
10053	NULL	NULL	customfield_10001	10031

(Here showing that that the "Default Configuration Scheme for SelectCF" applies to all projects)

mysql> select * from configurationcontext where fieldconfigscheme=10032;

ID	PROJECTCATEGORY	PROJECT	customfield	FIELDCONFIGSCHEME
10054	NULL	10000	customfield_10001	10032
10055	NULL	10010	customfield_10001	10032

(Here showing that "NewProj scheme" is restricted to projects with ids 10000 and 10010 ("Test Project" and "NewProj")).

Finally, the **issue types in scope** for each scheme is listed as records (one per issue type) in the **fieldconfigschemeissuetype** table:

mysql> select * from fieldconfigschemeissuetype where fieldconfigscheme = 10031;

ID	ISSUETYPE	FIELDCONFIGSCHEME	FIELDCONFIGURATION
10051	NULL	10031	10031

(Here showing that "Default Configuration Scheme for SelectCF" is not limited to any issue types)

mysql> select * from fieldconfigschemeissuetype where fieldconfigscheme = 10032;

ID	ISSUETYPE	FIELDCONFIGSCHEME	FIELDCONFIGURATION
10052	1	10032	10032
10053	4	10032	10032

(Here showing that "Newproj scheme" is limited to issue types with IDs 1 and 4).

Note that there should *always be a record* in `configurationcontext` and `fieldconfigschemeissuetype` for each issue type configuration scheme. If the scheme isn't restricted to any projects or issue types, the `project` and `issuetype` columns of the respective tables should be `NULL`. Incidentally JIRA has/had a bug where it didn't leave an entry when deleting an issue type (JRA-10461), so if you are making changes manually, don't make the same mistake.

Example SQL queries for JIRA

This is a page that lists example SQL queries that some JIRA users might find useful, mainly for reporting purposes.

Fixed by and Cascading Field Value

If you have a JIRA installation that uses a Cascading Select List custom field to track the version/build that the issue has been fixed in, and would like to find issues that were fixed by a particular user in a particular version/build, you can use the SQL query below. Please note that the SQL does not filter out reopened issues, but returns issues that were resolved at least once. Due to this, duplicates are also possible in the generated result set.

```
SELECT jiraissue.*
FROM jiraissue,
OS_HISTORYSTEP,
customfieldvalue,
customfieldoption
WHERE OS_HISTORYSTEP.ENTRY_ID = jiraissue.id
AND OS_HISTORYSTEP.ACTION_ID = <action_id>
AND OS_HISTORYSTEP.CALLER = <user_name>
AND customfieldvalue.issue = jiraissue.id
AND customfieldvalue.PARENTKEY = <parent_key>
AND customfieldvalue.stringvalue = customfieldoption.id
AND customfieldoption.customvalue like '<cf_value>';
```

Where

- <user_name> - the username of the desired user
- <action_id> - the id of your transition into the fixed state (may need multiple)
- <parent_key> - the id of the Level 1 option in customfieldoption - E.g. 10040
- <cf_value> - the Level 2 value of the cascading field. E.g 'realease%'

Find Fixed For versions for an issue

If you want to find out the Versions an Issue has been marked "Fix For" you can run the following query

```
SELECT projectversion.id, vname
FROM projectversion,
nodeassociation,
jiraissue
WHERE ASSOCIATION_TYPE = 'IssueFixVersion'
AND SINK_NODE_ID = projectversion.id
AND SOURCE_NODE_ID = jiraissue.id
AND pkey = '<issue_key>';
```

Where

- <issue_key> - the key of an issue. E.g. TEST-10

Find all issues changed by a user after a certain date

If you want to find out all the issues that a particular user has changed use the following query

```
SELECT DISTINCT(j.id) FROM jiraissue j, changegroup g
WHERE j.id = g.issueid
AND g.author = '<user name>'
AND g.created > '<date>';
```

Where

- <date> - the earliest desired date (The date should be in the format 'yyyy-mm-dd hh:mm:s'. E.g '2005-10-06 14:40:28')
- <username> - the name of the desired user

Find Statuses of all issues in a project on a given date

You can use this SQL to retreive the status of all issues on a give date in a give project: Note. This was tested under MySQL

```
SELECT JI.pkey, STEP.STEP_ID
FROM (SELECT STEP_ID, ENTRY_ID
      FROM OS_CURRENTSTEP
      WHERE OS_CURRENTSTEP.START_DATE < '<your date>'
UNION SELECT STEP_ID, ENTRY_ID
      FROM OS_HISTORYSTEP
      WHERE OS_HISTORYSTEP.START_DATE < '<your date>'
      AND OS_HISTORYSTEP.FINISH_DATE > '<your date>' ) As STEP,
(SELECT changeitem.OLDVALUE AS VAL, changegroup.ISSUEID AS ISSID
      FROM changegroup, changeitem
      WHERE changeitem.FIELD = 'Workflow'
      AND changeitem.GROUPID = changegroup.ID
UNION SELECT jiraissue.WORKFLOW_ID AS VAL, jiraissue.id as ISSID
      FROM jiraissue) As VALID,
jiraissue as JI
WHERE STEP.ENTRY_ID = VALID.VAL
AND VALID.ISSID = JI.id
AND JI.project = <proj_id>;
```

Where

- <your date> is the date you want to check
- <proj_id> is the project you want to check

Find Status counts for a Project on a given date

Or you can find out the counts on specific date: Note. This was tested under MySQL

```
SELECT count(*), STEP.STEP_ID
FROM (SELECT STEP_ID, ENTRY_ID
      FROM OS_CURRENTSTEP
      WHERE OS_CURRENTSTEP.START_DATE < '<your date>'
UNION SELECT STEP_ID, ENTRY_ID
      FROM OS_HISTORYSTEP
      WHERE OS_HISTORYSTEP.START_DATE < '<your date>'
      AND OS_HISTORYSTEP.FINISH_DATE > '<your date>' ) As STEP,
(SELECT changeitem.OLDVALUE AS VAL, changegroup.ISSUEID AS ISSID
      FROM changegroup, changeitem
      WHERE changeitem.FIELD = 'Workflow'
      AND changeitem.GROUPID = changegroup.ID
UNION SELECT jiraissue.WORKFLOW_ID AS VAL, jiraissue.id as ISSID
      FROM jiraissue) As VALID,
jiraissue as JI
WHERE STEP.ENTRY_ID = VALID.VAL
AND VALID.ISSID = JI.id
AND JI.project = <proj_id>
Group By STEP.STEP_ID;
```

Where

- <your date> is the date you want to check
- <proj_id> is the project you want to check

Find how Many Issue Moved into States for a given Period

Use this SQL to find out how many issues were Created, Resolved, ..., Closed during a given period. Note that if an issue moves through more than 1 transition, it will be counted more than once. Note. This was tested under MySQL

```
SELECT NEWSTRING AS Status, count(*) AS Number
FROM changeitem, changegroup, jiraissue
WHERE changeitem.field = 'Status'
    AND changeitem.groupid = changegroup.id
    AND changegroup.issueid = jiraissue.id
    AND jiraissue.project = <project_id>
    AND changegroup.CREATED >= '<date_from>'
    AND changegroup.CREATED < '<date_to>'
Group By NEWSTRING
UNION
SELECT 'Created' As Status, count(*) AS Number
FROM jiraissue
WHERE jiraissue.CREATED >= '<date_from>'
    AND jiraissue.CREATED < '<date_to>'
    AND jiraissue.project = <project_id>;
```

Where

- <date_from> is the date you want to check from
- <date_to> is the date you want to check to
- <project_id> is the project you want to check

Get Components for an Issue

Get all the Components for an Issue

```
SELECT jiraissue.pkey, component.cname
FROM nodeassociation, component, jiraissue
WHERE component.ID = nodeassociation.SINK_NODE_ID
AND jiraissue.id = nodeassociation.SOURCE_NODE_ID
AND nodeassociation.ASSOCIATION_TYPE = 'IssueComponent'
AND pkey = '<issue_key>';
```

Find date that Closed issues were closed

Find out the date an issue was Closed for all currnetly closed issues.

```
SELECT pKey, OS_CURRENTSTEP.STATUS, OS_CURRENTSTEP.START_DATE
FROM jiraissue, OS_CURRENTSTEP
WHERE issuestatus = 6 AND OS_CURRENTSTEP.ENTRY_ID = jiraissue.WORKFLOW_ID;
```

Simple join - jiraissue and jiraaction

```
SELECT *
FROM jiraissue LEFT JOIN jiraaction ON jiraissue.id = jiraaction.issueid;
```

Simple join - jiraissue and changegroup

```
SELECT *
FROM jiraissue LEFT JOIN changegroup ON jiraissue.id = changegroup.issueid;
```

Simple join - Changegroup and changeitem

```
SELECT *
FROM changegroup LEFT JOIN changeitem ON changegroup.id = changeitem.groupid;
```

Simple join - jiraissue and os_currentstep

```
SELECT *
FROM jiraissue LEFT JOIN OS_CURRENTSTEP ON jiraissue.WORKFLOW_ID = OS_CURRENTSTEP.ENTRY_ID;
```

Simple join - jiraissue and os_historystep

```
SELECT *
FROM jiraissue LEFT JOIN OS_HISTORYSTEP ON jiraissue.WORKFLOW_ID = OS_HISTORYSTEP.ENTRY_ID;
```

Issue fields

This page shows how to examine each of a JIRA issue's fields via SQL. We will use JRA-3166 as a sample issue in our queries.

Simple fields

Most fields in JIRA are kept in the `jiraissue` table:

```
mysql> desc jiraissue;
+----------------------+---------------+------+-----+---------+-------+
| Field                | Type          | Null | Key | Default | Extra |
+----------------------+---------------+------+-----+---------+-------+
| ID                   | decimal(18,0) | NO   | PRI |         |       |
| pkey                 | varchar(255)  | YES  | MUL | NULL    |       |
| PROJECT              | decimal(18,0) | YES  | MUL | NULL    |       |
| REPORTER             | varchar(255)  | YES  |     | NULL    |       |
| ASSIGNEE             | varchar(255)  | YES  | MUL | NULL    |       |
| issuetype            | varchar(255)  | YES  |     | NULL    |       |
| SUMMARY              | varchar(255)  | YES  |     | NULL    |       |
| DESCRIPTION          | longtext      | YES  |     | NULL    |       |
| ENVIRONMENT          | longtext      | YES  |     | NULL    |       |
| PRIORITY             | varchar(255)  | YES  |     | NULL    |       |
| RESOLUTION           | varchar(255)  | YES  |     | NULL    |       |
| issuestatus          | varchar(255)  | YES  |     | NULL    |       |
| CREATED              | datetime      | YES  |     | NULL    |       |
| UPDATED              | datetime      | YES  |     | NULL    |       |
| DUEDATE              | datetime      | YES  |     | NULL    |       |
| VOTES                | decimal(18,0) | YES  |     | NULL    |       |
| TIMEORIGINALESTIMATE | decimal(18,0) | YES  |     | NULL    |       |
| TIMEESTIMATE         | decimal(18,0) | YES  |     | NULL    |       |
| TIMESPENT            | decimal(18,0) | YES  |     | NULL    |       |
| WORKFLOW_ID          | decimal(18,0) | YES  |     | NULL    |       |
| SECURITY             | decimal(18,0) | YES  |     | NULL    |       |
| FIXFOR               | decimal(18,0) | YES  |     | NULL    |       |
| COMPONENT            | decimal(18,0) | YES  |     | NULL    |       |
+----------------------+---------------+------+-----+---------+-------+
```

They can be retrieved with a regular select:

```
mysql> select id, pkey, project, reporter, assignee, issuetype, summary from jiraissue where
pkey='JRA-3166';
+-------+----------+---------+-----------+----------+-----------+---------------------------------+
| id    | pkey     | project | reporter  | assignee | issuetype | summary                         |
+-------+----------+---------+-----------+----------+-----------+---------------------------------+
| 16550 | JRA-3166 |   10240 | mvleeuwen | NULL     | 2         | Database consistency check tool |
+-------+----------+---------+-----------+----------+-----------+---------------------------------+
```

User details

Say we wish to find out the email address and other details about our reporter, `mvleeuwen`. First we find this user's ID:

```
mysql> select id from userbase where username='mvleeuwen';
+-------+
| id    |
+-------+
| 13841 |
+-------+
```

Then use it to look up 'properties' of this userbase record (stored in propertysets. Each property has a record in the `propertyentry` table specifying its name and type, and a record in one of `propertystring`, `propertydecimal`, `propertydate`, `propertytext`, `propertydata` or `propertynumber`, depending on the type.

```
mysql> desc propertyentry;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| ID           | decimal(18,0) | NO   | PRI |         |       |
| ENTITY_NAME  | varchar(255)  | YES  |     | NULL    |       |
| ENTITY_ID    | decimal(18,0) | YES  | MUL | NULL    |       |
| PROPERTY_KEY | varchar(255)  | YES  |     | NULL    |       |
| propertytype | decimal(9,0)  | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+

mysql> select * from propertyentry where ENTITY_NAME='OSUser' and ENTITY_ID=(select id from userbase
where username='mvleeuwen');
+-------+-------------+-----------+--------------+--------------+
| ID    | ENTITY_NAME | ENTITY_ID | PROPERTY_KEY | propertytype |
+-------+-------------+-----------+--------------+--------------+
| 18352 | OSUser      |     13841 | email        |            5 |
| 18353 | OSUser      |     13841 | fullName     |            5 |
+-------+-------------+-----------+--------------+--------------+
```

So `email` and `fullName` are of type 5. which means the `propertystring` table. Here is the list of `propertytype` to table mappings:

propertyentry.propertytype value	Table value is stored in	Used for
1	propertynumber	Boolean values, eg. user preferences
5	propertystring	Most fields, eg. full names, email addresses
6	propertytext	Large blocks of text, eg. the introduction text, HTML portletconfigurations
2/3	propertydecimal	Unused in JIRA
7	propertydate	Unused in JIRA
10	propertydata	Unused in JIRA

So the `email` and `fullName` properties are strings, and so can be found in the `propertystring` table:

```
mysql> select * from propertystring where id in (18352, 18353);
+-------+---------------------+
| ID    | propertyvalue       |
+-------+---------------------+
| 18352 | lemval@zonnet.nl    |
| 18353 | Michael van Leeuwen |
+-------+---------------------+
```

Components and versions

Since each issue can have multiple components/versions, there is a join table between `jiraissue` and `version`/`component` tables called `nodeassociation`:

```
mysql> desc nodeassociation;
+--------------------+---------------+------+-----+---------+-------+
| Field              | Type          | Null | Key | Default | Extra |
+--------------------+---------------+------+-----+---------+-------+
| SOURCE_NODE_ID     | decimal(18,0) | NO   | PRI |         |       |
| SOURCE_NODE_ENTITY | varchar(60)   | NO   | PRI |         |       |
| SINK_NODE_ID       | decimal(18,0) | NO   | PRI |         |       |
| SINK_NODE_ENTITY   | varchar(60)   | NO   | PRI |         |       |
| ASSOCIATION_TYPE   | varchar(60)   | NO   | PRI |         |       |
| SEQUENCE           | decimal(9,0)  | YES  |     | NULL    |       |
+--------------------+---------------+------+-----+---------+-------+

mysql> select distinct SOURCE_NODE_ENTITY from nodeassociation;
+--------------------+
| SOURCE_NODE_ENTITY |
+--------------------+
| Issue              |
| Project            |
+--------------------+

mysql> select distinct SINK_NODE_ENTITY from nodeassociation;
+-----------------------+
| SINK_NODE_ENTITY      |
+-----------------------+
| IssueSecurityScheme   |
| PermissionScheme      |
| IssueTypeScreenScheme |
| NotificationScheme    |
| ProjectCategory       |
| FieldLayoutScheme     |
| Component             |
| Version               |
+-----------------------+

mysql> select distinct ASSOCIATION_TYPE from nodeassociation;
+------------------+
| ASSOCIATION_TYPE |
+------------------+
| IssueVersion     |
| IssueFixVersion  |
| IssueComponent   |
| ProjectScheme    |
| ProjectCategory  |
+------------------+
```

So to get fix-for versions of an issue, run:

```
mysql> select * from projectversion where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueFixVersion' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
| ID    | PROJECT | vname | DESCRIPTION | SEQUENCE | RELEASED | ARCHIVED | URL  | RELEASEDATE |
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
| 11614 |   10240 | 3.6   | NULL        |      131 | NULL     | NULL     | NULL | NULL        |
+-------+---------+-------+-------------+----------+----------+----------+------+-------------+
```

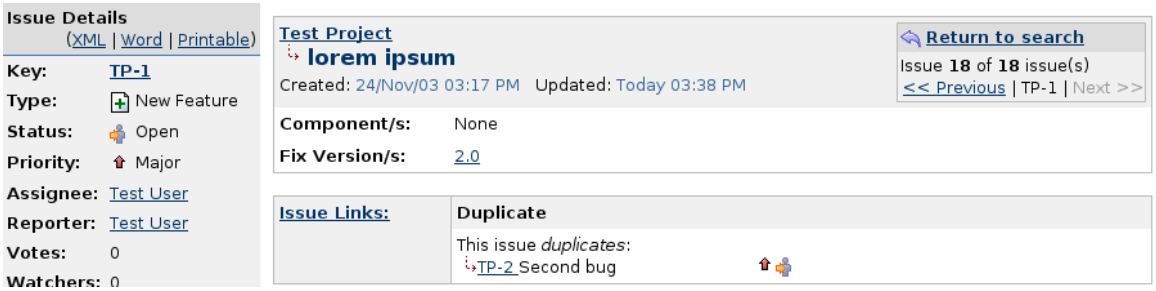
Similarly with affects versions:

```
mysql> select * from projectversion where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueVersion' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
ID    | PROJECT | vname              | DESCRIPTION | SEQUENCE | RELEASED | ARCHIVED | URL  |
RELEASEDATE         |
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
10931 |   10240 |  3.0.3 Professional | NULL        |       73 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
| 10930 |   10240 |  3.0.3 Standard     | NULL        |       72 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
| 10932 |   10240 |  3.0.3 Enterprise   | NULL        |       74 | true     | NULL     | NULL |
2004-11-19 00:00:00 |
+-------+---------+--------------------+-------------+----------+----------+----------+------+----------
```

and components:

```
mysql> select * from component where id in (
    select SINK_NODE_ID from nodeassociation where ASSOCIATION_TYPE='IssueComponent' and
SOURCE_NODE_ID=(
        select id from jiraissue where pkey='JRA-5351')
    );
+-------+---------+---------------+-------------+------+------+--------------+
| ID    | PROJECT | cname         | description | URL  | LEAD | ASSIGNEETYPE |
+-------+---------+---------------+-------------+------+------+--------------+
| 10126 |   10240 | Web interface | NULL        | NULL | NULL |         NULL |
+-------+---------+---------------+-------------+------+------+--------------+
```

Issue links



JIRA issue links are stored in the `issuelink` table, which simply links the IDs of two issues together, and records the link type:

```
mysql> desc issuelink;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ID          | decimal(18,0) | NO   | PRI |         |       |
| LINKTYPE    | decimal(18,0) | YES  | MUL | NULL    |       |
| SOURCE      | decimal(18,0) | YES  | MUL | NULL    |       |
| DESTINATION | decimal(18,0) | YES  | MUL | NULL    |       |
| SEQUENCE    | decimal(18,0) | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

For instance, to list all links between TP-1 and TP-2:

```
mysql> select * from issuelink where SOURCE=(select id from jiraissue where pkey='TP-1') and
DESTINATION=(select id from jiraissue where pkey='TP-2');
+-------+----------+--------+-------------+----------+
| ID    | LINKTYPE | SOURCE | DESTINATION | SEQUENCE |
+-------+----------+--------+-------------+----------+
| 10020 |    10000 |  10000 |       10010 |     NULL |
+-------+----------+--------+-------------+----------+
1 row in set (0.00 sec)
```

Link types are defined in `issuelinktype`. This query prints all links in the system with their type:

```
mysql> select j1.pkey, issuelinktype.INWARD, j2.pkey from jiraissue j1, issuelink, issuelinktype,
jiraissue j2 where j1.id=issuelink.SOURCE and j2.id=issuelink.DESTINATION and
issuelinktype.id=issuelink.linktype;
+-------+---------------------+-------+
| pkey  | INWARD              | pkey  |
+-------+---------------------+-------+
| TP-4  | jira_subtask_inward | TP-5  |
| TP-4  | jira_subtask_inward | TP-7  |
| TP-4  | jira_subtask_inward | TP-8  |
| TP-11 | jira_subtask_inward | TP-12 |
| TP-4  | jira_subtask_inward | TP-6  |
| TP-1  | is duplicated by    | TP-2  |
+-------+---------------------+-------+
6 rows in set (0.00 sec)
```

Subtasks

As shown in the last query, JIRA records the issue-subtask relation as a link. The "subtask" link type is hidden in the user interface (indicated by the 'pstyle' value below), but visible in the database:

```
mysql> select * from issuelinktype;
+-------+------------------+---------------------+----------------------+--------------+
| ID    | LINKNAME         | INWARD              | OUTWARD              | pstyle       |
+-------+------------------+---------------------+----------------------+--------------+
| 10000 | Duplicate        | is duplicated by    | duplicates           | NULL         |
| 10001 | jira_subtask_link | jira_subtask_inward | jira_subtask_outward | jira_subtask |
+-------+------------------+---------------------+----------------------+--------------+
2 rows in set (0.00 sec)
```

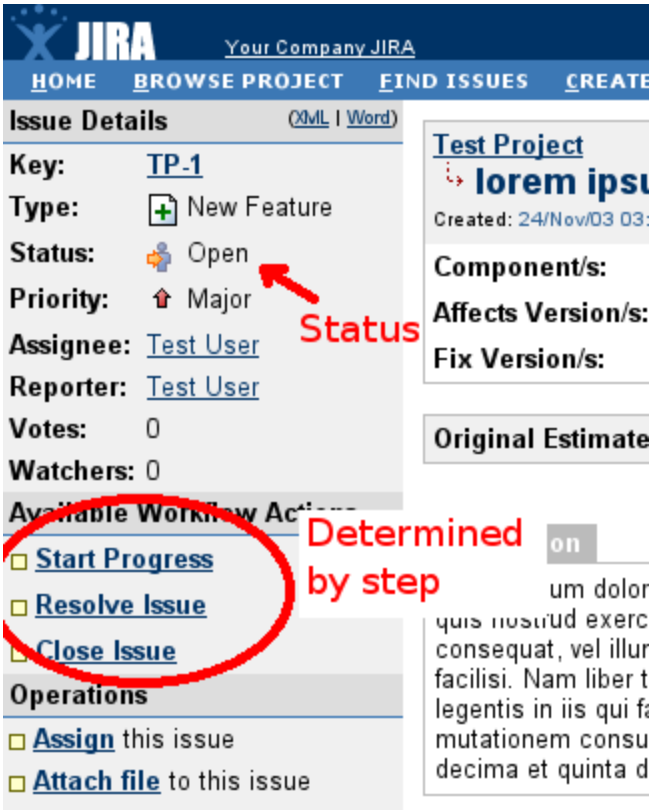
This means it is possible to convert an issue to a subtask, or vice-versa, by tweaking `issuelink` records.

Custom fields have their own set of tables. For details, see Custom fields

Issue status and workflow

This page describes the database tables involved in issue workflow. It will be useful for people who wish to insert issues into the database manually, or diagnose/fix corrupted databases.
JIRA issues have both:

- a **status** (Open, Closed, In Progress etc).
- a **workflow step**, which governs which transitions are available



Issue status

In the database, the **status** (Open, Closed etc) is stored on the `jiraissue` table:

```
mysql> select issuestatus from jiraissue where pkey='TP-1';
+-------------+
| issuestatus |
+-------------+
| 1           |
+-------------+
1 row in set (0.00 sec)

mysql> select pname from issuestatus, jiraissue where issuestatus.id=jiraissue.issuestatus and
pkey='TP-1';
+-------+
| pname |
+-------+
| Open  |
+-------+
1 row in set (0.00 sec)
```

Issue workflow step

Originally JIRA issues only had a status. Then in version 2.0, *workflow* was added, so that transitions between statuses could be customized. An issue's workflow step is stored in new tables, referenced from `jiraissue` by the `workflow_id`:

```
mysql> select * from OS_WFENTRY where ID=(select workflow_id from jiraissue where pkey='TP-1');
+-------+------+-------------+-------+
| ID    | NAME | INITIALIZED | STATE |
+-------+------+-------------+-------+
| 10000 | jira | 0           | 1     |
+-------+------+-------------+-------+
1 row in set (0.02 sec)
```

The TP-1 issue's OS_WFENTRY row indicates that the issue uses the 'jira' (default, built-in) workflow.

```
mysql> select * from OS_CURRENTSTEP where ENTRY_ID=(select workflow_id from jiraissue where
pkey='TP-1');
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
ID    | ENTRY_ID | STEP_ID | ACTION_ID | OWNER | START_DATE          | DUE_DATE | FINISH_DATE | STATUS
| CALLER |
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
10000 | 10000    | 1       | 0         |       | 2003-11-24 15:17:50 |          |             | Open
|        |
+-------+----------+---------+-----------+-------+---------------------+----------+-------------+--------
row in set (0.13 sec)
```

The issue's OS_CURRENTSTEP row specifies the issue's *current step*. The only field really used is `STEP_ID`. This references a step definition in the workflow:

```
<step id="1" name="Open">
```

The workflow definition for the built-in 'jira' workflow can be seen in `atlassian-jira/WEB-INF/classes/jira-workflow.xml`

How status and step relate

An issue's status and workflow step are kept in synch:

```
mysql> select issuestatus.pname status, issuestatus, OS_CURRENTSTEP.STEP_ID, OS_CURRENTSTEP.STATUS
from
   issuestatus, jiraissue, OS_CURRENTSTEP where
   issuestatus.id=jiraissue.issuestatus and jiraissue.workflow_id=OS_CURRENTSTEP.ENTRY_ID;
+-------------+-------------+---------+----------+
| status      | issuestatus | STEP_ID | STATUS   |
+-------------+-------------+---------+----------+
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
...
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| Open        | 1           | 1       | Open     |
| In Progress | 3           | 3       | Underway |
| Closed      | 6           | 6       | Closed   |
+-------------+-------------+---------+----------+
32 rows in set (0.00 sec)

mysql>
```

Status and step are kept in synch is with a workflow post-function (UpdateIssueStatusFunction), which updates the status whenever the step changes.

If the step gets out of synch with the status, then incorrect (or no) workflow operations appear on the issue page. Eg. if OS_CURRENTSTEP.STEP_ID was 6 ("Closed") when jiraissue.issuestatus was 1 ("Open"), then the issue would have only one transition ("Reopen issue") which would break if anyone clicked on it.

Summary

- For each `jiraissue` row, there is a `OS_CURRENTSTEP` and `OS_WFENTRY` row.
- `OS_WFENTRY` specifies the applicable workflow. `OS_CURRENTSTEP` specifies the step in that workflow.
- The relations are:
  - jiraissue.WORKFLOW_ID == OS_WFENTRY.ID
  - jiraissue.WORKFLOW_ID == OS_CURRENTSTEP.ENTRY_ID

User and Group Tables

User and Group Tables

User Tables

JIRA uses an open source framework called OSUser to manage its users. The user records are stored in the `userbase` database table. The table holds little information:

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| ID            | decimal(18,0) |      | PRI | 0       |       |
| username      | varchar(255)  | YES  | MUL | NULL    |       |
| PASSWORD_HASH | varchar(255)  | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

OSUser also stores some properties for the user. The properties are: *full name* and *e-mail*. These properties are stored in the `propertyentry` and `propertystring` tables.

For each user the `propertyentry` table:

```
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| ID           | decimal(18,0) |      | PRI | 0       |       |
| ENTITY_NAME  | varchar(255)  | YES  | MUL | NULL    |       |
| ENTITY_ID    | decimal(18,0) | YES  |     | NULL    |       |
| PROPERTY_KEY | varchar(255)  | YES  |     | NULL    |       |
| propertytype | decimal(9,0)  | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
```

stores 'OSUser' in the ENTITY_NAME column, the id of the `userbase` record in the ENTITY_ID column, and 'fullName' or 'email' in the PROPERTY_KEY column.

The `propertystring` table:

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| ID            | decimal(18,0) |      | PRI | 0       |       |
| propertyvalue | text          | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

stores the actual values of the properties. The ID column is the same as the ID of the `propertyentry` record and the propertyvalue column would store the full name or e-mail of the user.

Group Tables

The groups are stored in the `groupbase` table:

```
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| ID        | decimal(18,0) |      | PRI | 0       |       |
| groupname | varchar(255)  | YES  | MUL | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
```

Group Membership

The `membershipbase` table records which users belong to which groups:

```
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| ID         | decimal(18,0) |      | PRI | 0       |       |
| USER_NAME  | varchar(255)  | YES  | MUL | NULL    |       |
| GROUP_NAME | varchar(255)  | YES  | MUL | NULL    |       |
+------------+---------------+------+-----+---------+-------+
```

The USER_NAME column is set to the username column in the `userbase` table. The GROUP_NAME is set to the groupname record in the `groupbase` table.

Watches and Votes

Watches and votes are recorded in the userassociation table:

```
+------------------+---------------+------+-----+---------+-------+
| Field            | Type          | Null | Key | Default | Extra |
+------------------+---------------+------+-----+---------+-------+
| SOURCE_NAME      | varchar(60)   | NO   | PRI |         |       |
| SINK_NODE_ID     | decimal(18,0) | NO   | PRI |         |       |
| SINK_NODE_ENTITY | varchar(60)   | NO   | PRI |         |       |
| ASSOCIATION_TYPE | varchar(60)   | NO   | PRI |         |       |
| SEQUENCE         | decimal(9,0)  | YES  |     | NULL    |       |
+------------------+---------------+------+-----+---------+-------+
```

For example:

```
mysql> select * from userassociation;
+---------------+--------------+------------------+------------------+----------+
| SOURCE_NAME   | SINK_NODE_ID | SINK_NODE_ENTITY | ASSOCIATION_TYPE | SEQUENCE |
+---------------+--------------+------------------+------------------+----------+
| asmith        |     108433   | Issue            | WatchIssue       |     NULL |
| droberts      |     100915   | Issue            | WatchIssue       |     NULL |
| dfernandez    |     106387   | Issue            | VoteIssue        |     NULL |
...
```

For example, here user 'asmith' is watching issue with id 108433.

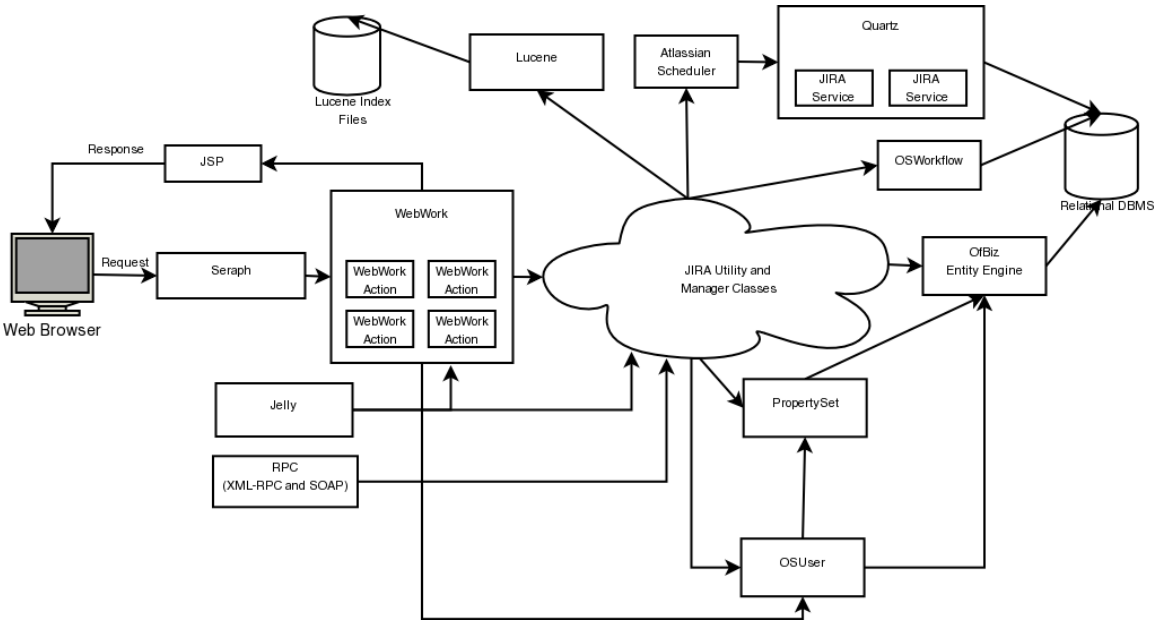
JIRA Architectural Overview

JIRA Overview

This page provides a very high level overview of JIRA's dependencies and the role each one plays in JIRA. This page makes references to external resources (websites, books) where one can find more information.

Technical Introduction to JIRA

JIRA is a web application written in Java. It is deployed as a standard Java WAR file into a java Servlet Container such as Tomcat.



WebWork

As JIRA is a web application, users interact with JIRA using a web browser. JIRA uses OpenSymphony's WebWork 1 to process web requests submitted by users. Please note that WebWork 1, **not** 2, is used. WebWork 1 is a MVC framework similar to Struts. Each request is handled by a WebWork action which usually uses other objects, such as utility and Manager classes to accomplish a task.

JIRA uses JSP for the View layer. So most of HTML that is served to the user as the response to their web request is generated by a JSP. Therefore, to generate a response the WebWork action uses a JSP.

Also see JIRA Webwork Actions.

For more information on WebWork 1 please see its online documentation.

Seraph

Almost all authentication in JIRA is performed through Seraph, Atlasian's open source web authentication framework. The goal of seraph is to provide a simple, extensible authentication system that we can use on any application server.

Seraph is implemented as a servlet filter. Its sole job is, given a web request, to associate that request with a particular user. It supports several methods of authentication, including HTTP Basic Authentication, form-based authentication (ie. redirect to an internal or external login form), and looking up credentials already stored in the user's session (e.g. a cookie set by a SSO system).

Seraph performs no user management itself. It merely checks the credentials of the incoming request, and delegates any user-management functions (looking up a user, checking a user's password is correct) to JIRA's user-management - OSUser (discussed later in this document).

If you were looking to integrate JIRA with a Single Sign-On (SSO) solution, you would do so by writing a custom Seraph authenticator (and in fact, many customers have done so). Please note that by default JIRA is not shipped with any SSO integration, customers have to write a custom Authenticator themselves.

Another **very important** function that Seraph performs in JIRA is to only allow users with Global Admin permission to access WebWork actions that allow the user to perform administration tasks. These WebWork actions are accessed by URLs starting with "/admin". For more information on JIRA's permission please see JIRA's documentation.

For more information on how seraph works internally please see this page.

OSUser

OSUser is OpenSymphony's user and group management framework.

OSUser provides the following functionality:

1. Stores users and groups in JIRA's database
2. Stores group membership (which users are part of which groups) in JIRA's db
3. Stores user preferences (e.g. whether a user would like to receive HTML or Text e-mails)
4. Authenticates users (checks if the users password matches)
5. Provides API that allows to manage (create, delete) users, manage group membership (add and remove users from groups), and manage users' preferences.

As mentioned previously, Seraph delegates to OSUser to authenticate the user (i.e. check whether the correct password has been entered when a user tries to login).

OSUser uses the following tables in the database:

1. `userbase` - stores one record for each JIRA user
2. `groupbase` - stores one record for each JIRA group
3. `membershipbase` - stores which users are members of which group

Note that `userbase` table only stores the numeric (arbitrary) id of the user, username, and a hashed password.

OSUser uses PropertySet (covered in the next section) to store user's e-mail address, and full name. PropertySet is also used to store user preferences. In JIRA the preferences include things like:

- whether the user would like to receive HTML or Text e-mail
- number of issues to display in JIRA's Issue Navigator
- whether to receive notifications for user's own updates to issues
- Locale (Language) of the user

OSUser also provides simple authentication against LDAP, which JIRA can utilise. Note, that **only** LDAP authentication is supported; the user records must still exist in JIRA's database. For more information on JIRA's LDAP autentication please see this document.

For more information on OSUSer, please see its documentation.

Note that we are hoping to migrate JIRA to Polis for user management so that full LDAP user and group management is supported. Confluence has already done so.

PropertySet

OpenSymphony's PropertySet is a framework that can store a set of properties (key/value pairs) against a particular "entity" with a unique id. An "entity" can be anything one wishes. For exmaple, OSUser uses PropertySet to store user's e-mail address, full name and preferences. Therefore, in case of OSUSer, the "entity" is a User.

Each property has a key (which is always a java.lang.String) and a value, which can be:

1. java.lang.String
2. java.lang.Long
3. java.util.Date
4. java.lang.Double

Each property is always associated with **one** entity. As far as PropertySet is concerned an "entity" has an entity name, and a numeric id. As long as the same entity name/id combination is used to store the value and retrieve the value, everything will work.

In JIRA PropertySet uses the following database tables:

1. `propertyentry` - records the entity name and id for a property, its key, and the data type of the property's value. Each record in this table also has a unique id.
2. `propertystring` - records String values
3. `propertydecimal` - records Double values
4. `propertydate` - records Date values
5. `propertynumber` - records Long values

Each of the records in property<type> tables also has an id column. The id is the same as the id of the propertyentry record for this property. As the property's key and value are split across 2 tables, to retrieve a property value, a join needs to be done, between propertyentry table and one of the property<type> tables. Which property<type> table to join with is determined by the value of the `propertytype` column in the propertyentry record.

Here is an example of a full name stored for a user:
(to do)

PropertySet is used in JIRA:

1. By the OSUser framework to store users e-mail, fullname, and preferences
2. To store Application Properties, which are configurable settings that a user can change to customise their installation of JIRA. For more information on Application Properties please see JIRA's documentation
3. To store chosen preferences of Portlets on user's Dashboards.

For more information on PropertySet please see its documentation. Also see JIRA Database Schema.

JIRA Utility and Manager Classes

A lot of business logic in JIRA is implemented in 100s of java classes. The classes can be simple utility classes or Manager Objects.

Manager Objects in JIRA usually have one specific goal (or topic). For example

`com.atlassian.jira.project.version.VersionManager` is used to work with project versions, i.e. create, update, delete and retrieve versions.

Manager objects use a lot of external dependencies, most of which are open source, but some are developed by Atlassian and are usually shared between Atlassian products. The external dependencies developed by Atlassian are described in this list of product dependencies.

JIRA Webwork Actions

Webwork Actions and actions.xml

JIRA uses the Webwork (1.x) framework to define MVC mappings. This has been superseded by Webwork 2 which is used by Confluence, Bamboo and Crowd. The best source of documentation is: http://opensymphony.com/webwork_old/src/docs/manual/ and http://wiki.opensymphony.com/display/WW1/Home

The mappings are declared in the actions.xml file (`src/webapp/WEB-INF/classes/actions.xml`).

Note the parts of the action mapping.

Commands are optional, you use them when several interactions belong to the same Action. A command name is specified on the url like this:

```
SomeAction!myCommand.jspa
```

The command is implemented in a method in the Action class with the corresponding name:

```
public String doMyCommand() {

    // implement the command logic here

    return "someview";
}
```

The doExecute method is run when no command is requested i.e. the bare `/path/to/MyAction.jspa`.

Security

When adding an action to the actions.xml You must ensure the appropriate `roles-required` value is specified. This will ensure only users in the authorised role can execute the action. For actions that can be handled by application level security, such as those actions that can be given to project administrators identified through the course of administering JIRA, no role may be required, likewise setup actions and others that manage their own permissions. For some actions, the **use** role must be present. This ensures the user is logged in and identified.

The **admin** role is required for all administration actions so you must be sure when adding an action that your new action has **roles-required="admin"** or confident that it doesn't need it. The **sysadmin** role requires the user be a system administrator, and the **use** role requires that they just be logged in. (Other definitions can be found in `Permissions.java` in the source).

Actions don't care about the path of the URI, just the ActionName.jspa and optionally the !commandName suffix.

Webwork Plugins

JIRA can have Actions defined in plugins.

PicoContainer and JIRA

Picocontainer

JIRA uses Picocontainer as a central object factory. Picocontainer is responsible for instantiating objects and resolving their constructor dependencies. This greatly simplifies code, in that any Picocontainer-instantiated object (eg. a Webwork action) can obtain an instance of another (eg. a Manager class) simply by requesting one in its constructor. PicoContainer will ensure each object required in the constructor is passed in (aka dependency injection). Eg. the ViewIssue action:

ViewIssue.java

```
public class ViewIssue extends AbstractViewIssue
{
    ....
    public ViewIssue(RepositoryManager repositoryManager, PermissionManager permissionManager,
TrackbackManager trackbackManager,
                     ThumbnailManager thumbnailManager, SubTaskManager subTaskManager,
IssueLinkManager issueLinkManager,
                     IssueLinkTypeManager issueLinkTypeManager, VoteManager voteManager,
WatcherManager watcherManager,
                     PluginManager pluginManager)
    {
        super(issueLinkManager, subTaskManager);
        this.trackbackManager = trackbackManager;
        this.thumbnailManager = thumbnailManager;
        this.issueLinkTypeManager = issueLinkTypeManager;
        this.pluginManager = pluginManager;
        this.pagerManager = new PagerManager(ActionContext.getSession());
        this.repositoryManager = repositoryManager;
        this.permissionManager = permissionManager;
        this.voteManager = voteManager;
        this.watcherManager = watcherManager;
    }
    ....
}
```

Non-managed classes

Classes not managed by Picocontainer (eg. workflow conditions / functions, Services and Listeners, or JSP scriptlets) can still get pico-instantiated objects statically using static methods of ManagerFactory or ComponentManager.getInstance(). For example:

```
ComponentManager.getInstance().getProjectManager();
ComponentManager.getInstance().getIssueFactory();
ManagerFactory.getCustomFieldManager();
//or
ManagerFactory.getApplicationProperties();
```

Using ComponentManager.getInstance() is prefered as ManagerFactory will become obsolete. However, if a method that you are after does not exist on the ComponentManager, use ManagerFactory.

Register new Picocontainer-managed classes

Picocontainer-managed classes need to be registered with Picocontainer. This happens automatically for Webwork actions, but other classes need to be registered manually. This is done in ComponentManager's registerComponents() method:

ComponentManager.java

```
    private void registerComponents()
    {
         ....
        internalContainer.registerComponentInstance(CoreFactory.getActionDispatcher());
        internalContainer.registerComponentInstance(CoreFactory.getGenericDelegator());
        internalContainer.registerComponentInstance(CoreFactory.getAssociationManager());
        internalContainer.registerComponentInstance(MailFactory.getServerManager());
        internalContainer.registerComponentInstance(UserManager.getInstance());
        internalContainer.registerComponentInstance(SearchProviderFactory.getDefaultProvider());
        .....
        internalContainer.registerComponentImplementation(IssueUpdater.class,
DefaultIssueUpdater.class);
        internalContainer.registerComponentImplementation(IssueEventDispatcher.class);
        internalContainer.registerComponentImplementation(ChangeLogUtils.class);
        internalContainer.registerComponentImplementation(QueryCreator.class);
        internalContainer.registerComponentImplementation(ReleaseNoteManager.class);
        internalContainer.registerComponentImplementation(ApplicationProperties.class,
ApplicationPropertiesImpl.class);
        internalContainer.registerComponentImplementation(VelocityManager.class,
JiraVelocityManager.class);
        internalContainer.registerComponentImplementation(ConstantsManager.class,
DefaultConstantsManager.class);
        internalContainer.registerComponentImplementation(PermissionSchemeManager.class,
DefaultPermissionSchemeManager.class);
        internalContainer.registerComponentImplementation(CacheManager.class,
DefaultCacheManager.class);
        internalContainer.registerComponentImplementation(TrackbackManager.class,
TrackbackManagerImpl.class);
        internalContainer.registerComponentImplementation(WatcherManager.class,
DefaultWatcherManager.class);
        internalContainer.registerComponentImplementation(EntityUtils.class);
        internalContainer.registerComponentImplementation(AttachmentManager.class,
DefaultAttachmentManager.class);
        internalContainer.registerComponentImplementation(FieldManager.class,
DefaultFieldManager.class);
      .....
     }
```

Non-managed classes

Classes not managed by Picocontainer (eg. workflow conditions / functions, Services and Listeners, or JSP scriptlets) can still get pico-instantiated objects statically using static methods of ManagerFactory or ComponentManager.getInstance(). For example:

```
ComponentManager.getInstance().getProjectManager();
ComponentManager.getInstance().getIssueFactory();
ManagerFactory.getCustomFieldManager();
//or
ManagerFactory.getApplicationProperties();
```

Using ComponentManager.getInstance() is prefered as ManagerFactory will become obsolete. However, if a method that you are after does not exist on the ComponentManager, use ManagerFactory.

Sample Code

- Creating and Editing an Issue
- How to retrieve an issue with an ID or an IssueKey
- How to search for Issues based on CustomField
- How to search for Issues from within a Plugin
- Retrieving issue's links
- Working with Custom Fields

Creating and Editing an Issue

This documentation is in-progress. But this is enough to get started.

Creating a new Issue

There are different ways of getting the MutableIssue object depending on which version of JIRA you are using.

If you are using **JIRA 3.6.5 and below** then you will call:

```
MutableIssue issueObject = (MutableIssue) JiraUtils.loadComponent(IssueImpl.class);
```

If you are using **JIRA 3.7 and above** then you will call:

```
MutableIssue issueObject = issueFactory.getIssue();
```

As mentioned below you can get a handle on an IssueFactory either by Constructor Injection or explicitly via a call like:

```
ComponentManager.getInstance().getIssueFactory()
```

Here is sniplet of code that creates a new issue:

```
// Use this if you are using JIRA 3.6.5 and below
//MutableIssue issueObject = (MutableIssue) JiraUtils.loadComponent(IssueImpl.class);

// Use this if you are using JIRA 3.7 and above
MutableIssue issueObject = issueFactory.getIssue();

// Regular Fields
issueObject.setProject(projectManager.getProject(new Long(10000)));
issueObject.setIssueType(constantsManager.getIssueType("1"));
issueObject.setSummary("Test Issue");
issueObject.setReporter(UserUtils.getUser("admin"));
issueObject.setAssignee(UserUtils.getUser("admin"));
issueObject.setPriority(constantsManager.getPriority("1"));
issueObject.setDescription("Test description");
issueObject.setAffectedVersions(EasyList.build(versionManager.getVersion(new Long(10000)),
versionManager.getVersion(new Long(10001))));
issueObject.setFixVersions(EasyList.build(versionManager.getVersion(new Long(10002))));
issueObject.setComponents(EasyList.build(projectManager.getComponent(new Long(10000)),
projectManager.getComponent(new Long(10001))));

// Custom Fields
CustomField customField = customFieldManager.getCustomFieldObject(new Long(10020));
issueObject.setCustomFieldValue(customField, "Test Value");

Map params = new HashMap();
params.put("issue", issueObject);
GenericValue issue = issueManager.createIssue(authenticationContext.getUser(), params);
```

This code example uses a lot of Manager objects. You can get a reference to them by declaring a dependency in the constructor of your plugin.

Classes not managed by Picocontainer (eg. workflow conditions / functions, Services and Listeners, or JSP scriptlets) can still get pico-instantiated objects statically using static methods of ManagerFactory or ComponentManager.getInstance(). For example:

```
ComponentManager.getInstance().getProjectManager();
ComponentManager.getInstance().getIssueFactory();
ManagerFactory.getCustomFieldManager();
//or
ManagerFactory.getApplicationProperties();
```

Using ComponentManager.getInstance() is prefered as ManagerFactory will become obsolete. However, if a method that you are after does not exist on the ComponentManager, use ManagerFactory.

The code above also sets a value for a custom field on the issue. Please note that the value must be an object of the type that the Custom Field expects. As the above code was using a Text custom field, a simple `java.lang.String` is fine. For more information on working with custom fields please see Working with Custom Fields.

Editing an existing Issue

The code below edits a Due Date field of an issue and sets it to 24 hours from now. A comment that is visible by everyone who has permission to see the issue is also added.

```
MutableIssue issue = issueFactory.getIssue(issueGV);
issue.setDueDate(new Timestamp(System.currentTimeMillis() + 24*60*60*1000));
Map actionParams = EasyMap.build("issue", issue.getGenericValue(), "issueObject", issue, "remoteUser",
authenticationContext.getUser());
actionParams.put("comment", "Test Comment");
actionParams.put("commentLevel", null);
ActionResult aResult = CoreFactory.getActionDispatcher().execute(ActionNames.ISSUE_UPDATE,
actionParams);
```

You can also specify a group name as the `commentLevel` parameter to restrict the visibility of comments. If you use the above code to update

an issue all the relevant change history entries will be made, and an Updated Issue Event generated.

The code above created an issue object from an issue `GenericValue`.

You can retrieve the issue generic value using the following code:

```
// By Id
GenericValue issueGV1 = issueManager.getIssue(new Long(10000));
// By Issue Key
GenericValue issueGV2 = issueManager.getIssue("TST-1");
```

To learn how to update custom fields please see Working with Custom Fields.

Events

It's worth noting that in the examples above, when the issues are created or modified events are fired, and any notifications that are associated with those events will be triggered.

How to retrieve an issue with an ID or an IssueKey

You can retrieve the issue generic value using the following code:

```
// By Id
GenericValue issueGV1 = issueManager.getIssue(new Long(10000));
// By Issue Key
GenericValue issueGV2 = issueManager.getIssue("TST-1");
```

How to search for Issues based on CustomField

This example is tested in JIRA 3.6.5. It searches issues which match the search term of a given CustomField.

```
User you = ((JiraAuthenticationContext)
ComponentManager.getComponentInstanceOfType(JiraAuthenticationContext.class)).getUser();
SearchProvider sProvider = (SearchProvider)
ComponentManager.getComponentInstanceOfType(SearchProvider.class);
SearchRequest sRequest = new SearchRequest(you);

CustomFieldManager cfm = ComponentManager.getInstance().getCustomFieldManager();
CustomField cf = cfm.getCustomFieldObject("customfield_10021");
IssueSearcher searcher = cf.getCustomFieldSearcher();
FieldValuesHolder fvh = new FieldValuesHolderImpl();
CustomFieldParams cfp = new CustomFieldParamsImpl();
List list = new ArrayList();
list.add("hello");
cfp.put(null, list);
cfp.setCustomField(cf);
fvh.put("customfield_10021", cfp);
searcher.populateSearchRequest(sRequest, fvh);

List searched = sProvider.search(sRequest, you, PagerFilter.getUnlimitedFilter()).getIssues();
```

See also: How to search for Issues from within a Plugin

How to search for Issues from within a Plugin

When developing plugins it is often desired to find issues that meet a certain criteria. Here is a code sample that constructs a `SearchRequest` and loops through all unresolved issues in project with id of 10000 and prints out their summary:

```
public class MyPlugin extends ...
{
    private final SearchProvider searchProvider;
    private final JiraAuthenticationContext authenticationContext;

    public MyPlugin(SearchProvider searchProvider, JiraAuthenticationContext authenticationContext)
    {
        this.searchProvider = searchProvider;
        this.authenticationContext = authenticationContext;
    }

    public void findIssues()
    {
        try
        {
            SearchRequest sr = new SearchRequest(authenticationContext.getUser());
            sr.addParameter(new ProjectParameter(new Long(10000)));
            sr.addParameter(new ResolutionParameter("-1"));
            SearchResults searchResults = searchProvider.search(sr, authenticationContext.getUser(),
PagerFilter.getUnlimitedFilter());
            List issues = searchResults.getIssues();
            for (Iterator iterator = issues.iterator(); iterator.hasNext();)
            {
                Issue issue = (Issue) iterator.next();
                System.out.println("issue = " + issue.getSummary());
            }
        }
        catch (SearchException e)
        {
            e.printStackTrace();
        }
    }
}
```

The issues returned in the SearchResults are `Issue` objects and you can check out the linked javadocs for information on available methods.

Please note that the class declares its dependencies in the constructor (`SearchProvider` and `JiraAuthenticationContext`). When MyPlugin is instantiated it will be given the dependencies by JIRA. The process is explained in more detail here.

Note, that using a `SearchRequest` it is possible to construct the same type of queries as using the Issue Navigator. It is possible to add as many parameters to the search request as required. The summary of available parameters can be found in JIRA's API documentation.

Examples of using other Search Parameters

To search for all `unresolved` issues in a project with id `10000` that are assigned to the user with username `"admin"`, create the search request as follows:

```
SearchRequest sr = new SearchRequest(getRemoteUser());
sr.addParameter(new ProjectParameter(new Long(10000)));
sr.addParameter(new ResolutionParameter("-1"));
sr.addParameter(new UserParameter(DocumentConstants.ISSUE_ASSIGNEE, "admin"));
```

Examples of populating CustomFieldParams into SearchRequest

```
SearchRequest sRequest = new SearchRequest(you);

CustomFieldManager cfm = ComponentManager.getInstance().getCustomFieldManager();
CustomField cf = cfm.getCustomFieldObject("customfield_10021");
IssueSearcher searcher = cf.getCustomFieldSearcher();
FieldValuesHolder fvh = new FieldValuesHolderImpl();
CustomFieldParams cfp = new CustomFieldParamsImpl();
List list = new ArrayList();
list.add("hello");
cfp.put(null, list);
cfp.setCustomField(cf);
fvh.put("customfield_10021", cfp);
searcher.populateSearchRequest(sRequest, fvh);
```

Figuring out what Search Parameters you need from the JIRA source

You can get a better understanding of how searching works in JIRA by looking at the source code. Each SearchableField in JIRA has associated searchers. The IssueSearchers are responsible for all of the field's searching functionality like taking the search criteria from the user and adding it to the SearchRequest.

For examples of which parameters a particular searcher uses look at the `populateSearchRequest` method of the searcher. For example look at the class FixForVersionsSearcher for more information on what SearchParameters you should be for fix for versions (you should use FixForParameter).

For custom fields, you choose an associated searcher on creation. You can find which searchers are available for a given CustomFieldType in the config file `system-customfieldtypes-plugin.xml` where all the system custom field types and custom field searchers are specified.

For example, we can see that the searcher for multicheckboxes is the checkboxsearcher (This uses `MultiSelectSearcher`). Examining the class MultiSelectSearcher, we can see that it returns the search parameter:

```
new GenericMultiValueParameter(field.getId(), cleanedValues)
```

This is the thing you need to add to your search request. "cleanedValues" is a List of values you want to search on.

Retrieving issue's links

The following code sample iterates over a list of issues and for each issue retrieves its linked issues. This code can be useful if you are creating a custom report that shows linked issues.

Remember that each link has a direction and a type. Therefore the issues in the `Link Collection` are grouped by link type and direction.

```
// A list of GenericValues representing issues
List issues = ...
for (Iterator iterator = issues.iterator(); iterator.hasNext();)
{
    GenericValue issue = (GenericValue) iterator.next();
    // Retrieve a collection of all linked issues and their link types
LinkCollection linkCollection = getIssueLinkManager().getLinkCollection(issue,
authenticationContext.getUser());
    Set linkTypes = linkCollection.getLinkTypes();
    if (linkTypes != null)
    {
        // For each link type
for (Iterator iterator1 = linkTypes.iterator(); iterator1.hasNext();)
        {
            IssueLinkType linkType = (IssueLinkType) iterator1.next();
            // Get the outward linked issues
List outwardIssues = linkCollection.getOutwardIssues(linkType.getName());
            if (outwardIssues != null)
            {
                for (Iterator iterator2 = outwardIssues.iterator(); iterator2.hasNext();)
                {
                    GenericValue outwardIssue = (GenericValue) iterator2.next();
                    System.out.println("outwardIssue = " + outwardIssue);
                }
            }
            // And the inward linked issues
List inwardIssues = linkCollection.getInwardIssues(linkType.getName());
            if (inwardIssues != null)
            {
                for (Iterator iterator2 = inwardIssues.iterator(); iterator2.hasNext();)
                {
                    GenericValue inwardIssue = (GenericValue) iterator2.next();
                    System.out.println("inwardIssue = " + inwardIssue);
                }
            }
        }
    }
}
```

One way to retrieve a list of issues is to make and run a Search Request.

Please note that the code above uses `JiraAuthenticationContext` to retrieve the remote user. The easiest way to get access to an instance of the JiraAuthenticationContext is to declare it as a dependency in the constructor of your class. The process is explained in more detail here.

Working with Custom Fields

It is fairly straight forward to programmatically read or update a value of a custom field in JIRA.

Here is a code snapshot that reads a value and then updates it:

```
CustomFieldManager customFieldManager =
ComponentManager.getInstance().getFieldManager().getCustomFieldManager();
CustomField customField = customFieldManager.getCustomFieldObject(new Long(10040));

// Obtain the FieldLayoutItem associated with the custom field for this issue
FieldLayoutItem fieldLayoutItem =
ComponentManager.getInstance().getFieldLayoutManager().getFieldLayout(issue).getFieldLayoutItem(customFie
// Read a value
Object value = customField.getValue(issue);
System.out.println("Custom Field Value: " + value);

// Create a modified value object with the old value and the new value
ModifiedValue modifiedValue = new ModifiedValue(value, "Option 2");

// Update the value
customField.updateValue(fieldLayoutItem, issue, modifiedValue, new DefaultIssueChangeHolder());
// Show updated value
System.out.println("Custom Field Value: " + customField.getValue(issue));
```

Please note the type of the `value` object depends on the custom field in question. For text and select list custom fields this is a simple String (as you can see above). For a Multi-Select Custom Field it is a `java.util.List` of Strings.

Note that the CustomField object was retrieved using its id, which is unique for each custom field. The easiest way to determine the id of a custom field is to navigate to the View Custom Fields page in the Administration section and hover over one of the "Operation" links next to the custom field. The id will be one of the URL parameters in the link.

Here is another example which reads the values of a Cascading Select List custom field:

```
CustomFieldManager customFieldManager = ManagerFactory.getCustomFieldManager();
CustomField customField = customFieldManager.getCustomFieldObject(new Long(10010));
CustomFieldParams cfParams = (CustomFieldParams) customField.getValue(issue);

// Get the 1st level value
Collection value = (List) cfParams.getValuesForKey(null);
Option option = (Option) value.iterator().next();
System.out.println("1st level value = " + option.getValue());

// Get the 2nd level value
value = (List) cfParams.getValuesForKey("1");
option = (Option) value.iterator().next();
System.out.println("2nd level value = " + option.getValue());
```

Developer Tutorials

Developer Tutorials

- Jelly Examples
- JIRA Custom Installer Guide
- JIRA Plugin Tutorials
  - Available Permissions
  - How to create a JIRA Portlet
  - How to create a JIRA Report
  - How to create a new Custom Field Type
  - How to create Custom Workflow Elements for JIRA 3
- JIRA RPC Services — This page provides a starting point for all your remote procedure call needs
  - Creating a SOAP Client — This document contains notes on how to perform various operations (e.g. creating issues) via a Java SOAP client (JIRA 3.1 or above required).
    - Remote API (SOAP) Examples
  - Creating a XML-RPC Client — In this tutorial, we write a basic XML-RPC client (using Apache XML-RPC) that logs in, retrieves projects and then log out again. A Python client is also demonstrated.
    - Remote API (XML-RPC) Examples
  - Enabling the RPC plugin
  - JIRA XML-RPC Overview
  - Frequently Asked RPC Questions and Known Issues
- Modifying JIRA Templates and JSPs
  - Adding Custom Fields to Email
  - Adding Custom Field to Issue Summary
  - Adding JavaScript to all pages E.g. Google Analytics
  - Contents of the Velocity Context
  - Customising interface based on user's role
  - Customising JIRA Excel Output
  - Customizing text
  - Mapping custom events to new email templates
  - Tutorial - How to Create a Custom Release Notes Template Containing Release Comments
  - Velocity Context for Email Templates
- Services, Listeners and Handlers
  - Adding your own email handling classes — In JIRA, emails may be periodically fetched (eg. via POP or IMAP) and processed (eg. create a new issue, create a comment). You can add new ways of processing emails by implementing the MessageHandler interface in JIRA (or subclassing one if its implementations).
- 3.6 Technical Improvements
  - Programmatically determining automatic assignee
  - Resolutions per workflow step
  - Subtask creation form enhancements

Jelly Examples

Jelly is a scripting and templating language from Apache's Jakarta project.

We use it within JIRA to import and manipulate data.

This page contains example scripts highlighting the more advanced capabilities of Jelly.

Retrieving a list of all users

Jelly can actually create and invoke methods on Java objects. This script utilizes this to retrieve a list of all the users in JIRA:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:core="jelly:core">

<core:invokeStatic className="com.opensymphony.user.UserManager"method="getInstance" var="instance"/>
<core:invoke on="${instance}" method="getUsers" var="users"/>

<core:forEach var="user" items="${users}">
//do something with ${user}
</core:forEach>
</JiraJelly>
```

Retrieving all users from a group

Like the above script, this script creates an instance of the UserManager. It then retrieves all the users from a group.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:j="jelly:core"
xmlns:util="jelly:util">
<j:invokeStatic className="com.opensymphony.user.UserManager" method="getInstance" var="instance"/>
<j:invoke on="${instance}" method="getGroup" var="grp">
<j:arg type="java.lang.String" value="thegroupname"/>
</j:invoke>
<j:invoke on="${grp}" method="getUsers" var="users"/>
</JiraJelly>
```

Timeout an email to a new state with auto-response

See Jelly Escalation For Support.

JIRA Custom Installer Guide

This page lists instructions for how to build a custom installer for JIRA Standalone.

> **JIRA Version**
>
> The installer was added in JIRA v3.8. It currently only supports Microsoft Windows.

Requirements

The JIRA installer is built using install4j (v4.0.5) for which you will need to buy a valid license. If you would like to sign your installer executables you will also need a valid code signing certificate. If you are building the installer on Linux, you may also require mono and mono-mcs for code-signing purposes.

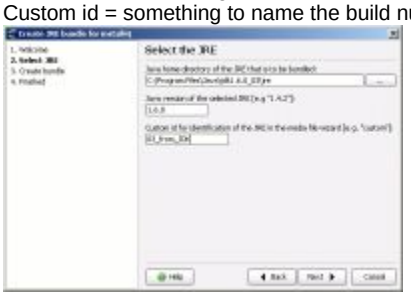
General

The installer is configured using install4j. To edit the installer, open jira-template.install4j in the install4j GUI.

Installed JRE

Install4j will bundle the installer with one of the JREs defined in the <INSTALL4J_HOME>/jres directory.
Prior to version 3.12, the JIRA installer was bundled with a standard jre bundle downloaded from the install4j server called 'windows-x86-1.6.0.tar.gz'.

As of v3.12, we are creating a custom JRE bundle, in order to allow the use of the server JVM (only client JVM is included in the default windows jre bundle).
In order to do this, we create a custom JRE bundle from the JDK. JIRA v3.12 uses JDK 1.6.0_03:

1. Download and install the latest Windows JDK
2. Start install4j application.
3. Select Project > Create a JRE Bundle
4. On the "Select JRE" page, fill in the following values:
   Java home directory = where you just installed the JDK eg. "C:\Program Files\Java\jdk1.6.0_03\jre"
   Java version = the 3 digit version number of the JDK eg. "1.6.0"
   Custom id = something to name the build number and distinguish it form the standard bundle eg. "03_from_jdk"
   

Once you have created the custom JRE bundle, make sure that jira-template.install4j refers to the same JRE bundle name.
See the follow property in jira-template.install4j:

```
includedJRE="windows-x86-1.6.0_03_from_jdk"
```

or browse to Media > Windows > Bundled JRE in the install4j GUI.

Building the Installer

All source files regarding the installer reside in the main JIRA source tree in /jira/subprojects/installer. Edit project.properties to correctly identify your install4j home directory, as well as your certificate keystore:

subprojects/installer/project.properties

```
# The location used for code-signing the executable.  This location should contain an SPC (software
platfrom certificates)
# and PVK (Private key) file for windows.  Not, you will need mono to run this on unix/linux.
atlassian.keystore.location=/path/to/your/keystore
install4j.home=/path/to/your/install4j
```

Please note that the directory defined by the 'atlassian.keystore.location' property **must** contain 2 key files: atlassian.spc and private.pvk. Please see the Code signing section below for how to create these.

Once install4j has been installed, and project.properties has been correctly configured, the following maven command can be executed from the JIRA root directory to build the installer (Note: this will also build JIRA Standalone for the specified edition, i.e. Enterprise, Professional or Standard):

```
maven jira:installer -Dedition=enterprise
```

Code signing

Code signing is useful if you would like to ensure that the executable generated cannot be altered by anyone else. A code signed executable will also list the publisher (i.e. Atlassian Software Systems) correctly when opening the installer exe.

Background

To get a code-signing certificate, you firstly generate a public/private key pair using Java's keytool. The public key/certificate is then submitted to a key signing authority (Thawte in our case) who will verify that you really are who you claim to be, sign your public certificate and send it back. Your keystore stores your private key, public certificate as well as the whole certificate chain from your key signing authority.

Implementation

In Windows, you sign executables with a utility called signcode. There's a mono version for this and it's also included in install4j (with a slight modification that will allow you to specify a password for your private key).

Signcode takes a number of arguments:

1. An SPC file (Software Platform Certificates) which is basically a Windows format of your public certificate.
2. Your private key, in PVK format (also a Windows format).
3. Password for the private key.
4. A location of a timestamp server, such as http://timestamp.verisign.com/scripts/timstamp.dll.
5. The executable you want to sign.

To get all this information you will need some information from your keystore that Java's keytool can't return. The following Java class should return all required information:

```
import sun.misc.BASE64Encoder;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.*;
import java.security.cert.Certificate;

class ExportPriv
{
    public static void main(String args[]) throws Exception
```

```
    {
        ExportPriv myep = new ExportPriv();
        myep.doit();
    }

    public void doit() throws Exception
    {

        KeyStore ks = KeyStore.getInstance("JKS");
        String fileName = "/path/to/your/keystore";

        char[] passPhrase = "password".toCharArray();
        BASE64Encoder myB64 = new BASE64Encoder();


        File certificateFile = new File(fileName);
        ks.load(new FileInputStream(certificateFile), passPhrase);

        KeyPair kp = getPrivateKey(ks, "password", passPhrase);

        Certificate[] certificateChain = ks.getCertificateChain("atlassian");
        for (int i = 0 ; i < certificateChain.length; i++)
        {
            File output = new File("/path/to/your/output/directory/cert"+i+".crt");
            FileOutputStream out = new FileOutputStream(output);
            out.write(certificateChain[i].getEncoded());
            out.flush();
            out.close();

        }
        PrivateKey privKey = kp.getPrivate();


        String b64 = myB64.encode(privKey.getEncoded());

        File output = new File("/path/to/your/output/directory/private.key");
        FileOutputStream out = new FileOutputStream(output);
        out.write(privKey.getEncoded());

        out.flush();
        out.close();


        System.out.println("-----BEGIN PRIVATE KEY-----");
        System.out.println(b64);
        System.out.println("-----END PRIVATE KEY-----");

    }

    public KeyPair getPrivateKey(KeyStore keystore, String alias, char[] password)
    {
        try
        {
            // Get private key
Key key = keystore.getKey(alias, password);
            if (key instanceof PrivateKey)
            {
                // Get certificate of public key
Certificate cert = keystore.getCertificate(alias);

                // Get public key
PublicKey publicKey = cert.getPublicKey();

                // Return a key pair
return new KeyPair(publicKey, (PrivateKey) key);
            }
        } catch (UnrecoverableKeyException e)
        {
        } catch (NoSuchAlgorithmException e)
        {
```

```
        } catch (KeyStoreException e)
        {
        }
        return null;
    }
```

```
}
```

(Note: This code was copied from here with some modifications)

The get the SPC file you will first have to export your **entire** certificate chain from the keystore first using the Java class above (keytool does not provide a method to do this).
You can then convert this to a SPC file using:

```
cert2spc cert0.crt cert1.crt cert2.crt atlassian.spc
```

The private key returned from the keystore is in the wrong format for the signcode utiltiy. There exists a tool for Windows to convert the private key to the correct Windows PVK format.

Signcode can now be executed:

```
mono /path/to/your/install4j/resource/signcode.exe -spc atlassian.spc  -v private.pvk -vp password -t
http://timestamp.verisign.com/scripts/timstamp.dll JIRA_Enterprise_3_8-DEV_windows.exe
```

JIRA Plugin Tutorials

- Available Permissions
- How to create a JIRA Portlet
- How to create a JIRA Report
- How to create a new Custom Field Type
- How to create Custom Workflow Elements for JIRA 3

Available Permissions

For certain JIRA plugins (e.g. Portlets), you can specify permissions required to use the plugin (e.g. for portlets, these are the permissions required to add the portlet to your dashboard). The table below lists the available permissions and the defined constants for these permissions.

Key word	Corresponding JIRA Permission
admin	ADMINISTER
use	USE
project	PROJECT_ADMIN
browse	BROWSE
create	CREATE_ISSUE
attach	CREATE_ATTACHMENT
edit	EDIT_ISSUE
update	EDIT_ISSUE
scheduleissue	SCHEDULE_ISSUE
assignable	ASSIGNABLE_USER
assign	ASSIGN_ISSUE
resolve	RESOLVE_ISSUE
comment	COMMENT_ISSUE
close	CLOSE_ISSUE
work	WORK_ISSUE
link	LINK_ISSUE
delete	DELETE_ISSUE

sharefilters	CREATE_SHARED_FILTER
groupsubscriptions	MANAGE_GROUP_FILTER_SUBSCRIPTIONS
move	MOVE_ISSUE
setsecurity	SET_ISSUE_SECURITY
pickusers	USER_PICKER
viewversioncontrol	VIEW_VERSION_CONTROL
modifyreporter	MODIFY_REPORTER
viewvotersandwatchers	VIEW_VOTERS_AND_WATCHERS
managewatcherlist	MANAGE_WATCHER_LIST

How to create a JIRA Portlet

In JIRA 4, portlets have been superceded by gadgets.
For details, please see Gadget Development.

Overview

A portlet displays summary information on project/issue data through the JIRA dashboard. This information can range from statistical data to quick links for commonly used filters.

With JIRA 3.0, the introduction of the plugin system aimed to provide a simple point of extensibility for custom features users may wish to add to JIRA. This tutorial will describe how to create a custom portlet within JIRA 3.0, using this plugin interface.

The following articles provide further detailed information on writing a JIRA 3 plugin:

- JIRA Plugin Guide,
- How to create a new Custom Field Type
- How to create a JIRA Report
- Vincent Massol's tutorial

A Portlet Plugin Module Primer

In order to make a custom portlet available within JIRA, it is necessary to create a portlet plugin module. As with all plugin modules, the portlet plugin will consist of the following components:

- Java classes encapsulating portlet logic
- Resource templates for display of the portlet
- Plugin descriptor to enable the portlet module in JIRA

all contained within a single JAR file.

Portlet Logic

The Java classes include the necessary logic to retrieve the data used in configuring and displaying the portlet. The module class must implement *com.atlassian.jira.portal.Portlet* and will usually extend *com.atlassian.jira.portal.PortletImpl*. The *PortletImpl* interface provides methods to initialise the portlet through the parameters defined in the descriptor (name, key, thumbnail image, etc.) and retrieve the portlet template location. It is also possible to extend existing portlets to add further custom functionality.

Resource Templates

The second component consists of templates used to render the portlet - Velocity templates can be used here. The templates can include:

- Dashboard view - the actual dashboard view

The plugin system parses the *atlassian-plugin.xml* file for any configuration properties associated with the portlet that are required in order to display the portlet. The plugin system constructs a suitable configuration screen requesting the user to specify values for these properties.

Other Resources

It is possible to include i18n property files also - so as to allow other users to easily translate the strings used in the portlet for different languages. The following portlet examples are fully internationalized and include default property files.

Portlets are added through the dashboard configuration screen. The portlet selection displays the title and description along with a thumbnail image which links to a full image preview of the actual portlet display. It is necessary to include these image files (both the thumbnail and full image) within the final plugin JAR. To make the process as simple as possible, the images must be manually extracted from the JAR and copied to the *webapp/portlets/dashboard/thumbnails* directory in order for JIRA to access them. The full image location is specified in the *atlassian-plugin.xml* file - JIRA will search for the 'thumbnail image' in the same location.

Plugin Descriptor

The portlet module descriptor is the only mandatory part of the plugin. It must be called *atlassian-plugin.xml* and be located in the root of the JAR file.

Here is a sample portlet module descriptor element:

```
<!--
    The module class must implement
        com.atlassian.jira.portal.Portlet
    and will usually extend:
        com.atlassian.jira.portal.PortletImpl
-->
<portlet key="assignedtome" name="Assigned Issues" class=
"com.atlassian.jira.portal.portlets.AssignedToMePortlet">
    <description key="portlet.assignedtome.description">i18n description</description>
    <!-- this template produces the eventual HTML of the portlet -->
    <resource type="velocity" name="view" location="templates/plugins/jira/portlets/assignedtome.vm"
/>
    <label key="portlet.assignedtome.name" />
    <!-- an optional thumbnail image used to preview the portlet for users -->
    <thumbnail>portlets/dashboard/thumbnails/assigned.gif</thumbnail>
    <!-- the permissions required to add this portlet (optional - not often required) -->
    <permission>assignable</permission>
    <objectdescriptor key="portlet.assignedtome.display.name" />
    <!-- the properties of this report which the user must select before running it -->
    <properties>
        <property>
            <key>numofentries</key>
            <name>portlet.assignedtome.field.numofentries.name</name>
            <description>portlet.assignedtome.field.numofentries.description</description>
            <type>long</type>
            <default>10</default>
        </property>
    </properties>
</portlet>
```

In this sample, the portlet logic is encapsulated in the *AssignedToMePortlet* Java class. The view template location is specified in the *templates/plugins/jira/portlets* directory. The preview image is located in the *portlets/dashboard/thumbnails* directory. The **Assignable** permission is required for users to add this portlet to their dashboards. Following that, the parameters required to configure the portlet are specified - in this case, a numeric field specifying the number of issues to display in the portlet. You can find the list of available permissions here

JIRA Development Kit

You can choose to develop your plugins however you wish. However, we recommend using Maven 1.0 and the JIRA Plugin Development Kit.

Maven is an ant-like build tool that downloads any specified project dependencies automatically (just one of the many features).

The JIRA Plugin Development Kit consists of a number of examples (including the ones discussed here) to help developers extend JIRA through the plugin interface as easily as possible.

Once the JIRA Plugin Development Kit has been setup, you need only run the command:

```
maven jar
```

to build the desired plugin. Using Maven is **not** a requirement - you can use ant or any other build tool, however, it *will* make your life a lot easier.

Examples

The following examples detail how to extend an existing system portlet and how to create custom portlets. Each example is included in the JIRA Plugin Development Kit. When compiled, the *atlassian-jira-plugin-portlets-example-1.0.jar* JAR includes each portlet and can be copied to the JIRA *lib* directory in order to make the portlets available within the system.

Example 1 - Extending an Existing System Portlet

In this example, an existing system portlet - **'Assigned To Me'** - is extended to display the status of the issue. The original portlet displays a specified number of issues assigned to the current user, displaying the issue key, summary and priority.

This full source of this example is included in the development kit - the files of interest are:

- src/etc/templates/historyportlet/assignedtomeextended.vm
- src/etc/templates/historyportlet/issuesummaryextended.vm
- src/etc/atlassian_plugins.xml
- src/etc/com/atlassian/jira/plugin/portlet/example/assigned/assigned_portlet.properties
- src/etc/portlets/dashboard/thumbnails/assignedextended.png
- src/etc/portlets/dashboard/thumbnails/corner_assignedextended.png

In order to do this, it is only necessary to edit the view templates and i18n property files as all other aspects of the portlet can be reused. The information required to display the status is already passed to the template in the form of the issue itself - so the portlet logic does not need to be modified.

Firstly, the *atlassian-plugin.xml* file is created to include a new portlet module:

```
<atlassian-plugin key="com.atlassian.jira.plugin.portlet.example" name="JIRA Portlet Examples Plugin">
    <plugin-info>
        <description>Portlet Examples Plugin.</description>
        <version>1.0</version>
        <application-version min="3.0" max="3.0"/>
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
    </plugin-info>

    <!-- An simple example of customising an existing portlet - adding the status of an issue to the
template -->
    <portlet key="assignedtomeextended" name="Example: Assigned IssNues Extended"
        class="com.atlassian.jira.portal.portlets.AssignedToMePortlet">
        <description key="portlet.asssignedtome.extended.description">i18n description</description>
        <resource type="velocity" name="view" location=
"templates/assignedportlet/assignedtomeextended.vm" />
        <resource type="i18n" name="i18n" location=
"com.atlassian.jira.plugin.portlets.example.assigned.assigned_portlet" />
        <label key="portlet.assignedtome.extended.name" />
        <thumbnail>portlets/dashboard/thumbnails/assignedextended.png</thumbnail>
        <permission>assignable</permission>
        <objectdescriptor key="portlet.assignedtome.extended.display.name" />
        <properties>
            <property>
                <key>numofentries</key>
                <name>portlet.assignedtome.field.numofentries.name</name>
                <description>portlet.assignedtome.field.numofentries.description</description>
                <type>long</type>
                <default>10</default>
            </property>
        </properties>
    </portlet>

</atlassian-plugin>
```

As can be seen, the original *AssignedToMePortlet* Java class is reused - no code changes are necessary.

The remainder of the portlet definition specifies the following information:

- the name and description properties
- the location of the i18n files
- the location of the thumbnail files
- the parameters required to configure the portlet

The main change to note is the reference to a modified view template - one that displays the status. The modified view templates simply over-ride the templates used by the original **'Assigned to Me'** portlet. The *issuesummaryextended.vm* includes the following section in order to display the issue status in the dashboard view:

```
<td nowrap width=1%>
    #displayConstantIcon($issue.getStatusObject())
</td>
```

The i18n property strings have also been modified (in order to change the portlet name and description). These i18n property strings are defined in files that are included in the final JAR.

Finally, the thumbnails displaying a preview of the portlet are included in the JAR also. However, the thumbnails need to be manually copied to the *webapp/portlets/dashboard/thumbnails* directory in order for JIRA to access them.

The portlet will appear on the dashboard once selected, displaying the status of each issue.

Example 2 - Recently Updated Issues

In this example, a custom portlet is coded to display the most recently updated issues within a specific project on the dashboard. The portlet can be configured to display a specified number of issues over a specified number of days in the past.

This full source of this example is included in the development kit - the files of interest are:

- src/etc/templates/historyportlet/history.vm
- src/etc/templates/historyportlet/issuesummaryhistory.vm
- src/etc/atlassian_plugins.xml
- src/etc/com/atlassian/jira/plugin/portlet/example/history/history_portlet.properties
- src/etc/portlets/dashboard/thumbnails/history.png
- src/etc/portlets/dashboard/thumbnails/corner_history.png
- src/java/com/atlassian/jira/plugin/portlet/example/history/HistoryPortlet.java

Firstly, the *atlassian-plugin.xml* file is created/modified to include the new portlet:

```
<!-- A simple example portlet - displays recently issues updated for a specific user -->
    <portlet key="history" name="Example: Recently Updated Issues"
        class="com.atlassian.jira.plugin.portlet.example.history.HistoryPortlet">
        <description key="portlet.history.description">i18n description</description>
        <resource type="velocity" name="view" location="templates/historyportlet/history.vm" />
        <resource type="i18n" name="i18n" location=
"com.atlassian.jira.plugin.portlets.example.history.history_portlet" />
        <label key="portlet.history.name" />
        <thumbnail>portlets/dashboard/thumbnails/assignedextended.png</thumbnail>
        <objectdescriptor key="portlet.history.display.name" />
        <properties>
            <property>
                <key>projectid</key>
                <name>portlet.history.field.project.name</name>
                <description>portlet.history.field.project.description</description>
                <type>select</type>
                <values class="com.atlassian.jira.portal.ProjectValuesGenerator"/>
            </property>
            <property>
                <key>dayshistory</key>
                <name>portlet.history.field.dayshistory.name</name>
                <description>portlet.history.field.dayshistory.description</description>
                <type>long</type>
                <default>3</default>
            </property>
            <property>
                <key>numofentries</key>
                <name>portlet.history.field.numofentries.name</name>
                <description>portlet.history.field.numofentries.description</description>
                <type>long</type>
                <default>10</default>
            </property>
        </properties>
    </portlet>
```

As can be seen, a custom Java class - *HistoryPortlet* - encapsulates the logic required for configuring and displaying this portlet.

The remainder of the portlet definition in *atlassian-plugin.xml* specifies the following information:

- the name and description properties
- the location of the i18n files
- the location of the thumbnail files
- the parameters required to configure the portlet

In this case, the portlet requires three parameters to correctly display the data:

- projectid - a select field specifying the project on which the portlet will focus. The possible project options available are retrieved through another JIRA class - *com.atlassian.jira.portal.ProjectValuesGenerator*.
- dayshistory - a numeric field specifying the number of days to look back over.
- numofentries - a numeric field specifying the number of issues to display.

The class *HistoryPortlet* encapsulates the logic necessary to retrieve the issues most recently updated within the specified project over the specified number of days.

The view templates are very similar to the ones used in the **'Assigned To Me Extended'** example discussed above - the main changes taking place with the name and description properties.

Again, as with the previous example, the i18n property files and image previews are also included. The thumbnails need to be manually copied to the *webapp/portlets/dashboard/thumbnails* directory in order for JIRA to access them.

The portlet will appear on the dashboard once selected, displaying the specified number of recently updated issues within the specified project.

How to create a JIRA Report

Overview

A JIRA report can display statistical information based on all elements within JIRA - e.g. issues, projects, users, issue types, etc. The report logic collates and parses the data, which is then presented to the user as required.

With JIRA 3.0, the introduction of the plugin system aims to provide a simple point of extensibility for custom features users may wish to add to JIRA. This tutorial will describe how to create a custom report within JIRA 3.0, using this plugin interface.

The following articles provide further detailed information on writing a JIRA 3 plugin:

- JIRA Plugin Guide,
- How to create a new Custom Field Type
- How to create a JIRA Portlet
- Vincent Massol's tutorial

A Report Plugin Module Primer

In order to make a custom report available within JIRA, it is necessary to create a report plugin module. As with all plugin modules, the report plugin will consist of the following components:

- Java classes encapsulating report logic
- Resource templates for display of the report
- Plugin descriptor to enable the report module in JIRA

all contained within a single JAR file.

Report Logic

The Java classes include the necessary logic to retrieve the data used in configuring and displaying the report. The module class can implement the interface *com.atlassian.jira.plugin.report.Report* - or it can extend *com.atlassian.jira.plugin.report.impl.AbstractReport*. The main methods of interest are:

- generateReportHtml - generate HTML view of report
- generateReportExcel - generate Excel view of report
- getParams - retrieve the required data to be passed to the view template
- validate - validate any parameters used in configuring the report

Resource Templates

The second component consists of templates used to render the report - Velocity templates can be used here. The templates can include:

- Report view - the actual report view
- Excel view - an Excel view for the report (if required)

The plugin system parses the *atlassian-plugin.xml* file for any configuration parameters associated with the report - parameters required in order to display the report. The plugin system constructs a suitable configuration screen requesting the user to specify values for these parameters.

If an Excel view template is provided, users have the ability to view and further manipulate the data through Excel. If the Excel template is provided please ensure that your report also implements the following method:

```
public boolean isExcelViewSupported()
{
return true;
}
```

Other Resources

It is possible to include i18n property files also - so as to allow other users to easily translate the strings used in the report for different languages. The following report examples are fully internationalized and include default property files.

Plugin Descriptor

The report module descriptor is the only mandatory part of the plugin. It must be called *atlassian-plugin.xml* and be located in the root of the JAR file.

Here is a sample report module descriptor element:

```
<!--
 The "class" attribute here defines the actual report object.
 This is an implementation of com.atlassian.jira.plugin.report.Report.
 There is also an abstract implementation which may be useful at
 com.atlassian.jira.plugin.report.impl.AbstractReport.
-->
<report key="time-tracking" name="Time Tracking Report"
 class="com.atlassian.jira.plugin.report.impl.TimeTrackingReport">
<description key="report.timetracking.description">
 This report shows the time tracking details for a specific project.
</description>

<!-- the label of this report, which the user will use to select it -->
<label key="report.timetracking.label" />

<!-- the 'view' template is used to render the HTML result -->
<resource type="velocity" name="view"
 location="templates/plugins/jira/reports/time-tracking-report.vm" />
<!--
 The 'excel' template is used to render an Excel result.
 The 'Excel view' of the report will only be visible if
 this template exists for the plugin module
-->
<resource type="velocity" name="excel"
 location="templates/plugins/jira/reports/time-tracking-report-excel.vm" />

<!-- this is a .properties file containin the i18n keys for this report -->
<resource type="i18n" name="i18n"
 location="com.atlassian.jira.plugins.reports.timetracking" />

<!-- the properties of this report which the user must select before running it -->
<properties>
 <property>
  <key>versionId</key>
  <name>common.concepts.version</name>
  <description>report.timetracking.version.description</description>
  <!-- valid types are string, text, long, select, date -->
  <type>select</type>
  <!-- the values generator is a class which will
  generate values for this select list -->
  <values class="com.atlassian.jira.portal.VersionOptionalValuesGenerator"/>
 </property>
 <property>
  <key>sortingOrder</key>
  <name>report.timetracking.sortingorder</name>
  <description>report.timetracking.sortingorder.description</description>
  <type>select</type>
  <values class="com.atlassian.jira.portal.SortingValuesGenerator"/>
 </property>
 <property>
  <key>completedFilter</key>
  <name>report.timetracking.filter</name>
  <description>report.timetracking.filter.description</description>
  <type>select</type>
  <values class="com.atlassian.jira.portal.FilterValuesGenerator"/>
 </property>
</properties>
</report>
```

In this sample, the report logic is encapsulated in the *TimeTrackingReport* Java class. The view template location is specified in the *templates/plugins/jira/reports/time-tracking-report.vm* directory. The internationalistaion property files are located at *com.atlassian.jira.plugins.reports.timetracking*. Following that, the parameters required to configure the report are specifed - in this case, the version, the sort order and a filter.

JIRA Plugin Development Kit

You can choose to develop your plugins however you wish. However, we recommend using Maven 1.0 and the JIRA Plugin Development Kit.

Maven is an ant-like build tool that downloads any specified project dependencies automatically (just one of the many features).

The JIRA Plugin Developement Kit consists of a number of examples (including the ones discussed here) to help developers extend JIRA through the plugin interface as easily as possible.

Once the development kit has been setup, you need only run the command:

```
maven jar
```

to build the desired plugin. Using Maven is **not** a requirement - you can use ant or any other build tool, however, it *will* make your life a lot easier.

Examples

The following examples detail how to extend an existing sytem report and how to create a custom report. Each example is included in the JIRA Plugin Developement Kit. When compiled, the *atlassian-jira-plugin-report-example-1.0.jar* file includes each report and can be copied to the JIRA *lib* directory in order to make the reports available within the system.

Example 1 - Extending an Existing System Report

In this example, an existing system report - **'Single Level Group By Report'** - is extended to display the assignee and the last updated time of the issue. The logic and view templates within JIRA are slightly modified in order to achieve this extension.

The full source of this example is included in the JIRA Plugin Developement Kit in the *examples/plugin-report-sample* directory - the files of interest are:

- src/etc/templates/groupreport/single-groupby-reportextended.vm
- src/etc/atlassian_plugins.xml
- src/etc/com/atlassian/jira/plugin/reports/example/singlelevelgroup/singlelevelgroup_report.properties
- src/java/com/atlassian/jira/plugin/report/example/singlelevelgroup/SingleLevelGroupByReportExtended.java

Firstly, the *atlassian-plugin.xml* file is created/modifed to define the new report module element:

```
<!-- An simple example of customising an exisiting report -
 adding the admin and updated date of an issue to the template -->
<report key="singlelevelgroupbyextended" name="Example: Group By Report Extended"
 class="com.atlassian.jira.plugin.report.example.singlelevelgroup.SingleLevelGroupByReport">
<description key="report.singlelevelgroupby.description">i18n description</description>
<resource type="velocity" name="view"
 location="templates/groupreport/single-groupby-report-extended.vm" />
<resource type="i18n" name="i18n"
 location="com.atlassian.jira.plugin.reports.example.singlelevelgroup.singlelevelgroup_report" />
<label key="report.singlelevelgroupby.label.extended" />
<properties>
 <property>
  <key>filterid</key>
  <name>report.singlelevelgroupby.filterId</name>
  <description>report.singlelevelgroupby.filterId.description</description>
  <type>select</type>
  <values class="com.atlassian.jira.portal.SearchRequestValuesGenerator"/>
 </property>
 <property>
  <key>mapper</key>
  <name>report.singlelevelgroupby.mapper</name>
  <description>report.singlelevelgroupby.mapper.description</description>
  <type>select</type>
  <values class="com.atlassian.jira.issue.statistics.FilterStatisticsValuesGenerator"/>
 </property>
</properties>
</report>
```

The report definition specifies the following information:

- the class encapsulating the report logic (*SingleLevelGoupByReportExtended*)
- the name and description properties
- the location of the i18n files
- the parameters required to configure the report

The logic for the report is encapsulated in the class *SingleLevelGroupByReportExtended* (a slightly modified version of the original *SingleLevelGroupByReport* class within JIRA). The view template requires a manager to correctly display the last updated time for the issue - so the class passes the *OutLookDateManager OutlookDate* object to the velocity template:

```
...
try
{
I18nHelper i18n = new I18nBean(authenticationContext.getUser());

startingParams = EasyMap.build("action", action,
"statsGroup", getOptions(request, authenticationContext.getUser(), mapper),
"searchRequest", request, "mapperType", mapperName, "customFieldManager", customFieldManager,
"portlet", this);

startingParams.put("outlookDate",
ManagerFactory.getOutlookDateManager().getOutlookDate(authenticationContext.getLocale()));

return descriptor.getHtml("view", startingParams);
}
...
```

The last updated time can now appear in the correct date/time format as configured within JIRA.

The report view template is also edited from the original to display the assignee and the last updated time for the issue:

```
...
<td>$issue.getString('assignee')</td>
<td>$outlookDate.format($issue.getTimestamp('updated'))</td>
...
```

Once the compiled JAR file is placed in the lib directory - the report is available from the report menu on the 'Browse Project' page.

Example 2 - Issue Creation Report

In this example, a custom report is coded to display a histogram of issues created over a specified time. The report will collate all issues created within a specific project over the specifed duration, subdivided by a configurable time interval.

The full source of this example is included in the JIRA Plugin Developement Kit - the files of interest are located in the *example/plugin-report-sample* directory:

- src/etc/templates/creationreport/issuecreation-report.vm
- src/etc/atlassian_plugins.xml
- src/etc/com/atlassian/jira/plugin/reports/example/issuecreation/issue_creation_report.properties
- src/java/com/atlassian/jira/plugin/report/example/creationreport/CreationReport.java

Firstly, the *atlassian-plugin.xml* file is created/modified to include the new report module element:

```
<!-- An 'Issue Creation' Report - displays a histogram of issue create within a
 specifed project over a specified time -->
<report key="issuecreationreport" name="Example: Issue Creation Report"
 class="com.atlassian.jira.plugin.report.example.creationreport.CreationReport">
<description key="report.issuecreation.description">i18n description</description>
<label key="report.issuecreation.label" />

<resource type="velocity" name="view"
 location="templates/creationreport/issuecreation-report.vm" />
<resource type="i18n" name="i18n"
 location="com.atlassian.jira.plugin.reports.example.issuecreation.issuecreation_report" />

<properties>
 <property>
  <key>projectid</key>
  <name>report.issuecreation.projectid.name</name>
  <description>report.issuecreation.projectid.description</description>
  <type>select</type>
  <values class="com.atlassian.jira.portal.ProjectValuesGenerator"/>
 </property>
 <property>
  <key>startDate</key>
  <name>report.issuecreation.startdate</name>
  <description>report.issuecreation.startdate.description</description>
  <type>date</type>
 </property>
 <property>
  <key>endDate</key>
  <name>report.issuecreation.enddate</name>
  <description>report.issuecreation.enddate.description</description>
  <type>date</type>
 </property>
 <property>
  <key>interval</key>
  <name>report.issuecreation.interval</name>
  <description>report.issuecreation.interval.description</description>
  <type>long</type>
  <default>3</default>
 </property>
</properties>
</report>
```

The report module element defines that the logic is encapsulated in the class *CreationReport*. The name and description properties and the location of the i18n files are also specified.

In this case, the report requires four parameters to correctly display the data and are specified as follows:

- projectid - a select field specifiying the project on which the report will focus. The possible project options available are retrieved through another JIRA class - *com.atlassian.jira.portal.ProjectValuesGenerator*.
- startDate - a date field specifiying the start of the time period for the report.
- endDate - a date field specifiying the end of the time period for the report.
- interval - a numeric field specifiying the time interval used to divide the overall time period (this is the histogram interval).

The plugin system will construct a suitable report configuration screen - allowing the user to specify the above parameters.

Report Logic

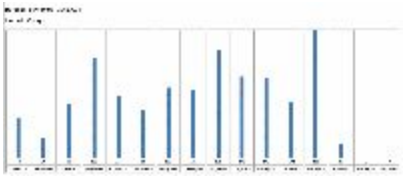
The *CreationReport* class retrieves the parameters as specified by the user. Next, the relevant issue counts are retrieved from the system for the specified time over the specifed time interval from the specified project. The issue counts are normalised in order to produce a balanced histogram. Finally, the relevant details are passed to the velocity template.

Resources

The view template displays the histogram constructed from the data passed from the class *CreationReport*.

The i18n property files are also included - allowing the text strings to be internationalised for users in different locales.

The report appears within the report section on the 'Browse Project' page.



Note: There is no Excel view for this report.

How to create a new Custom Field Type

Since JIRA 3.0 you have been able to create your own Custom Field Types through the plugin interface. In this tutorial, we'll take a look at a few simple examples and explain how you can easily achieve this.

Before you start, you may also want to familiarise yourself with the JIRA Plugin Guide.

> **A note about changed interfaces**
> We're always endeavouring to make JIRA better with each release. This often leads to new improvements and changes to the public interfaces in major JIRA versions (e.g. 3.1 to 3.2). This guide has been updated for the latest JIRA release, so if you're building a plugin for a prior JIRA version, keep an eye out for notes about differences in the interface.

A Quick Custom Field Types Primer

There's a few things you need to understand before diving into custom fields. A custom field type can have three components.

- Java Class encapsulating custom field logic
- Resource templates for display of custom field
- Module descriptor to enable the custom field module in JIRA

A custom field class extends the interface CustomFieldType. This interface provides methods to retrieve and store custom fields values. There are several *extension points* that are available to make creating new custom field types easier (e.g. CalculatedCFType, AbstractSingleFieldType, AbstractMultiSettableCFType). It is also possible to extend existing custom field types to add functionality (e.g. A currency type extending NumberCFType).

The second component are the resource templates which renders the custom field. There are four view types available, each representing a different context to render the custom field.

1. **`view`** - basic read-only view of the value (e.g. view issue, move issue confirm screen)
2. **`column-view`** - read-only view for displaying in the issue navigator. will default to *view* if omitted
3. **`edit`** - renders the edit widget for the custom field (e.g. edit issue, edit defaults)
4. **`xml`** - xml view of the value (e.g. rss, xml views)

Linking the Java code and rendering views are the plugin-module descriptors in your **`atlassian-plugin.xml`**. They allow JIRA to recognise what custom fields are available to the system and how to render them.

Example module descriptor

```
<atlassian-plugin key="com.atlassian.jira.plugin.customfield.example" name="JIRA Customfields Examples
Plugin">
<plugin-info>
<description>Customfields Examples Plugin.</description>
<version>1.0</version>
<application-version min="3.3" max="3.3"/>
<vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
</plugin-info>

<customfield-type key="textarea" name="Free Text Field (unlimited text)"
class="com.atlassian.jira.issue.customfields.impl.TextAreaCFType">
<description>A multiline text area custom field to allow input of longer text strings.</description>

<resource type="velocity" name="view" location="templates/plugins/fields/view/view-basictext.vm"/>
<resource type="velocity" name="column-view" location=
"templates/plugins/fields/view/view-limited-text.vm"/>
<resource type="velocity" name="edit" location="templates/plugins/fields/edit/edit-textarea.vm"/>
<resource type="velocity" name="xml" location="templates/plugins/fields/xml/xml-basictext.vm"/>
</customfield-type>
</atlassian-plugin>
```

You can also take a look at the default custom fields that shipped with JIRA here.

Information about setting up a complete plugin development environment for a plugin can be found here.
You can compile the examples below in the same way.

Admin-only editable field

For the first example, we'll construct a custom field that is only be editable by JIRA administrators and appear as a plain text to others. This is a simple customisation of the shipped `TextCFType` custom field and can be done by change one edit template.

First, we need to add the module to the `atlassian-plugin.xml`.

```
...
<customfield-type key="admintextfield" name="Admin Editable Text Field"
class="com.atlassian.jira.issue.customfields.impl.TextCFType">
<description>A text field only editable by JIRA-administrators. Others will see only text.
</description>
<resource type="velocity" name="view" location="templates/plugins/fields/view/view-basictext.vm"/>
<resource type="velocity" name="edit" location="templates/edit-jiraadminonlytext.vm"/>
<resource type="velocity" name="xml" location="templates/plugins/fields/xml/xml-basictext.vm"/>
</customfield-type>
...
```

A few points:

- *key* must uniquely identify the module in this plugin file.
- *name* & *description* are displayed when creating a new custom field instance

This module definition exactly matches that of a standard text field except for one line.

```
<resource type="velocity" name="edit" location="templates/edit-jiraadminonlytext.vm"/>
```

We are customizing the edit Velocity template so that it displays as a text box for an administrator but appears as uneditable text for others. Source for **`edit-jiraadminonlytext.vm`** is below.

edit-jiraadminonlytext.vm

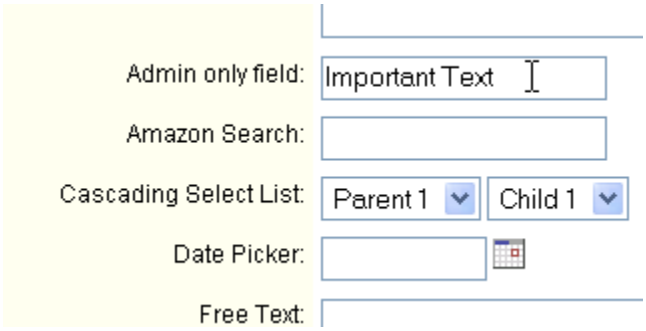
```
#controlHeader ($action $customField.id $customField.name $fieldLayoutItem.required
$displayParameters.noHeader)

#if ($authcontext.user.inGroup('jira-administrators'))
<input type="text"
name="$customField.id"
value="$!value" />
#else
#if($value && ! $value.equals(""))
#set ($displayValue = ${value})
#else
#set ($displayValue = 'N/A')
#end
<span title="This field is editable only by JIRA administrators">$!displayValue</span>
<input type="hidden"
name="$customField.id"
value="$!value" />
#end
#controlFooter ($action $fieldLayoutItem.fieldDescription $displayParameters.noHeader)
```

The above template checks if the user is part of group *jira-administrators*. If they are, display the text box, else display the value only as uneditable text.

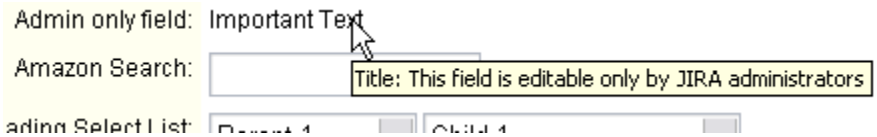
There's a few points to note.

- For what variables are available for a custom field you should check out the velocity context guide.
- #controlHeader and #controlFooter provide each custom field with the appropriate label and surrounding HTML table tags. This is required for all edit templates.

And that's it, a new custom field type. Deploy the JAR, login as an administrator and then a normal user and try it out.



Logged in as an admin



Logged in as a non-admin

Last commented user calculated field

The next example deals with a *Calculated Custom Field*. Calculated don't actually store any values. You often want or need this when you want to search on fields not normally available in JIRA, but the information can be derived. In this case, we want to return the last user who commented on the issue, if they are not an administrator. We only want this field to be visible in the issue navigator and not the edit or view pages.

Coding the Custom Field Type

Before you implement the interface CustomFieldType you should check out the latest javadoc. A useful extension point for calculated custom fields is, unsurprisingly, CalculatedCFType, where only three methods need to be implemented (getStringFromSingularObject, getSingularObjectFromString, and getValueFromIssue). If you also choose to implement SortableCustomField you will need to implement compare() as well.

The key method used to retrieve the value of our custom field is getValueFromIssue.

```
public Object getValueFromIssue(CustomField field, Issue issue)
{
User currentUser = authenticationContext.getUser();
User lastUser = null;
try
{
List comments = actionManager.getComments(issue.getGenericValue(), currentUser);
if (comments != null && !comments.isEmpty())
{
Comment lastComment = (Comment) comments.get(comments.size()-1);
User commenter = lastComment.getUser();
if (!commenter.inGroup(JIRA_ADMIN))
{
lastUser = commenter;
}
}
}
catch (GenericEntityException e)
{
}

return lastUser;
}
```

Note that prior to 3.3, the method had a GenericValue as the issue parameter. If you're developing for those JIRA versions make sure you correct your method signatures.

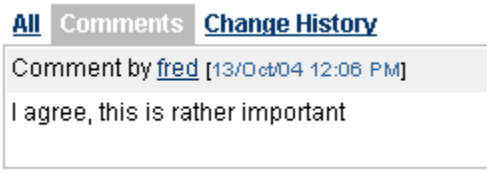
The return type `Object` is also known as the **Transport Object**. In this instance it is of type `User`, but it could be any other type. The Transport type must remain consistent across all methods such as create, update and also the view and edit templates.

Wiring it together

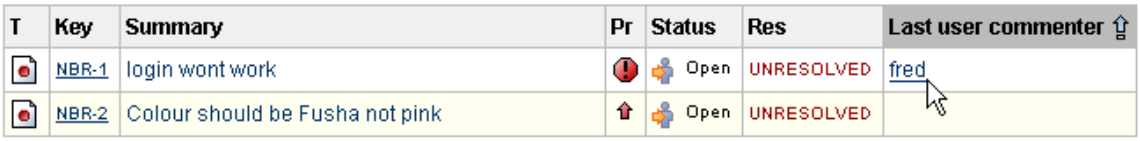
Much like the previous example, we can reuse some of the the templates that ship with JIRA.

```
<customfield-type key="lastusercommented" name="Last user commenter"
class="com.atlassian.jira.plugin.customfield.example.LastUserCommentedCFType">
<description>This is a lookup field that displays the last commenter who is not a JIRA administrator
</description>
<resource type="velocity" name="column-view" location="templates/plugins/fields/view/view-user.vm"/>
<resource type="velocity" name="xml" location="templates/plugins/fields/xml/xml-user.vm"/>
</customfield-type>
```

We can omit any resource types that we don't require. Thus both the edit and view templates are omitted here. The field should only appear when viewing through the issue navigator (column-view) and XML/RSS views (xml). The view user adds a link to the user details page and displays the full user name.



Fred is the last commenter



View in issue navigator

Enable Searching

The last commenter field in itself isn't all that useful. While we can see it in on the issue navigator, we can't search for a particular user who commented it last. Searching in JIRA 3 is handled by `CustomFieldSearchers`. Again several pre-configured searchers are available. You must ensure that the Transport Object are compatible between the custom field and the custom field searcher. Thus we can only use the UserPicker searcher since this is the only one that indexes User objects.

```
<customfield-searcher key="userpickersearcher" name="User Picker Searcher"
    i18n-name-key="admin.customfield.searcher.userpickersearcher.name"
    class="com.atlassian.jira.issue.customfields.searchers.UserPickerSearcher">
    <description key="admin.customfield.searcher.userpickersearcher.desc">Allow to search for a user
using a userpicker.</description>
    <resource type="velocity" name="label" location=
"templates/plugins/fields/view-searcher/label-searcher-user.vm"/>
    <resource type="velocity" name="search" location=
"templates/plugins/fields/edit-searcher/search-userpicker.vm"/>
    <resource type="velocity" name="view" location=
"templates/plugins/fields/view-searcher/view-searcher-basictext.vm"/>

    <valid-customfield-type package="com.atlassian.jira.plugin.customfield.example" key=
"lastusercommented"/>
</customfield-searcher>
```

This is quite similar to the CustomFieldType definition. The tag *valid-customfield-type* is used to associate the searcher to any number of custom field types. **Package** refers to the atlassian-plugin key attribute at the top of a plug-in and and the **key** refers to the module/customfield key.

Now when you create/edit your Last User Commented custom field, you'll be able to select the User Picker as a search template. You can now search on the last commenter field in the issue issue navigator.

Important When you change a search template for a custom field, you may need to perform a reindex before the search will work correctly. This issue is being tracked at JRA-4641.

Searching enabled

Sorting in Issue Navigator

To enable sorting you simply need to implement the interface **`SortableCustomField`**

```
public class LastUserCommentedCFType extends AbstractCustomFieldType implements SortableCustomField
```

The interface simply extends Comparable, so you need to implement the compare method.

```
public int compare(Object customFieldObjectValue1, Object customFieldObjectValue2, CustomFieldConfig
customFieldConfig)
{
return new BestNameComparator().compare(customFieldObjectValue1, customFieldObjectValue2);
}
```

The BestNameComparator is a simple helper type to facilitate comparing two users. You could just as easily write your own custom compare method.

Amazon search plugin

Lastly, a frivolous plug-in to give you some ideas on how to implement custom fields that perform remote look ups. Basically, we want a custom field that will take a text string and display a results from a search through the Amazon. There are several approaches to this, but by simplest solution is to treat the stored value as a simple text field and then add a *viewHelper* object that effectively transforms the string into the desired result.

Coding and Attaching the view helper

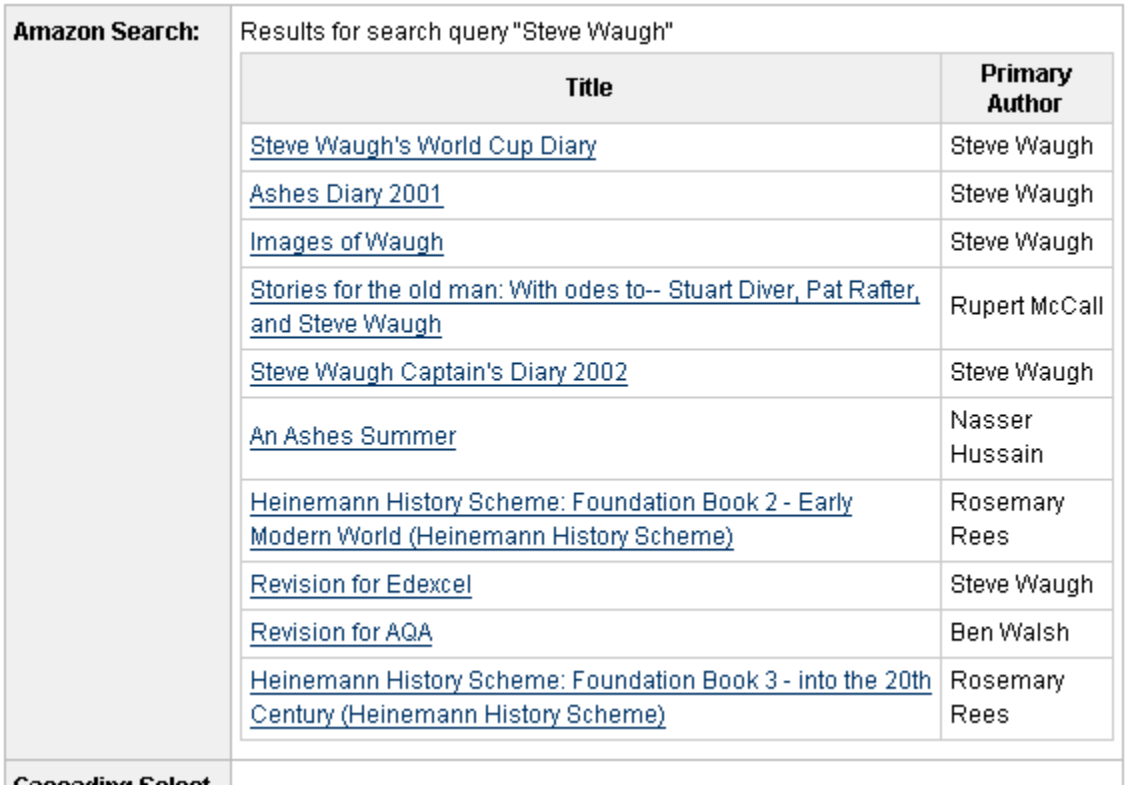
First we need to code our Amazon view helper. You can take a look in the source, but how it's been implemented isn't all that relevant. Once we have the view helper, we can pass this helper to the Velocity templates through the method *getVelocityParameters*

```
public Map getVelocityParameters(Issue issue)
{
Map map = new HashMap();
map.put("amazonSearchViewHelper", new AmazonSearchViewHelper());
return map;
}
```

The object `AmazonSearchViewHelper` is now accessible the velocity template. It has the method `searchForBooks` which returns a list of Books given some key words. We simply invoke this helper method in the template and display the results in a table.

```
#if ($value)
Results for search query "${value}" <br />

<table class="grid">
<tr>
<th>Title</th>
<th>Primary Author</th>
</tr>
#foreach ($book in $amazonSearchViewHelper.searchForBooks($value))
<tr>
<td><a target="_new" href="${book.link}">${book.title}</a></td>
<td>${book.firstAuthor}</td>
</tr>
#end
</table>
#end
```

You can utilise this same idea to display data from other remote systems, or even combine it with the readonly field to create your very own remote custom field.



Amazon Search: Results for search query "Steve Waugh"

Title	Primary Author
Steve Waugh's World Cup Diary	Steve Waugh
Ashes Diary 2001	Steve Waugh
Images of Waugh	Steve Waugh
Stories for the old man: With odes to-- Stuart Diver, Pat Rafter, and Steve Waugh	Rupert McCall
Steve Waugh Captain's Diary 2002	Steve Waugh
An Ashes Summer	Nasser Hussain
Heinemann History Scheme: Foundation Book 2 - Early Modern World (Heinemann History Scheme)	Rosemary Rees
Revision for Edexcel	Steve Waugh
Revision for AQA	Ben Walsh
Heinemann History Scheme: Foundation Book 3 - into the 20th Century (Heinemann History Scheme)	Rosemary Rees

Confluence Page Link custom field

> ⚠ This plugin is available here - and is not included in the jira-development-kit.

The 'Confluence Page Link' custom field plugin provides an example of implementing a custom field that performs a remote look up through XML/RPC..

This custom field provides a pop-up searcher - allowing the user to enter a search query that is executed over publicly accessible pages within a specified Confleunce instance. The user can select a result and the URL of thatpage is stored in the custom field - a simple text field. The Confluence instance to search against is specified in a properties file.

A new webwork action 'ConfluencePageBrowserAction' is required - allowing the popup window view to be associated with the action that performs and returns results from the Confluence page search.

The webwork action is registered in the atlassian-plugin.xml file as follows:

```
<webwork1 key="confluencepagebrowseraction" name="Confluence Page Browser Action" class="java.lang.
Object">
<actions>
<action name="com.atlassian.jira.plugin.confluencelinker.ConfluencePageBrowserAction" alias=
"ConfluencePageBrowser">
<view name="success">/templates/confluence_page_browser.vm</view>
</action>
</actions>
</webwork1>
```

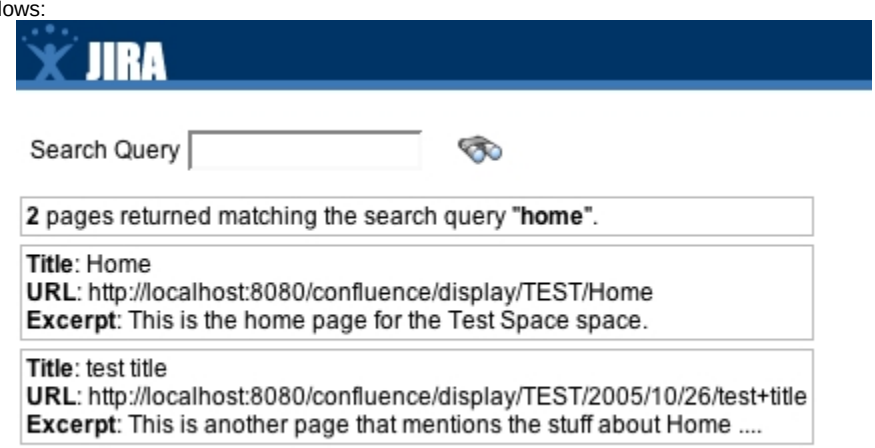
The ConfluencePageBrowserAction class is where the search logic is coded:

```
XmlRpcClient rpcClient = new XmlRpcClient(confluenceURL);
Vector xmlrpcResults = (Vector) rpcClient.execute("confluence1.search", makeParams(getSearchQuery(),
100));
if (xmlrpcResults != null)
{
earchResults = new ArrayList();
for (Iterator iterator = xmlrpcResults.iterator(); iterator.hasNext();)
{
Hashtable xmlrpcResult = (Hashtable) iterator.next();
searchResults.add(new SearchMatch(xmlrpcResult));
}
...
```

The Confluence page browser template displays the search query text box and the results:

```
#foreach ($result in $action.getSearchResults())
<tr onmouseover="rowHover(this)" onclick="selectLink('$result.getUrl()')">
<td>
<div class="borderedbox">
<b>Title</b>:   $result.getTitle()<br>
<b>URL</b>:     $result.getUrl()<br>
<b>Excerpt</b>: #if($result.getExcerpt())$result.getExcerpt() #else None #end
</div>
</td>
</tr>
```

The popup appears as follows:



How to create Custom Workflow Elements for JIRA 3

Overview

JIRA 3 provides the ability to create fully customized workflows - giving the user full control over the life cycle of a JIRA issue. This powerful feature allows the workflow designer to specify:

- the available actions at each step
- the users/groups who can execute a workflow transition
- functions that are executed on completion of a workflow transition

This tutorial focuses on the **Condition** and **Post Function** elements of a workflow - with an example of creating a custom condition and post function, and how they are integrated with JIRA through the plugin system.

The JIRA documentation contains further detailed information on workflows.

Also, the following documents expand on working with the JIRA plugin system:

- JIRA Plugin Guide
- How to create a new Custom Field Type

- How to create a JIRA Portlet
- Vincent Massol's tutorial

Plugin Structure

In order to make a custom workflow element (e.g condition, post-function) available within JIRA, it is necessary to create a workflow plugin module. As with all plugin modules, the workflow plugin will consist of the following components:

- Java classes encapsulating workflow element logic
- Resource templates for display of the workflow element
- Plugin descriptor to enable the workflow module in JIRA

all contained within a single JAR file.

Each element is further discussed in the examples below.

Jira Plugin Development Kit

The full source for each example is available in the JIRA Plugin Development Kit.

Using the JIRA Plugin Development Kit, it is possible to navigate to the workflow example directory and build the workflow example JAR file with the command:

maven jar

The resulting JAR file contains all the workflow examples discussed here. The workflow example plugin becomes available once the JAR file is copied to the JIRA *lib* directory.

Workflow Conditions

A **Condition** restricts the execution of a workflow transition until certain criteria are met. If the **Condition** fails, the transition link will not be available on the 'View Issue' page.

This section of the tutorial focuses on the **Condition** element and provides an example custom **Condition** which can be plugged into JIRA.

JIRA 3 System Conditions

JIRA 3 provides a number of system conditions available on setup - *DisallowIfInStepCondition*, *AllowOnlyAssignee*, *IssueAssignedCondition*, etc - each allowing the user to define when a worklfow transition becomes available.

The *SubTaskBlockingCondition* (another system condition) determines if a transition is available for an issue based on the status of its associated sub-tasks. The user specifies a list of statuses that will permit the transition to be available.

For example, the 'Close Issue' workflow transition link for an issue can be conditioned to be only available if all related sub-tasks are associated with the 'Closed' status. In effect, this transition link is not available for the parent issue until all sub-tasks are closed.

Custom Workflow Conditions

For developers designing a custom workflow condition, we recommend that the custom condition class extend the JIRA *AbstractJiraCondition* class. In order to avoid multiple database calls to retrieve the original issue object for the condition check, there are two possibilities available to the condition designer. Firstly, the custom condition class can overwrite the following method:

```
Issue getIssue(Map transientVars)
```

The logic within this method should retrieve the original issue object as required.

Alternatively, if the getIssue method is not overwriten, it is possible to pass the original issue object to the *transientVars* map, for example:

```
GenericValue origianlIssueGV =
ComponentManager.getInstance().getIssueManager().getIssue(issue.getId());
fields.put(AbstractJiraCondition.ORIGNAL_ISSUE_KEY, IssueImpl.getIssueObject(origianlIssueGV));
```

In this instance, the *fields* object will be passed to the *getIssue* method as the *transientVars* map.

This ensures that the original issue is examined during the condition check and minimal database calls are made.

Example - Parent Issue Blocking Condition

This example provides the reverse condition of the *SubTaskBlockingFunction* - in that it determines if a transition is available for a sub-task based

on the status of its associated parent issue.

In this example, the condition has been configured to display the workflow transition 'Reopen' for a sub-task, only if the parent issue is associated with an unresolved status (e.g. 'Open', 'In Progress' , Unresolved').

The condition is applied to the 'Reopen' transition in a copy of the default JIRA workflow associated with the 'Sub-Task' issue type.

In effect, the condition will prevent the transition for any sub-task from the 'Closed' to the 'Reopened' status, if the parent issue is not associated with an unresolved status.

Condition Logic

The condition logic is contained in the class *ParentIssueBlockingCondition* class that implements the interface *Condition*.

The only method requiring implementation is the *passesCondition(...)* method. Within this example, this method retrieves the parent issue and then determines if its associated status is contained in the user specified list of statuses. The condition passes if the specified list of statuses contains the status associated with the parent issue.

The list of statuses is specified when adding the workflow condition to a transition.

The class *WorkflowParentIssueBlockingConditionFactoryImpl* is also included - this class manages passing the required parameters to the resource templates.

Condition Resources

The workflow condition requires a number of resources in order to display the input, edit and view screens.

In this example, a velocity template is provided for each screen:

- *templates/issueblockingcondition/issueblockingcondition-input-params.vm*
- *templates/issueblockingcondition/issueblockingcondition-edit-params.vm*
- *templates/issueblockingcondition/issueblockingcondition-view.vm*

allowing the user to initially specify the statuses which will result in a 'pass', to edit these statuses and also a screen displaying the selected statuses.

Plugin Descriptor

As with all plugins, the workflow condition must be defined in a file named *atlassian-plugin.xml* and be located in the root of the JAR file.

The definition of the *ParentIssueBlockingCondition* condition is as follows:

```
<atlassian-plugin key="com.atlassian.jira.plugin.workflow.example" name="Workflow Examples Plugin">
<plugin-info>
<description>Example JIRA Workflow Elements</description>
<version>1.0</version>
<application-version min="3.0" max="3.0"/>
<vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
</plugin-info>

<workflow-condition key="issueblocking-condition" name="Parent Issue Blocking Condition"
class="com.atlassian.jira.plugin.workflow.example.WorkflowParentIssueBlockingConditionFactoryImpl">
<description>Condition to block sub-task issue transition depending on
parent issue status.</description>

<condition-class>
com.atlassian.jira.plugin.workflow.example.condition.ParentIssueBlockingCondition</condition-class>

<resource type="velocity" name="view"
location="templates/issueblockingcondition/issueblocking-condition-view.vm"/>
<resource type="velocity" name="input-parameters"
location="templates/issueblockingcondition/issueblocking-condition-input-params.vm"/>
<resource type="velocity" name="edit-parameters"
location="templates/issueblockingcondition/issueblocking-condition-edit-params.vm"/>
</workflow-condition>
</atlassian-plugin>
```

The workflow condition entry specifies the key, name and the Workflow Condition Factory for this condition. The factory class provides methods for passing input, edit and view parameters to the view templates. The condition description is also specified.

The class containing the condition logic, *ParentIssueBlockingCondition*, is specified next in the *<condition-class>* tag.

Finally, the location of the resource templates are specified - with an individual template for input, edit and view screens.

Parent Issue Blocking Condition - In Action

Once the workflow example JAR file has been placed in the JIRA *lib* directory, the **Parent Issue Blocking Condition** is now available as a condition within the workflow editor.

Workflow Post Functions

A **Post Function** executes specifed actions immediately after a transition is executed (hence the name post-function). Example possible actions include updating issue fields, generating change history, adding a comment, generating an event that signals that an issue has been progressed through workflow, etc.

This section of the tutorial focuses on the **Post Function** element and provides an example **Post Function** which can be plugged into JIRA.

JIRA 3 System Post Functions

JIRA 3 provides a number of system post functions available on setup - *UpdateIssueStatusFunction*, *CreateCommentFunction*, etc - each allowing the user to specify that certain actions should be executed following a specific workflow transition.

Note: Certain JIRA system post functions cannot be edited, deleted or ordered, as they must be executed during every transition. These post functions are essential for JIRA's issue life cycle, and would compromise other functionality if not executed.

Example - Close Parent Issue Post Function

This example post function will close the parent issue once the final sub-task is closed (all other associated sub-tasks are already closed).

The post function will ensure that the parent issue is still open and that all other associated sub-tasks are also closed before attempting to close the parent issue.

The post function can be applied to the 'Close Issue' transition in a copy of the default JIRA workflow associcated with the 'Sub-Task' issue type.

Post Function Logic

The post function logic is contained in the class *CloseParentIssueFunction* class that implements the interface *FunctionProvider*.

The *execute* method retrieves the sub-task from the parameters. From this, the parent issue is determined and a check is made as to whether the parent issue is closed or not.

If the parent issue is not closed, the statuses of the rest of the associated sub-tasks are also checked. If all sub-tasks are closed, the parent issue can be closed.

This function does not require any input or configuration - the action to be executed is defined within the post function logic.

The only user input required is to associate the post function with a particular transition within a workflow.

Post Function Resources

This post function only requires a view template as there is no configuration or editing required.

The velocity template is provided for the view screen:

- *templates/closeparentfunction/closeparentissue-function-view.vm*

Plugin Descriptor

As with all plugins, the post function condition must be defined in a file named *atlassian-plugin.xml* and be located in the root of the JAR file.

The definition of the *CloseParentIssueFunction* condition is as follows:

```
...
<workflow-function key="closeparentissue-function" name="Close Parent Issue Function"
class="com.atlassian.jira.plugin.workflow.WorkflowNoInputPluginFactory">
<description>Closes the parent issue on closing final associated
sub-task (all other sub-tasks are closed).</description>

<function-class>
com.atlassian.jira.plugin.workflow.example.function.CloseParentIssueFunction</function-class>

<orderable>true</orderable>
<unique>true</unique>
<deletable>true</deletable>
<default>true</default>

<resource type="velocity" name="view"
location="templates/closeparentfunction/closeparentissue-function-view.vm"/>
</workflow-function>
...
```

Here, the WorkflowNoInputPluginFactory class must implement WorkflowPluginFunctionFactory.

The post function entry specifies the key, name and the Post Function Factory for this condition. The factory class provides methods for passing parameters to the view templates - in this case, no parameter passing is required. The post function description is also specified.

The class containing the post function logic, *CloseParentIssueFunction*, is specified next in the *<function-class>* tag.

It is also possible to configure the post function as it appears in the workflow editor. The following options can be specified:

- orderable - specifies if this post function can be re-ordered within the list of post functions associated with a transition. The postion within the list determines when the post function actually executes.
- unique - specifies if this post function is unique or not - i.e. if it is possible to add multiple instances of this post function on a single transition.
- deleteable - specifies if the post funtion can be removed from a transition.

- default - specifies if this post function is automatically associated with any new transitions created.

It is also possible to specify a *weight* configuration parameter - however this is mainly intended for JIRA system post function elements. This parameter is used in conjunction with the *default* parameter - if the post function is to be added to all new transitions, the *weight* parameter is used to determine the post function position within the post function list.

Finally, the location of the resource view template is specified.

Close Parent Issue Post Function - In Action

Once the workflow example JAR file has been placed in the JIRA *lib* directory, the **Close Parent Issue Post Function** is now available as a post function within the workflow editor.

JIRA RPC Services

JIRA ships with an RPC plugin which enables limited remote access to JIRA. It is available through SOAP and XML-RPC interfaces. We recommend using the SOAP interface when possible as it is more complete and will be our primary focus in the future. This page provides a starting point for all your remote procedure call needs

You can find the latest news on the RPC plugin in the JIRA Extensions space. The full source of the plugin is available and you are free to modify and the extend the source. We'd also be happy to accept code contributions to the project, as Simon Mittag has done in the past. Check out the RPC Endpoint Plugin Module for more information.

SOAP Service

SOAP is the preferred method for remote method calls in JIRA and is the most frequently updated. The JIRA SOAP service is backed by Apache Axis, an open-sourced Java web services framework. If you're familiar with SOAP, ensure that RPC is enabled and point your stub generator to the **`/rpc/soap/jirasoapservice-v2?wsdl`** path of your JIRA install (e.g. http://jira.atlassian.com/rpc/soap/jirasoapservice-v2?wsdl) and you're away. For those less familiar with SOAP, you should first check out the SOAP tutorial.

For the latest information on what remote methods are available, you should check out the latest javadoc for JiraSoapService.

- SOAP tutorial
- Latest javadoc for JiraSoapService

XML RPC Service

XML-RPC can also be used as an alternative where SOAP is unavailable. You can find some background information on XML-RPC by reading the overview. Start building your own client by following the instructions in the tutorial. The XML-RPC interface is backed by the SOAP interface, and will usually have a subset of methods from the SOAP interface.

For the latest information on what remote methods are available, you should check out the latest javadoc for XmlRpcService . Method calls should be prefixed with **`jira1.`** and be made to the URL **`/rpc/xmlrpc`** of your install. The javadoc will often refer to "hashtables with fields from RemoteObject". To the hashtable will contain keys that map to the fields available through reflection of the particular RemoteObject. For example, the object `RemoteVersion`, has the methods getReleaseDate(), getSequence(), isArchived() and isReleased(). This will be converted into a Hashtable with keys releaseDate, sequence, archived and released.

- JIRA XML-RPC Overview
- Creating a XML-RPC Client
- Latest javadoc for XmlRpcService

HTTP/REST and other remote services

Lastly, all JIRA functions can be accessed through REST styled service. All functions in JIRA can be accessed through URL manipulation as a GET request. By appending the parameters `?os_username=<username>&os_password=<password>` and other required parameters, users can essentially run any functions in JIRA. e.g. `http://<your server>/secure/admin/jira/IndexReIndex.jspa?os_username=<admin_username>&os_password=<admin_password>` can trigger a reindex. The advantage of this lies in the number of methods exposed and is used by the Eclipse plugin and the IDEA plugin. This approach can be difficult, however, as it usually requires parsing of HTML or XML. Building the the correct URL requires knowing what parameters to submit. For details, see the Automating JIRA operations via wget page.

Other alternative methods to access JIRA would also be other APIs to navigate a web application such as jWebUnit. While normally used for testing, it can successfully be used as a remote interface into JIRA. You can use jWebUnit to log in, navigate to a certain page, and create issues, delete projects etcetera; all the functions available to any real user. If performance is not a major concern, using jWebUnit can be less error prone than simply using REST styled services.

Creating a SOAP Client

JIRA 3.0 and above ships with the RPC plugin which enables remote access through XML-RPC and SOAP.This document contains notes on how to perform various operations (e.g. creating issues) via a Java SOAP client (JIRA 3.1 or above required).

Remotely exposed operations.

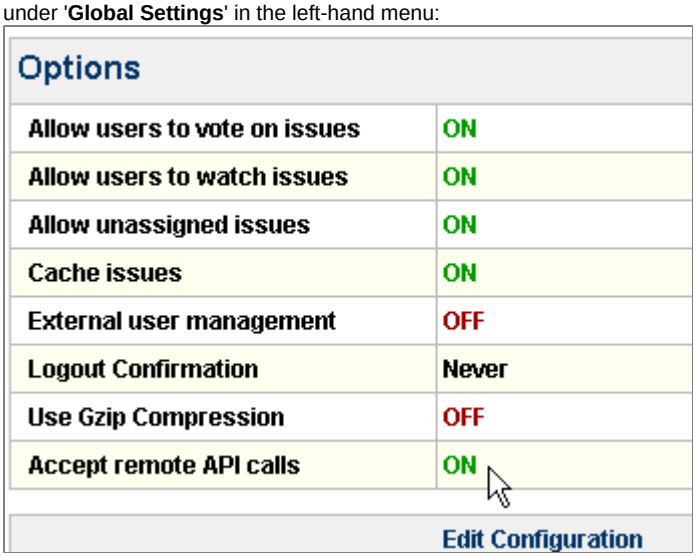
Before you begin, check out the javadoc for the RPC plugin, specifically JiraSoapService, which has information on all the methods available through SOAP and and XML-RPC. Also check the list of RPC bugs, listed on the RPC plugin page, to see that none will affect you.

Please note that the SOAP service respects the permissions and screen configurations that are set up in JIRA. For example, if you have configured JIRA so that the screen for a given issue type does not include a 'Summary' field, then you will not be permitted to set a value for the 'Summary' field through the SOAP request.

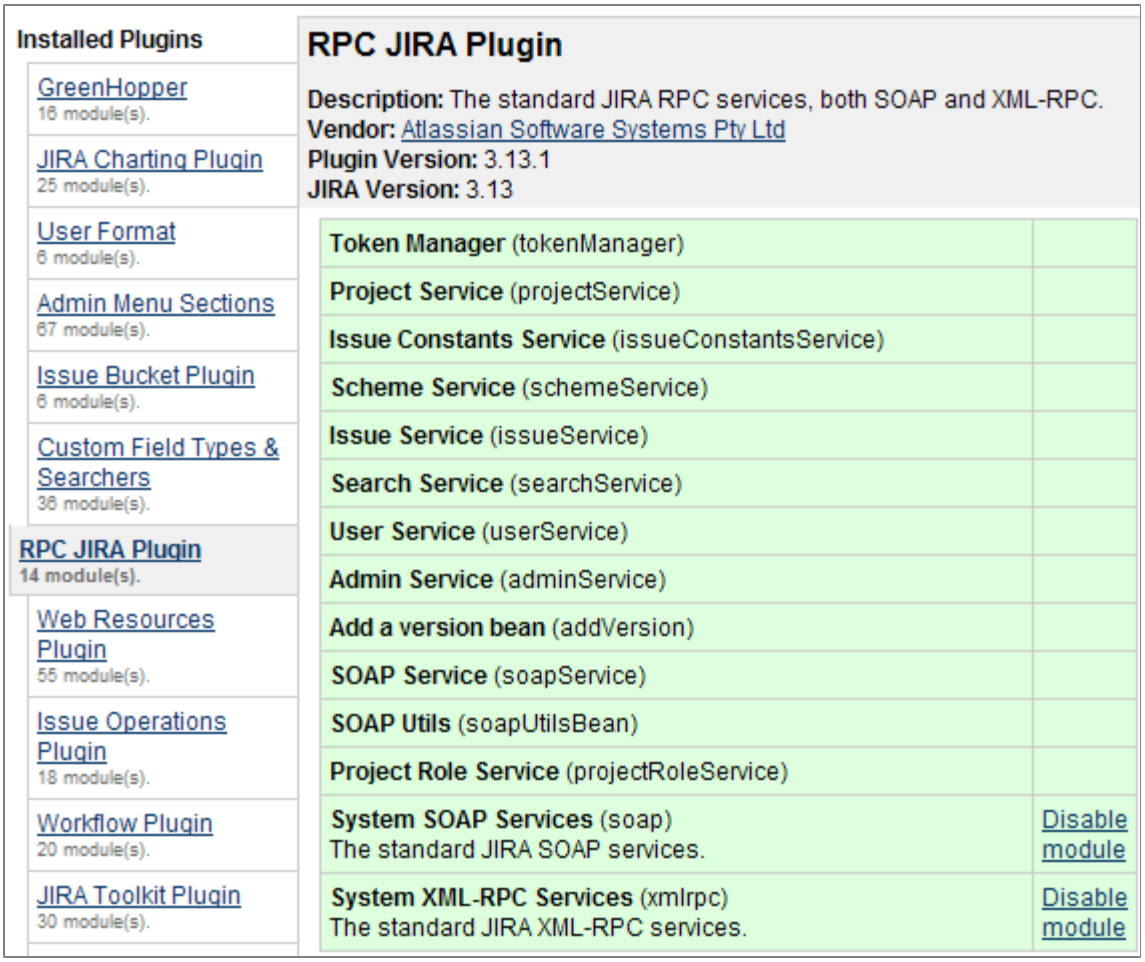
Some of our users have recommended SOAPUI (http://www.soapui.org/).
It allows you to import the WSDL from JIRA and it then displays all remote calls and all sorts of details and testing possibilities.
Check it out!
Thanks goes to Karl-Koenig Koenigsson for this info.

Enable the RPC plugin

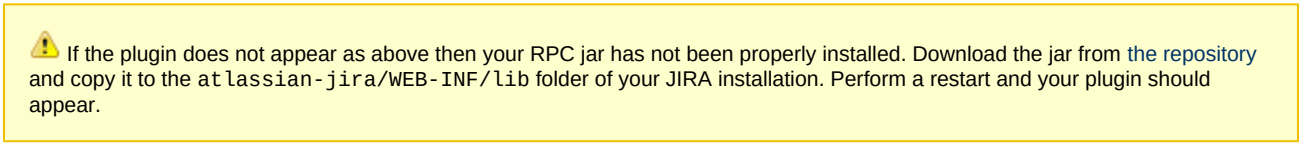
To invoke JIRA operations remotely, you should ensure that the RPC plugin is enabled on the JIRA installation you are targeting. If you simply want to create a client to http://jira.atlassian.com/ then you can skip this step. First you need to check if the Accept Remote API Calls has been enabled in '**General Configuration**' under '**Global Settings**' in the left-hand menu:



Then you need to enable the JIRA RPC Plugin in '**Plugins**' under '**System**' in the left-hand menu:



!



Your server should now be ready to accept remote procedure calls.

WSDL descriptor

Regardless of the language or SOAP API used, you will need the WSDL descriptor for your JIRA installation. This is found at http://`your_installation`/rpc/soap/jirasoapservice-v2?wsdl. For instance, http://jira.atlassian.com's WSDL file is:

http://jira.atlassian.com/rpc/soap/jirasoapservice-v2?wsdl

In JIRA 3.0.x, the URL is http://`your_installation`/rpc/soap/jiraservice-v1.wsdl. The sample SOAP client below has this pre-packaged, so you don't need to do anything further if using it.

Sample Java SOAP client

Download the latest demo SOAP client distribution. This contains a Maven project configured to use Apache Axis, and a sample Java SOAP client which creates test issues in http://jira.atlassian.com.
Read the README.txt in the root directory for further instructions.

To give you an idea of what a Java SOAP client looks like, here is the source for our demonstration client, which creates issues on http://jira.atlassian.com/browse/TST

```
System.out.println("Creating a test issue on http://jira.atlassian.com ...");
JiraSoapServiceService jiraSoapServiceGetter = new
JiraSoapServiceServiceLocator();

JiraSoapService jiraSoapService = jiraSoapServiceGetter.getJirasoapserviceV2();

String token = jiraSoapService.login("soaptester", "soaptester");

// Create the issue
RemoteIssue issue = new RemoteIssue();
issue.setProject("TST");
issue.setType("1");
issue.setDuedate(Calendar.getInstance());

RemoteComponent component = new RemoteComponent();
component.setId("10242");

issue.setComponents(new RemoteComponent[]{component});
issue.setSummary("This is a new SOAP issue " + new Date());

// Make up some remote versions
RemoteVersion version = new RemoteVersion();
version.setId("10330");
RemoteVersion[] remoteVersions = new RemoteVersion[]{version};
issue.setFixVersions(remoteVersions);

RemoteIssue returnedIssue = jiraSoapService.createIssue(token, issue);
System.out.println("Successfully created issue http://jira.atlassian.com/browse/"
+returnedIssue.getKey());
```

The various external classes (JiraSoapService etc) are the classes generated automatically from WSDL by the Maven Axis plugin.

Python (SOAPPy) client

Due to a JIRA/Axis bug, Python clients will not work with JIRA versions earlier than 3.3.1. In 3.3.1 and above, the following code demonstrates how to create an issue and comment (on http://jira.atlassian.com):

```
#!/usr/bin/python

# Sample Python client accessing JIRA via SOAP. By default, accesses
# http://jira.atlassian.com with a public account. Methods requiring
# more than basic user-level access are commented out. Change the URL
# and project/issue details for local testing.
#
# Note: This Python client only works with JIRA 3.3.1 and above (see
# http://jira.atlassian.com/browse/JRA-7321)
#
# Refer to the SOAP Javadoc to see what calls are available:
# http:
//docs.atlassian.com/software/jira/docs/api/rpc-jira-plugin/latest/com/atlassian/jira/rpc/soap/JiraSoapSe

import SOAPpy, getpass, datetime

soap = SOAPpy.WSDL.Proxy('http://jira.atlassian.com/rpc/soap/jirasoapservice-v2?wsdl')
#soap = SOAPpy.WSDL.Proxy('http://localhost:8090/jira/rpc/soap/jirasoapservice-v2?wsdl')

#jirauser = raw_input("Username for jira [fred]: ")
#if jirauser == "":
#    jirauser = "fred"
#
#passwd = getpass.getpass("Password for %s: " % jirauser)
#passwd="fredspassword"

jirauser='soaptester'
```

```
passwd='soaptester'

# This prints available methods, but the WSDL doesn't include argument
# names so its fairly useless. Refer to the Javadoc URL above instead
#print 'Available methods: ', soap.methods.keys()

def listSOAPmethods():
 for key in soap.methods.keys():
     print key, ': '
     for param in soap.methods[key].inparams:
  print '\t', param.name.ljust(10), param.type
     for param in soap.methods[key].outparams:
  print '\tOut: ', param.name.ljust(10), param.type


auth = soap.login(jirauser, passwd)

issue = soap.getIssue(auth, 'TST-3410')
print "Retrieved issue:", issue
print

# Note: if anyone can get timestamps to work, please let us know how!

baseurl = soap.getServerInfo(auth)['baseUrl']
newissue = soap.createIssue(auth, {'project': 'TST', 'type': '1', 'summary': 'Issue created with
Python!'})
print "Created %s/browse/%s" % (baseurl, newissue['key'])

print "Adding comment.."
soap.addComment(auth, newissue['key'], {'body': 'Comment added with SOAP'})

print 'Updating issue..'
soap.updateIssue(auth, newissue['key'], [
  {"id": "summary", "values": ['[Updated] Issue created with Python'] },

  # Change issue type to 'New feature'
  {"id":"issuetype", "values":'2'},

  # Setting a custom field. The id (10010) is discoverable from
  # the database or URLs in the admin section

  {"id": "customfield_10010", "values": ["Random text set in updateIssue method"] },

  {"id":"fixVersions", "values":['10331']},
  # Demonstrate setting a cascading selectlist:
  {"id": "customfield_10061", "values": ["10098"]},
  {"id": "customfield_10061_1", "values": ["10105"]},
  {"id": "duedate", "values": datetime.date.today().strftime("%d-%b-%y")}

  ])

print 'Resolving issue..'
# Note: all fields prompted for in the transition (eg. assignee) need to
# be set, or they will become blank.
soap.progressWorkflowAction(auth, newissue['key'], '2', [
  {"id": "assignee", "values": "jefft" },
  {"id":"fixVersions", "values":['10331']},
  {"id": "resolution", "values": "2" }
  ])


# Re. 'assignee' above, see http://jira.atlassian.com/browse/JRA-9018

# This works if you have the right permissions
#user = soap.createUser(auth, "testuser2", "testuser2", "SOAP-created user", "newuser@localhost")
#print "Created user ", user

#group = soap.getGroup(auth, "jira-developers")
# Adding a user to a group. Naming the parameters may be required (see
# http://jira.atlassian.com/browse/JRA-7971). You may experience other
```

```
# problems (see http://jira.atlassian.com/browse/JRA-7920).
#soap.addUserToGroup(token=auth, group=group, user=user)

# Adding a version to a project. If you figure out the syntax for the date please submit it back to
Atlassian
#soap.addVersion(auth, "TST", {'name': 'Version 1'})


print "Done!"
```

```
# vim set textwidth=1000:
```

Python's SOAP support is considerably less well developed than Java's, and some SOAP calls may fail (eg. addUserToGroup()). If you have problems, please raise an issue with the Python snippet, and stacktrace.

Ruby client

Ben Walding (Codehaus) reports:

> *I've worked with the developers of SOAP4R and they've made the Ruby SOAP libraries work with JIRA. It was the client libraries at fault - they were ignoring the WSDL and sending integers instead of longs.*
>
> *A gem for marginally simpler access to JIRA can be found at http://jira4r.rubyhaus.org/ (there is also a confluence gem at http://confluence4r.rubyhaus.org/*
>
> *The gems take care of authentication and provide helper methods where the Jira / Confluence interfaces are missing useful methods or behave strangely (eg. provides a getProject; provides a getGroup that doesn't throw exceptions).*

There is a JIRA4R sample Ruby script available in the samples (thanks Jonathan Zhang). If you have any Ruby samples to share, please let us know so we can include them into the repository as well.

See also

Creating a XML-RPC Client

Remote API (SOAP) Examples

More code examples can be found in the SVN repository. You may also try looking for them in the jira-user and jira-developer mailing list (or forum).

Perl

Logging In

```
#!/usr/bin/perl

use SOAP::Lite;
use Data::Dumper;

my $soap = SOAP::Lite->proxy("http://localhost:8090/rpc/soap/jirasoapservice-v2?wsdl");
my $auth = $soap->login("admin", "admin");
```

Creating Issue

```
$issueMap = {"project" => SOAP::Data->type(string => "YQ"),
             "components" => [{"id" => SOAP::Data->type(string => "10010")}],
             # this is definitely not working as "10010" will be 'autotyped' to int:
             # "components" => [{"id" => "10010"}],
             "type" => SOAP::Data->type(string => "1"),
             "summary" => SOAP::Data->type(string => "Issue created via Perl/SOAP")
            };

my $issue = $soap->createIssue($auth->result(), $issueMap);
```

Python

Logging In

```
#!/usr/bin/python

import SOAPpy, getpass, datetime, array, base64, random
from SOAPpy import Types
soap = SOAPpy.WSDL.Proxy('http://localhost:8090/rpc/soap/jirasoapservice-v2?wsdl')
jirauser='admin'
passwd='admin'

auth = soap.login(jirauser, passwd)
```

Adding User to Group

```
group = soap.getGroup(auth, 'foo')
user = soap.getUser(auth, 'admin')
user = {'name': user['name']} # without this line, you might be facing some funny problems. see
JRA-7920.
soap.addUserToGroup(auth, group, user)
```

Listing Workflow Actions and associated Fields and Progressing

```
# List
actions = soap.getAvailableActions(auth, 'MYC-28')
print "actions:"
for action in actions:
  print action
  fields = soap.getFieldsForAction(auth, 'MYC-28', action['id'])
  for field in fields:
    print field;
  print "-----"

# Progress
issue = soap.progressWorkflowAction(auth, 'MYC-28', '5', [{'id': 'resolution', 'values': ['2']},
{'id': 'assignee', 'values': ['admin']}, {'id': 'comment', 'values': ['testo!']}])
```

Creating a XML-RPC Client

JIRA 3.0 and above ships with the JIRA XML-RPC Plugin which enables remote access through XML-RPC and SOAP. Utilising this feature with XML-RPC couldn't be much easier with some help from the Apache XML-RPC package. In this tutorial, we write a basic XML-RPC client (using Apache XML-RPC) that logs in, retrieves projects and then log out again. A Python client is also demonstrated.

You may also be interested in the Creating a SOAP Client (more methods are exposed via SOAP than XML-RPC).

Getting the latest XML-RPC client

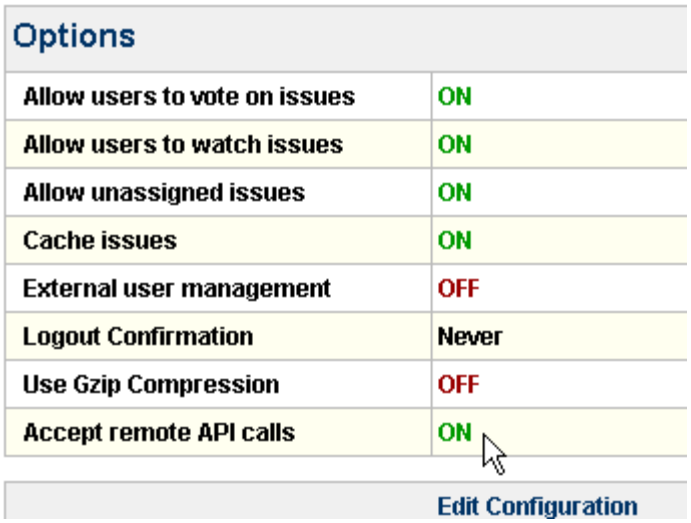
You can download the latest XML-RPC client with the JIRA Plugin Development Kit

Enabling the RPC plugin

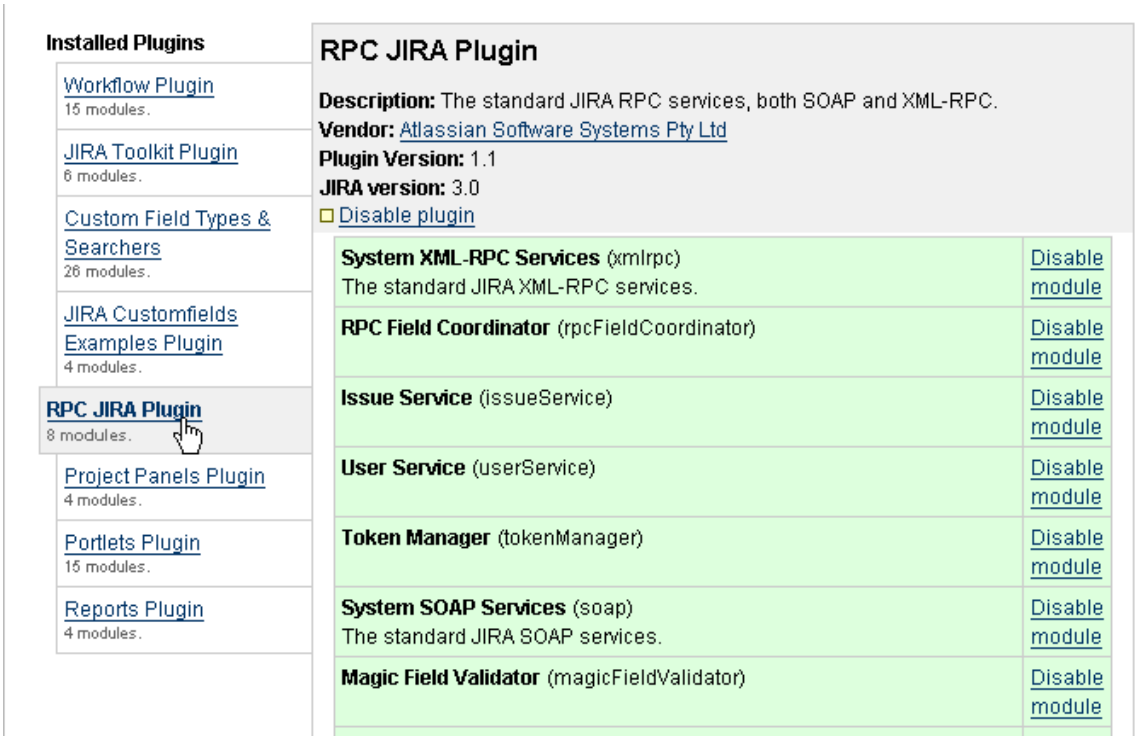
The methods exposed via XML-RPC are listed in the RPC plugin Javadoc for the XmlRpcService class. The JIRA XML-RPC API Spec has more information (though not guaranteed to be up-to-date).

To run the Java client in this tutorial, you'll need to download the Apache XML-RPC libraries and make it available in your classpath.

You should also ensure that the XML-RPC plugin has is enabled on the JIRA installation you are targeting. If you simply want to create a client to http://jira.atlassian.com/ then you can skip this step. First you need to check if the Accept Remote API Calls has been enabled in the General Configuration tab under Global Settings.



Then you need to enable the JIRA RPC Plugin as below.



If the plugin does not appear as above then your XML-RPC jar has not been properly installed. Download the jar from the repository and copy it to the `atlassian-jira/WEB-INF/lib` folder of your JIRA installation. Perform a restart and your plugin should appear

Now that your server is ready to accept remote procedure calls, we begin creating a Java XML-RPC client.

Python XML-RPC client

XML-RPC in Python is very easy. Here is a sample client that creates test issues on http://jira.atlassian.com:

```
#!/usr/bin/python

# Sample Python client accessing JIRA via XML-RPC. Methods requiring
# more than basic user-level access are commented out.
#
# Refer to the XML-RPC Javadoc to see what calls are available:
# http:
//docs.atlassian.com/software/jira/docs/api/rpc-jira-plugin/latest/com/atlassian/jira/rpc/xmlrpc/XmlRpcSe

import xmlrpclib

s = xmlrpclib.ServerProxy('http://jira.atlassian.com/rpc/xmlrpc')
#s = xmlrpclib.ServerProxy('http://192.168.0.87:8080/rpc/xmlrpc')
auth = s.jira1.login('xmlrpctester', 'xmlrpctester')

newissue = s.jira1.createIssue(auth, { 'project': 'TST', 'type': 2,
        'summary': 'Issue created via XML-RPC', 'description': 'Created with a Python client'})

print "Created %s/browse/%s" % (s.jira1.getServerInfo(auth)['baseUrl'], newissue['key'])

print "Commenting on issue.."
s.jira1.addComment(auth, newissue['key'], 'Comment added with XML-RPC')

print "Modifying issue..."
s.jira1.updateIssue(auth, newissue['key'], {
                "summary": ["[Updated] issue created via XML-RPC"],

                # Setting a custom field. The id (10010) is discoverable from
                # the database or URLs in the admin section

                "customfield_10010": ["Random text set in updateIssue method"],

                # Demonstrate setting a cascading selectlist:
                "customfield_10061": ["10098"],
                "customfield_10061_1": ["10105"],
                "components": ["10370"]

                })


print "Done!"
```

Java client

The goal of this tutorial is to create a client that makes three simple remote calls to JIRA. Here we login, retrieve the project information and then logout again. You can take a look at the full source code here (xmlrpc-2.x) or here (xmlrpc-3.x).

The first step is to configure your details.

```
public static final String JIRA_URI  = "http://jira.atlassian.com";
public static final String RPC_PATH  = "/rpc/xmlrpc";
public static final String USER_NAME = "enteryourlogin@atlassian.com";
public static final String PASSWORD  = "yourpassword";
```

All XML-RPC calls are invoked at with the path `/rpc/xmlrpc` by default. You need to configure your username and password appropriately.

```
// Initialise RPC Client
XmlRpcClient rpcClient = new XmlRpcClient(JIRA_URI + RPC_PATH);

// Login and retrieve logon token
Vector loginParams = new Vector(2);
loginParams.add(USER_NAME);
loginParams.add(PASSWORD);
String loginToken = (String) rpcClient.execute("jira1.login", loginParams);
```

Method calls to JIRA via XML-RPC need to be prefixed with "**`jira1.`**". Parameters to methods are passed as sequenced Objects in a Vector. In the above code, we log into jira.atlassian.com. We receive back a *`loginToken`* which will need to be passed to all subsequent method calls.

```
// Retrieve projects
Vector loginTokenVector = new Vector(1);
loginTokenVector.add(loginToken);
List projects = (List)rpcClient.execute("jira1.getProjectsNoSchemes", loginTokenVector);

// Print projects
for (Iterator iterator = projects.iterator(); iterator.hasNext();)
{
    Map project =  (Map) iterator.next();
    System.out.println(project.get("name") + " with lead " + project.get("lead"));
}
```

The RPC client calls the getProjectsNoSchemes() method passing the loginToken. This returns with a Vector of *projects* which are represented by `HashTable` objects. For information on what methods are available as well as what properties are available on returned projects, you'll again need to look at the JIRA XML-RPC API Spec.

```
// Log out
Boolean bool = (Boolean) rpcClient.execute("jira1.logout", loginTokenVector);
System.out.println("Logout successful: " + bool);
```

Lastly, we log out of the system, again passing the loginToken in a Vector form.

There it is! A simple client for JIRA XML-RPC. If you wish to extend or customize the JIRA XML-RPC plugin itself, you can download the latest source from the repository.

Perl Client

Here's an XMLRPC client that uses the `XMLRPC::Lite` module (distributed with ActivePerl and available for free on CPAN).

```
#!/usr/bin/perl

# toy jira perl client using XMLRPC
# logs in, creates an issue
# handles failure or prints issue fields
# logs out.

use strict;
use warnings;

use XMLRPC::Lite;
use Data::Dumper;

my $jira = XMLRPC::Lite->proxy('http://localhost:8080/jira/rpc/xmlrpc');
my $auth = $jira->call("jira1.login", "admin", "admin")->result();
my $call = $jira->call("jira1.createIssue", $auth, {
    'project' => 'CEL',
    'type' => 2,
    'summary' => 'Issue created via XMLRPC',
    'assignee' => 'admin',
    'description' => 'Created with a Perl client'});
my $fault = $call->fault();
if (defined $fault) {
    die $call->faultstring();
} else {
    print "issue created:\n";
    print Dumper($call->result());
}
$jira->call("jira1.logout", $auth);
```

`XMLRPC::Lite` is poorly documented, using it for this simple example required reading the code - it is not advised for newbie perl hackers.

See also

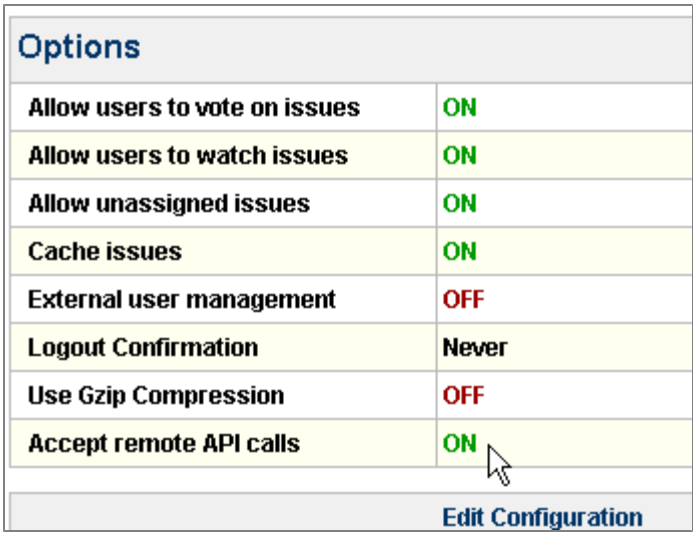
Creating a SOAP Client

Remote API (XML-RPC) Examples

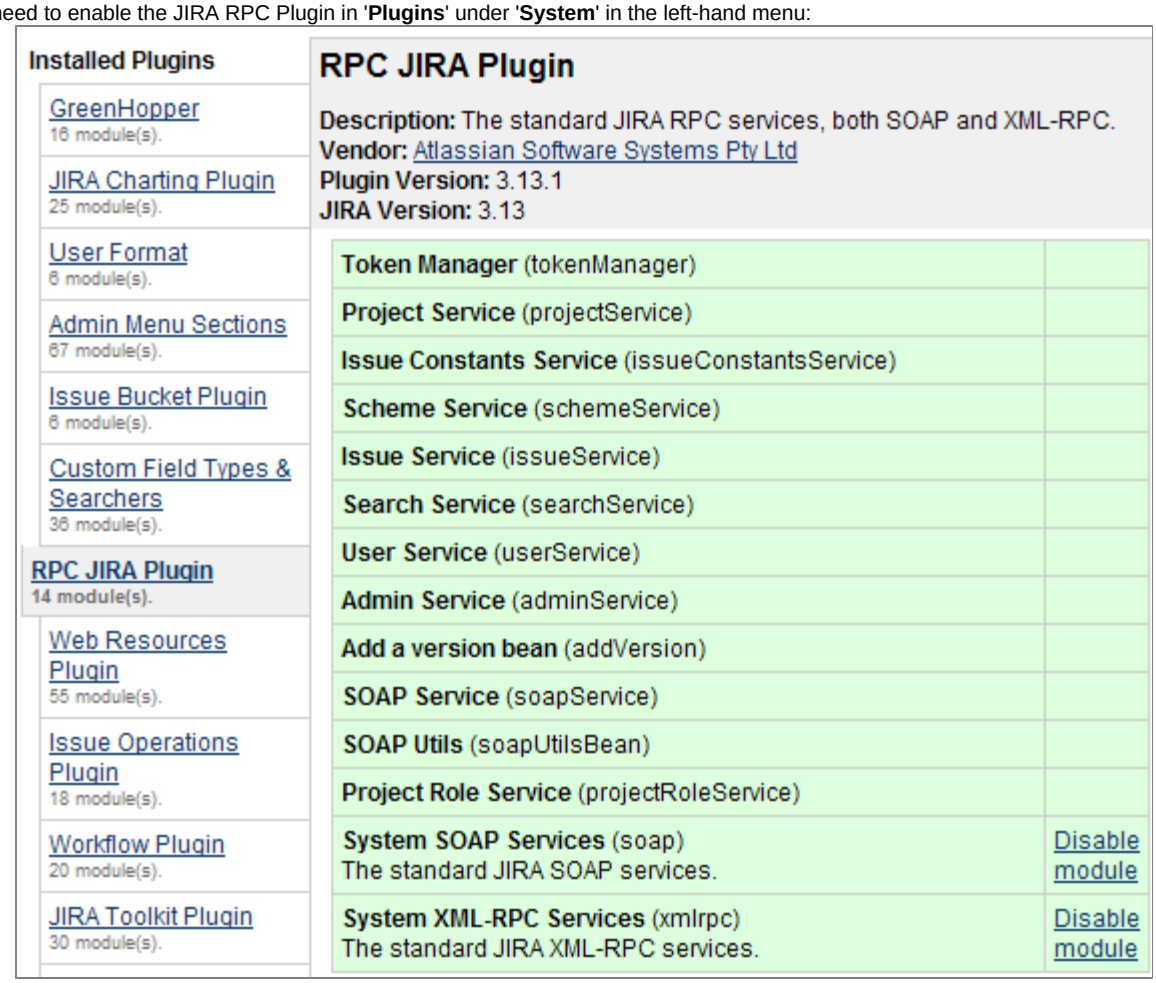
More code examples can be found in the SVN repository. You may also try looking for them in the jira-user and jira-developer mailing list (or forum).

Enabling the RPC plugin

To invoke JIRA operations remotely, you should ensure that the RPC plugin is enabled on the JIRA installation you are targeting. If you simply want to create a client to http://jira.atlassian.com/ then you can skip this step. First you need to check if the Accept Remote API Calls has been enabled in '**General Configuration**' under '**Global Settings**' in the left-hand menu:



Then you need to enable the JIRA RPC Plugin in '**Plugins**' under '**System**' in the left-hand menu:



!

> ⚠ If the plugin does not appear as above then your RPC jar has not been properly installed. Download the jar from the repository and copy it to the `atlassian-jira/WEB-INF/lib` folder of your JIRA installation. Perform a restart and your plugin should appear.

Your server should now be ready to accept remote procedure calls.

JIRA XML-RPC Overview

- Introduction
- Methods
- Data Objects
- Notes

Introduction

This page provides basic documentation on JIRA's XML-RPC capabilities. For latest methods available through the XML-RPC please refer to the latest javadoc for XmlRpcService.

> ⚠ JIRA supports both SOAP and XML-RPC. We recommend the SOAP interface as it is more complete.

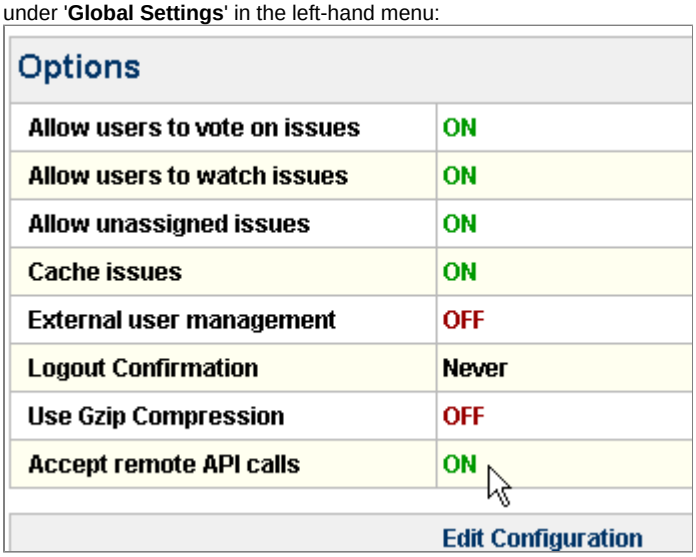
Notes:

- The URL for XML-RPC requests is http://`jira-install`/rpc/xmlrpc.
- All methods must be prefixed by `jira1` - to indicate this is version 1 of the API. We might introduce another version in the future.
- All keys in structs are case sensitive.
- All strings are passed as UTF-8, and not ASCII per the XML-RPC
- When reading the APInywhere you see the word Vector, you can interchange it with "Array" or "List" depending on what language you prefer. This is the array data type as defined in the XML-RPC spec.
- Anywhere you see the word Hashtable, you can interchange it with "Struct" or "Dictionary" or "Map" depending on what language you prefer. This is the struct data type as defined in the XML-RPC spec.
- The default XML-RPC session lifetime is 60 minutes, it will be editable in the future.

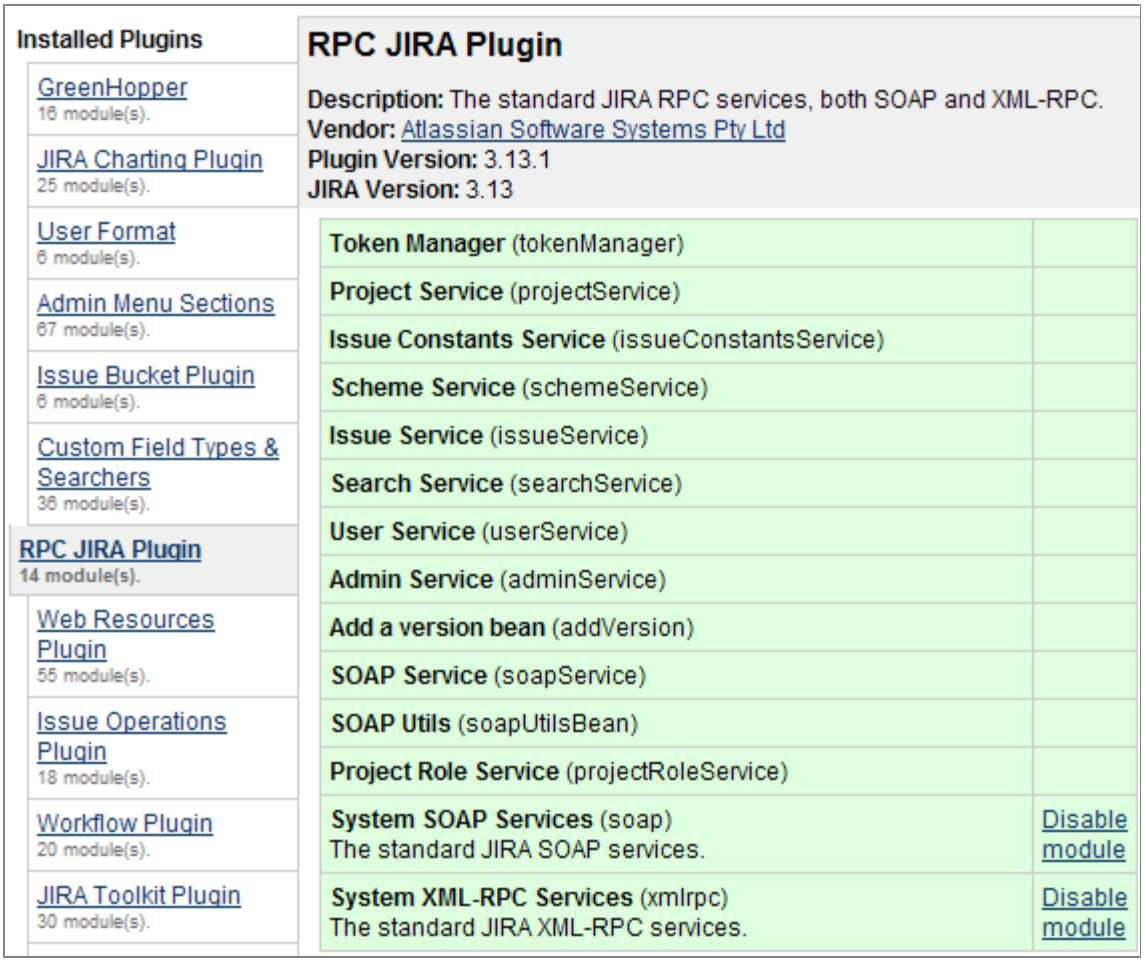
⚠ You may also wish to see the Creating a XML-RPC Client or Creating a SOAP Client if you're interested in creating a JIRA remote client.

Enable the RPC plugin

To invoke JIRA operations remotely, you should ensure that the RPC plugin is enabled on the JIRA installation you are targeting. If you simply want to create a client to http://jira.atlassian.com/ then you can skip this step. First you need to check if the Accept Remote API Calls has been enabled in '**General Configuration**' under '**Global Settings**' in the left-hand menu:



Then you need to enable the JIRA RPC Plugin in '**Plugins**' under '**System**' in the left-hand menu:



!

If the plugin does not appear as above then your RPC jar has not been properly installed. Download the jar from the repository and copy it to the `atlassian-jira/WEB-INF/lib` folder of your JIRA installation. Perform a restart and your plugin should appear.

Your server should now be ready to accept remote procedure calls.

Remote Methods

The most recent and up-to-date source of information of available exposed methods is the javadoc for the RPC plugin, specifically those on the XmlRpcService.

The javadoc will often refer to "hashtables with fields from RemoteObject". To the hashtable will contain keys that map to the fields available through reflection of the particular RemoteObject. For example, the object `RemoteVersion`, has the methods getReleaseDate(), getSequence(), isArchived() and isReleased(). This will be converted into a Hashtable with keys releaseDate, sequence, archived and released.

Data Objects

Most returned structs have a summary and a detailed form:

- The summary form is a primary key (ie project key) and a representative form (ie name)
- The detailed form will have all of the entity details as might be needed for the client.

Unless otherwise specified, all returned structs are in detailed form.

Project

Key	Type	Value
id	String	the id of the project

key	String	the project key
name	String	the name of the project
url	String	the url to view this project online
projectUrl	String	the url of this project in your organisation (ie not a JIRA URL)
lead	String	the username of the project lead
description	String	a description of this project

Component

Key	Type	Value
id	String	the id of the component
name	String	the name of the component

Version

Key	Type	Value
id	String	the id of the version
name	String	the name of the version
released	boolean	whether or not this version is released
archived	boolean	whether or not this version is archived

IssueType / Status / Resolution

Key	Type	Value
id	String	the id of this constant
name	String	the name of the constant
description	String	the description of this constant
icon	String	the URL to retrieve the icon of this constant

Priority

Key	Type	Value
id	String	the id of this constant
name	String	the name of the constant
description	String	the description of this constant
icon	String	the URL to retrieve the icon of this constant
colour	String	the colour of this constant

Filter

Key	Type	Value
id	String	the id of this filter
name	String	the name of the filter
description	String	the description of this filter
author	String	the username of this filter's owner
project	String	the id of the project this search relates to (null if the search is across projects)

xml	String	a complete XML representation of this search request - I don't recommend you use this for now, it's complex

User

Key	Type	Value
name	String	the username of this user
fullname	String	the full name of this user
email	String	the email address of this user

Frequently Asked RPC Questions and Known Issues

This page contains frequently asked questions and known issues about RPC, in relation to JIRA.

On this page:

- Common Problems
  - Couldn't build RPC Plugin due to NullPointerException
  - Getting InfrastructureException or ClassCastException upon invocation
- SOAP
  - Converting to type as defined in WSDL
  - Changing the authentication token timeout value
  - Providing schemes to method's arguments
  - Could not update an issue with a RemoteIssue object
  - Setting the value of cascading select customfield
  - Could not invoke methods due to session invalidation
  - Getting fixed targetNamespace
  - Having concurrency problem with multiple issue update
  - Could not invoke user-group methods due to deserialisation or invalid element problems
  - Using the addActorsToProjectRole method
  - Could not upload attachments to an issue due to OutOfMemoryError
  - Getting content-type complaint with .NET client
  - Setting the reporter value upon issue creation
  - Other Known RPC Issues
    - Logging work to an issue
    - Updating user information
    - Deleting version
    - Creating subtask
    - Changing status and resolution of an issue
    - Posting comment while progressing workflow
    - Setting comment "Viewable By" security
    - Getting required and available fields for issue creation
- XML-RPC
  - Could not create and update issue with same Dictionary
  - Setting the value of cascading select customfield
- Need more help?

Common Problems

Couldn't build RPC Plugin due to NullPointerException

If you are getting a `NullPointerException` and a stack trace which looks similar with the one described this document, you may need to disable unit testing in `project.xml` for a smooth build, as mentioned in the document referenced previously.

Getting InfrastructureException or ClassCastException upon invocation

If you are getting a `ClassCastException` or `com.atlassian.jira.InfrastructureException` upon invocation, you may have provided the incorrect data types for the arguments. To resolve this problem, please correct the data types of the arguments accordingly.

If this is not the case and this error is occurring, then it is highly likely that autotyping is the culprit. To resolve this problem, you will need to disable autotyping or alternatively, explicitly declare the data types for the arguments.

SOAP

Converting to type as defined in WSDL

You may occasionally encounter problems mapping the correct data type of your programming language to the one defined in WSDL. If you are running into difficulties related to data mapping, we recommend that you consult the user guide or API documentation of the client library you are using, e.g. WSDL<->Java mapping, Python's SOAPpy, etc, for assistance.

Changing the authentication token timeout value

If you wish to change the timeout value of the authentication token (which is obtained once you've logged in), you will need to update the value for the
`DEFAULT_TIMEOUT` constant in the current `com.atlassian.jira.rpc.auth.TokenManager` implementation, i.e. `com.atlassian.jira.rpc.auth.TokenManagerImpl`.

Providing schemes to method's arguments

Users are often unsure whether they should construct or retrieve the scheme object(s) which will be passed to an argument. For your understanding, internally, the RPC plugin recognises just the `id` of the scheme, hence the `name`, `type`, `description`, etc. are usually insignificant.

Could not update an issue with a RemoteIssue object

Unfortunately, you can only **create an issue** with a RemoteIssue object. You cannot **update an issue** with a RemoteIssue object. The `createIssue()` method expects a `RemoteIssue` object as its argument, however the `updateIssue()` method expects an array of `RemoteFieldValue` objects (as per the WDSL).

Setting the value of cascading select customfield

Setting the value of a cascading select custom field can be complicated, as the create issue and update issue scenarios are different. Please refer to the points below for assistance in setting up your cascading select custom field correctly:

- While creating an issue, the parent field is set by providing the `customfieldId` and `values` (single-element array with option's id) and the child field is set by providing the `customfieldId`, `key` (as `1`) and `values`.
- While updating an issue, the parent field is set by providing value (a single-element array with option's id) to the `customfieldId` and the child field is set by providing value to the `customfieldId:key` (`key` as `1`, e.g. `customfield_10000:1`).

The same logic applies to XML-RPC, please refer to this example.

Could not invoke methods due to session invalidation

A known issue exists where methods may be prevented from being invoked due to session invalidation, i.e. an `IllegalStateException` occurs when trying to invoke the RPC methods:

```
java.lang.IllegalStateException: getAttribute: Session already invalidated
 at org.apache.axis.message.SOAPFaultBuilder.createFault(SOAPFaultBuilder.java:221)
 at org.apache.axis.message.SOAPFaultBuilder.endElement(SOAPFaultBuilder.java:128)
 at org.apache.axis.encoding.DeserializationContext.endElement(DeserializationContext.java:1087)
 at
com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.endElement(AbstractSAXParser.java:633)
at
com.sun.org.apache.xerces.internal.impl.XMLNSDocumentScannerImpl.scanEndElement(XMLNSDocumentScanne
at
com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.di
at
com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanDocument(XMLDocumentFrag
at
com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(XML11Configuration.java:834)
 at
com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(XML11Configuration.java:764)
 at com.sun.org.apache.xerces.internal.parsers.XMLParser.parse(XMLParser.java:148)
 at
com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.parse(AbstractSAXParser.java:1242)
 at javax.xml.parsers.SAXParser.parse(SAXParser.java:375)
 at org.apache.axis.encoding.DeserializationContext.parse(DeserializationContext.java:227)
 at org.apache.axis.SOAPPart.getAsSOAPEnvelope(SOAPPart.java:696)
 at org.apache.axis.Message.getSOAPEnvelope(Message.java:424)
 at org.apache.axis.handlers.soap.MustUnderstandChecker.invoke(MustUnderstandChecker.java:62)
 at org.apache.axis.client.AxisClient.invoke(AxisClient.java:206)
 at org.apache.axis.client.Call.invokeEngine(Call.java:2754)
 at org.apache.axis.client.Call.invoke(Call.java:2737)
 at org.apache.axis.client.Call.invoke(Call.java:2413)
 at org.apache.axis.client.Call.invoke(Call.java:2336)
 at org.apache.axis.client.Call.invoke(Call.java:1793)
 at
com.atlassian.jira.rpc.soapclient.JirasoapserviceV2SoapBindingStub.getFieldsForAction(Jirasoapservi
```

Please refer to the bug report being tracked at JRA-8009. We would appreciate any comments added to this issue, if you encounter this problem.

Getting fixed targetNamespace

Getting a fixed targetNamespace is a known issue, which is being tracked at JRA-10849. The `targetNamespace` of the WSDL is defined by the first request to it.

Having concurrency problem with multiple issue update

Constantly or simultaneously updating an issue causes data inconsistency of the issue. This is a known issue which is currently being tracked at JRA-11382.

Could not invoke user-group methods due to deserialisation or invalid element problems

If you are using Python (SOAPpy) and encountering deserialisation or invalid element problems while invoking the `addUserToGroup()` method, e.g.

```
<Fault soapenv:Server.userException: org.xml.sax.SAXException: No deserializer defined for array
type {http://soapinterop.org/xsd}SOAPStruct:
<SOAPpy.Types.structType detail at 1085258540>: {'hostname': 'atlassian01.contegix.com',
'faultData': <SOAPpy.Types.structType faultData at
1084456684>: {'exception': None, 'message': 'No deserializer defined for array type
{http://soapinterop.org/xsd}SOAPStruct'}}>
Traceback (most recent call last):
File "./soapclient.py", line 58, in ?
soap.addUserToGroup(auth, group, user)
```

```
SOAPpy.Types.faultType: <Fault soapenv:Server.userException: org.xml.sax.SAXException: Invalid
element in com.atlassian.jira.rpc.soap.beans.
RemoteGroup - email: <SOAPpy.Types.structType detail at 29991600>: {'hostname': 'bruwa116',
'faultData': <SOAPpy.Types.structType faultData
at 29991680>: {'exception': None, 'message': 'Invalid element in
com.atlassian.jira.rpc.soap.beans.RemoteGroup - email'}}>
```

you should be able to resolve this by following the workarounds described in these issues — JRA-7920 and JRA-7971

Using the addActorsToProjectRole method

If you are having problems using the `addActorsToProjectRole` method, you may find the following advice helpful:
The `JiraSoapService.addActorsToProjectRole()` method internally invokes the `ProjectRoleService.addActorsToProjectRole()` method. You must provide either an array of users or groups based on the actorType value, i.e. UserRoleActor.TYPE or GroupRoleActor.TYPE.

For example,

```
jiraSoapService.addActorsToProjectRole(token, new String[]{"admin", "foo", "bar"}, projectRole,
project, UserRoleActor.TYPE);
```

Could not upload attachments to an issue due to OutOfMemoryError

If your Java (Axis) SOAP client is throwing an `OutOfMemoryError` while attaching files to an issue, Axis 1.3 may be the cause. This known issue has been commonly reported by users and is being tracked at JRA-11693.

Getting content-type complaint with .NET client

This seems to be the problem with the .NET web service library. We would appreciate your contributions to tracking issue, JRA-11515, if you encounter a similar problem.

Setting the reporter value upon issue creation

If you are not able to set the reporter value (to someone other than the current SOAP client) even though it is provided, you may be using the old RPC plugin (3.6.x and older). This improvement (JRA-8794) is delivered in the latest version (3.7.x onwards).

Other Known RPC Issues

Unfortunately there are a number of known RPC issues, which are currently unsupported. These have already been raised in our issue tracker and are listed below.

If you would like to try manually patching or customising the RPC plugin to address these issues yourself, the source code of the RPC plugin is freely available for you to modify.

Logging work to an issue

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-7260 with your comments and vote.

Updating user information

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-11252 with your comments and vote.

Deleting version

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-11532 with your comments and vote.

Creating subtask

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-6896 with your comments and vote.

Changing status and resolution of an issue

The only way to set the status and resolution of an issue is by invoking the `progressWorkflowAction()` method. Feel free contribute to the issue at JRA-10472 with your comments and vote.

Posting comment while progressing workflow

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-11278 with your comments and vote.

Setting comment "Viewable By" security

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-11278 with your comments and vote.

Getting required and available fields for issue creation

Unfortunately, this is not supported at the moment. Feel free contribute to the issue at JRA-11597 with your comments and vote.

XML-RPC

Please take note that a number of the frequently asked XML-RPC questions are similar to the respective SOAP questions above, as some of the functions of the XML-RPC service are underpinned by the SOAP service. Hence, the questions may not be mentioned again below.

Could not create and update issue with same Dictionary

If you are running into problems creating and updating issues with the same Directory, you may be specifying the arguments incorrectly. Internally, JIRA's XmlRpcService uses SOAP to create and update an issue. Due to this reason, the Dictionary (or Hashtable, or Map) argument in `createIssue()` and `updateIssue()` are different as explained above.

This Perl XMLRPC::Lite example demonstrates how an issue is created and updated:

```
# Create an issue with a RemoteIssue structure

$jira->call("jira1.createIssue", $auth, {
  "project" => "MYC",
  "type" => 1,
  "reporter" => "admin",
  "assignee" => "admin",
  "summary" => "Issue created via Perl XMLRPC :)",
  "customFieldValues" => [
    {"customfieldId" => "customfield_10000",
     "values" => [SOAP::Data->type(string => "10000")]
    },
    {"customfieldId" => "customfield_10000",
     "key" => "1",
     "values" => [SOAP::Data->type(string => "10002")]
    }
  ]
});

# Update an issue with field-values pairs

$jira->call("jira1.updateIssue", $auth, "MYC-3",
  {"customfield_10000" => [SOAP::Data->type(string => "10000")],
   "customfield_10000:1" => [SOAP::Data->type(string => "10002")]
  }
);
```

Setting the value of cascading select customfield

Please refer to the explanation in SOAP section.

Need more help?

If you can't find an answer to your question on this page, you may want to try searching the Atlassian forums and Issue Tracker. These two resources often contain helpful discussion on RPC topics that may solve your problem.

The following documents may also help answer your question:

- Creating a SOAP Client
- Creating a XML-RPC Client

Modifying JIRA Templates and JSPs

If you wish to change JIRA's behaviour by modifying the JSPs or Velocity templates, you can do so simply by changing files in the JIRA distribution. JIRA will pick up changes to its JSPs the next time that page is displayed: restarting isn't necessary. On the other hand, you will need to restart JIRA in order to see changes to the Velocity templates.

However, keep in mind that the next time you upgrade JIRA – or need a new installation for any reason – you will have to manually copy any changes you have made to the JSPs or tempates into the new installation of JIRA. If the JSPs or templates have changed in the newer version, you will have to port your customization into them.

- Adding Custom Fields to Email
- Adding Custom Field to Issue Summary
- Adding JavaScript to all pages E.g. Google Analytics
- Contents of the Velocity Context
- Customising interface based on user's role

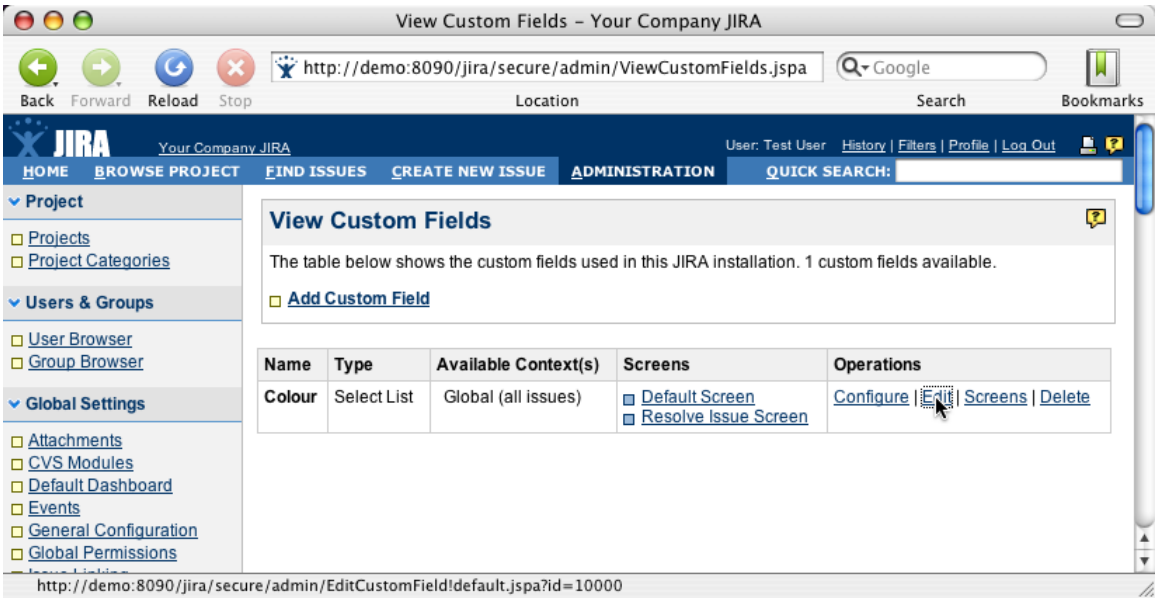
- Customising JIRA Excel Output
- Customizing text
- Mapping custom events to new email templates
- Tutorial - How to Create a Custom Release Notes Template Containing Release Comments
- Velocity Context for Email Templates

Adding Custom Fields to Email

Printing a custom field in a JIRA email is fairly easy in 3.6 and above.

Step 1: find the custom field Id

This can be discovered from the database, or by examining the URLs of custom field pages. For example, here the id of the custom field is 10000:



Step 2: edit the velocity template

Locate the Velocity template of the email type you wish to modify. For instance, you may want to modify the 'issue created' template, as well as the template reused in 'issue commented':

```
atlassian-jira/WEB-INF/classes/templates/email/text/issuecreated.vm
atlassian-jira/WEB-INF/classes/templates/email/text/includes/issuesummary.vm
```

Add the following snippet where you want it to appear in the file:

```
#if ($issue.getCustomFieldValue("customfield_10000"))
$stringUtils.leftPad($issue.getCustomField("customfield_10000").name, $padSize):
$issue.getCustomFieldValue("customfield_10000")
#end
```

*The javadoc for **$stringUtils.leftPad** can be found here.*

In `atlassian-jira/WEB-INF/classes/templates/email/text/includes/issuesummary.vm` you will want:

```
#if ($issue.getCustomFieldValue("customfield_10000"))
>$stringUtils.leftPad($issue.getCustomField("customfield_10000").name, $padSize):
$issue.getCustomFieldValue("customfield_10000")
#end
```

Note: **you need to change the custom field id** to the id observed in step 1. Here it is 10000 - yours will probably be different.

If you wish to iterate over all related custom fields you can use the following example:

```
#foreach ($value in $customFieldManager.getCustomFieldObjects($issue))
>$stringUtils.leftPad($value.getName(), $padSize): $!value.getValueFromIssue($issue)
#end
```

Restart JIRA.

To make the changes take effect, restart JIRA. If you wish to avoid the continual restarts during testing, edit `atlassian-jira/WEB-INF/classes/velocity.properties` and alter this section as the comment says:

```
# To enable autoreloading, set cache to false and uncomment the autoreload line
class.resource.loader.cache=true
#velocimacro.library.autoreload=true
```

In 3.1 (and possibly later)

The following velocity code allows you to print the custom field values in an email:

This code was contributed by a customer, and will almost certainly break in future versions of JIRA! As you can see it is a lot more brittle and uglier than the 3.6 equivalent.

```
#foreach ($cfv in ${issue.getRelated("ChildCustomFieldValue")})
 #set ($cf = ${cfv.getRelatedOne("ParentCustomField")})
  #if (${cfv.getString("stringvalue")} != " ")
    ${cf.getString("name")} : ${cfv.getString("stringvalue")}
  #elseif (${cfv.getString("textvalue")} != " ")
    ${cf.getString("name")} : ${cfv.getString("textvalue")}
  #elseif (${cfv.getString("numbervalue")} != " ")
    ${cf.getString("name")} : ${cfv.getDouble("numbervalue")}
  #elseif (${cfv.getString("datevalue")} != " ")
    ${cf.getString("name")} : ${cfv.getDate("datevalue")}
  #end
#end
```

Adding Custom Field to Issue Summary

On the View Issue page some of the issue's details are displayed in the top-left corner. With some minor customisation it is possible to display a custom field their as well. To do this please add the following code to the `includes/panels/issue/view_details.jsp` file found under the JIRA web application:

```
<webwork:iterator value="/fieldScreenRenderTabs" status="'status'">
    <%-- Show tab's fields --%>
    <webwork:iterator value="./fieldScreenRenderLayoutItems">
        <webwork:property value="./orderableField">
            <%-- don't display custom fields with no values --%>
            <webwork:if test="./id == 'customfield_<id>' && ./value(/issueObject) != null &&
./customFieldType/descriptor/viewTemplateExists != false">
                <tr id="rowFor<webwork:property value="./id" />">
                    <td valign=top><b><webwork:property value="name" />:</b></td>
                    <td valign=top>
                        <webwork:property value="/customFieldHtml(../fieldLayoutItem,., /issueObject)"
escape="false" />
                    </td>
                </tr>
            </webwork:if>
        </webwork:property>
    </webwork:iterator>
</webwork:iterator>
```

Note, that the <id> in the above code needs to be replaced with the valid numeric id of the custom field you wish to display. For example, 'customfield_10000'.

On newer editions of JIRA (3.7+), *view_details.jsp* is no longer used and has been supplanted by velocity templates. The file you'll want to modify will be: *WEB-INF/classes/templates/jira/issue/summary/issuesummary.vm*. The syntax for the changes will be different from above. You can see a community contributed example on our forums here.

Adding JavaScript to all pages E.g. Google Analytics

Some people wish to add some javascript to every page within JIRA. One such use case is Goolge Analytics.

To implement this you need to add the following javascript to all your pages:

```
<script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
</script>
<script type="text/javascript">
_uacct = "UA-xxxxx";
urchinTracker();
</script>
```

The easiest way to do this would be (as of 3.4.1) to add this script into the Annoucement Banner which is displayed on nearly every page. This will display an empty red box, but you can simply comment out the style in the CSS. This will take care of about 90 - 95% of pages. This can be edited through the Adminitration section.

Else, you can manually add it to the file: **`stylesheettag.jsp`** This is included on every page.

Contents of the Velocity Context

This is the listing of the contents of the Velocity Context used to process web templates.

For JIRA 3.6.x and later - please refer to the JIRA 3.6.x guide for details on the velocity context used to process email templates.

Velocity variable	JIRA Object/Value	Description	As of Version	Condition
constantsManager	com.atlassian.jira.config.ConstantsManager	Manager for issue types, statuses, priorities and resolutions.	3.4	always prese
projectManager	com.atlassian.jira.project.ProjectManager	Implementations of this interface are responsible for all management of project entities within JIRA.	3.4	always prese
customFieldManager	com.atlassian.jira.issue.CustomFieldManager	Functions for working with CustomFields	3.5	always prese
applicationProperties	com.atlassian.jira.config.properties.ApplicationProperties	provides access to JIRA properties stored in the DB	3.4	always prese
jirautils	com.atlassian.jira.util.JiraUtils	Miscellaneous utility methods.	3.4	always prese
jirakeyutils	com.atlassian.jira.util.JiraKeyUtils	utilities to determine the validity of JIRA project/issue keys	3.4	always prese
buildutils	com.atlassian.jira.util.BuildUtils	provides information on the running version of JIRA	3.4	always prese
velocityhelper	com.atlassian.jira.util.JiraVelocityHelper	A simple class store methods we want to expose to velocity templates	3.4	always prese
userutils	com.atlassian.core.user.UserUtils	A utility class for operating on users.	3.4	always prese

textutils	com.opensymphony.util.TextUtils	lots of utility methods for manipulating text	3.4	always prese
params	java.util.Map	parameters of the IssueEvent that triggered this email notification	3.4	always prese
issue	org.ofbiz.core.entity.GenericValue	a GenericValue representing the issue which triggered this email notification	3.4	always prese
issueObject	com.atlassian.jira.issue.MutableIssue	an Issue object representing the issue which triggered this email notification	3.5.2	always prese
remoteUser	com.opensymphony.user.User	the logged in user if they exist	3.4	remoteUser !
renderedDescription	java.lang.String	the rendered description field, it a renderer has been specified	3.4	renderer spe issue and fiel
renderedEnvironment	java.lang.String	the rendered environment field, it a renderer has been specified	3.4	renderer spe issue and fiel
timeoriginalestimate	java.lang.String	The DateUtils.getDurationPretty value of timeoriginalestimate from issue or "None" if null	3.4	always prese
timeestimate	java.lang.String	The DateUtils.getDurationPretty value of timeestimate from issue or "None" if null	3.4	always prese
timespent	java.lang.String	The DateUtils.getDurationPretty value of timespent from issue or "None" if null	3.4	always prese
duedate	java.sql.Timestamp	Duedate from the issue	3.4	always prese
versions	List(org.ofbiz.core.entity.GenericValue)	A list of GenericValues representing the Affected Versions of the issue	3.4	always prese
fixfors	List(org.ofbiz.core.entity.GenericValue)	A list of GenericValues representing the Fix Versions of the issue	3.4	always prese
components	List(org.ofbiz.core.entity.GenericValue)	A list of GenericValues representing the Components of the issue	3.4	always prese
assignee	com.opensymphony.user.User	The assignee of the issue	3.4	assignee != r
reporter	com.opensymphony.user.User	The reporter of the issue	3.4	reporter != nu
renderedComment	java.lang.String	the rendered comment field, it a renderer has been specified. The comment is the comment associated with the change	3.4	renderer spe issue and fiel comment crea
comment	org.ofbiz.core.entity.GenericValue	generic value representing the comment associated with the change	3.4	comment cre
commentauthor	com.opensymphony.user.User	the author of the comment	3.4	comment cre
timelogged	java.lang.String	The DateUtils.getDurationPretty value of timelogged from issue or "None" if null	3.4	always prese
changelog	org.ofbiz.core.entity.GenericValue	generic value representing the changelog associated with the change	3.4	changelog !=
changelogauthor	com.opensymphony.user.User	The change log author of the issue	3.4	changelog !=

visibilitylevel	java.lang.String	The security level of the comment	3.4	comment cre
i18n	com.atlassian.jira.util.I18nHelper	Bean that searches for i18n text in JiraWebActionSupport.properties	3.4	always prese
dateutils	`com.atlassian.core.util.DateUtils`	methods for working with dates	3.4	always prese
attachments	List(org.ofbiz.core.entity.GenericValue)	list of generic values that represents the attachments associated with the issue	3.4	always prese
security	org.ofbiz.core.entity.GenericValue	generic value representing the security level, if any, associated with this issue	3.4	if(isEnterpris
mailPluginsHelper	com.atlassian.jira.mail.JiraMailPluginsHelperImpl	provides access to isPluginModuleEnabled()	3.7	always prese

Customising interface based on user's role

Sometimes it is useful to remove certain elements from JIRA's user interface if the user does not belong to a certain group. Most of the time this can be achieved by editing certain JSP / Velocity files. The JSP files are easy to edit as they are text files, and one does not need access to JIRA's source code.

For JIRA 3.5+

In velocity, you can hide certain elements from the UI by surrounding the relevant code with:

```
#if ($authcontext.user.inGroup('jira-administrators'))
...
#end
```

So to hide the security level you need to edit the file `WEB-INF\classes\templates\jira\issue\field\comment-edit.vm` and end up with something like this.

```
#if ($authcontext.user.inGroup('jira-administrators'))
    #if ($commentLevels && !$commentLevels.isEmpty())
        #controlHeader ($action 'commentLevel' $i18n.getText('comment.viewable') false
$displayParameters.get('noHeader'))
        ...
        #controlFooter ($action '' $displayParameters.get('noHeader'))
    #end
#end
```

You need to update **both** places in the file and then restart JIRA.

For versions prior to 3.5

To hide certain elements from the UI one has to surround the relevant code with:

```
<webwork:if test="/remoteUser/inGroup('jira-administrators') == true">
...
</webwork:if>
```

If you would like to allow only members of the jira-administrators group to be able to set a *level* for a comment (restrict comment's visibility), just surround the code that creates the ' Viewable By:' drop down box.

So that the code looks something like:

```
<webwork:if test="/remoteUser/inGroup('jira-administrators') == true">
    <webwork:if test="commentLevels/size > 0">
        <ui:select label="text('comment.add.viewableby.label')" name="'commentLevel'" list=
"commentLevels"
              listKey="'.'" listValue="'.'">
            <ui:param name="'headerrow'" value="text('comment.constants.allusers')" />
            <ui:param name="'headervalue'" value="''" />
        </ui:select>
    </webwork:if>
</webwork:if>
```

You will need to do this in the following files that are located under the JIRA web application:

- secure/views/issue/addcomment.jsp
- includes/panels/updateissue_comment.jsp
- secure/views/issue/viewissue.jsp

If you make changes such as this, you will need to **remember** to port them to a new version of JIRA when you upgrade

Customising JIRA Excel Output

Overview

Microsoft Office 2000 supports Hypertext Markup Language (HTML) as a native file format. Allowing Microsoft Excel to save and open HTML files as spreadsheets with all the document formatting preserved. The document can be formatted through a combination of HTML, XML and Cascading Style Sheets (CSS).

The format of the excel output from JIRA can be modified by editing the template file *navigator-excel-current.jsp* (located within *src/webapp/secure/views/navigator/* of your JIRA home folder). Editing this file is essentially editing a HTML template file with tables and styles.

By default the template file *navigator-excel-current.jsp* should contain:

navigator-excel-current.jsp

```
...
<style>
@page
{
mso-page-orientation:landscape;
}

br
{
mso-data-placement: same-cell;
}
</style>
...
```

The above formats the default excel output to have the page orientation as landscape, and multi-lined text will be kept within one cell. Further examples are provided below.

Microsoft Office HTML and XML Reference

Microsoft Office HTML and XML reference contains a thorough listing of available styles and details on how to use them. The reference manual is available for download from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnoffxml/html/ofxml2k.asp

Examples

Example 1 - Customising Page Setup

Excel uses CSS and @page definitions to store some page setup settings and XML to store others. Below is a sample to setup the page with the following document format respectively.

- Set page orientation as landscape (default is portrait)
- Set the top, right, bottom and left margins as .75in .25in .75in .25in respectively
- Set header margin as .5in
- Set footer margin as .4in
- Set a custom footer displaying page number, date and time all centered.

Details of the default values, and special symbols to use for styles can be found in the Microsoft Reference Manual.

```
<style>
...
@page
{
mso-page-orientation: landscape;
margin:.75in .25in .75in .25in;
mso-header-margin:.5in;
mso-footer-margin:.4in;
mso-footer-data:"&C Page &P of &N Date: &D Time: &T";
<!-- other @page styles ... -->
}
</style>
```

Example 2 - Customising Cell Formatting

Both CSS and HTML are used to customise the cells. A global style that is applied to all table cells can be applied using TD definition.

```
<style>
...
TD
{
mso-numberformat:general;
text-align: left;
vertical-align: middle;
color:red
<!-- other TD styles ... -->
}
</style>
```

The above applies the following styles to all table cells respectively:

- Sets the number format of all table cells as *general*
- Horizontally aligns the text in the cell to the left
- Vertically aligns the text in the cell to the middle
- Set the text color to be red

Individual or group of cells may have other styles applied by adding a class value into the appropriate HTML tags

Customizing text

JIRA, and most Java applications in general, determine what internationalization package to use, by inheriting the System Locale, which is provided by the underlying operating system to the JAVA JVM. The language_default.jar is only chosen when the inherited locale does not map to any of the included languages. This means that if you modify language_default.jar *instead* of your specific language pack, (say, language_en_UK.jar), then you will need to remove language_en_UK.jar for JIRA to then default back to the modified, language_default.jar.

Most user-accessible pages in JIRA have been 'internationalized', meaning the pages will display in the user's preferred language. All text snippets on JIRA pages are stored centrally in 'properties' files, separate from the pages they occur in. There is one set of properties files for each language, and one default (English) set.

This centralized storage of text snippets makes it easy to change some text in JIRA to anything you like. This page describes how it can be done.

Say we wish to reduce the verbosity of the 'view issue' page's text, from:

- Assign this issue
- Attach file to this issue
- Attach screenshot to this issue
- Comment on this issue
- Delete this issue
- Edit this issue
- Link this issue to another issue
- Move this issue to another project

to:

- Assign
- Attach file
- Attach screenshot
- Add Comment
- Delete
- Edit
- Link to another issue
- Move to another project

Properties files are stored in a packaged (jar) format in `atlassian-jira/WEB-INF/lib`:

```
[atlassian-jira-professional-2.6.1-standalone-tomcat ~/atlassian-jira/WEB-INF/lib]$ ls \-1 language\*
language_da_DK.jar
language_de_DE.jar
language_default.jar
language_en_UK.jar
language_pt_BR.jar
language_ru_RU.jar
```

We wish to extract the contents of one of these for editing. To do this, go to the `atlassian-jira/WEB-INF/classes` directory, and unzip the relevant file:

jar files are actually zip files, so you can use a tool like WinZip to extract their contents. Here we use the 'jar' command that comes with Java

```
[atlassian-jira-professional-2.6.1-standalone-tomcat ~/atlassian-jira/WEB-INF/lib]$ cd ../classes
[atlassian-jira-professional-2.6.1-standalone-tomcat ~/atlassian-jira/WEB-INF/classes]$ jar xvf
../lib/language_default.jar
 created: META-INF/
extracted: META-INF/MANIFEST.MF
 inflating: language-descriptor.xml
 created: com/
 created: com/atlassian/
 created: com/atlassian/jira/
 created: com/atlassian/jira/web/
 created: com/atlassian/jira/web/action/
 created: com/atlassian/jira/web/action/filter/
 created: com/atlassian/jira/web/action/issue/
 created: com/atlassian/jira/web/action/portal/
 created: com/atlassian/jira/web/action/setup/
 created: com/atlassian/jira/web/action/user/
extracted: com/atlassian/jira/web/action/filter/ManageFilters.properties
extracted: com/atlassian/jira/web/action/AbstractViewIssueColumns.properties
extracted: com/atlassian/jira/web/action/Dashboard.properties
extracted: com/atlassian/jira/web/action/IssueActionSupport.properties
extracted: com/atlassian/jira/web/action/JiraWebActionSupport.properties
extracted: com/atlassian/jira/web/action/issue/AbstractCommentableIssue.properties
extracted: com/atlassian/jira/web/action/issue/AbstractIssueSelectAction.properties
extracted: com/atlassian/jira/web/action/issue/AddComment.properties
extracted: com/atlassian/jira/web/action/issue/AssignIssue.properties
extracted: com/atlassian/jira/web/action/issue/AttachFile.properties
extracted: com/atlassian/jira/web/action/issue/CreateIssue.properties
extracted: com/atlassian/jira/web/action/issue/DeleteIssue.properties
extracted: com/atlassian/jira/web/action/issue/EditIssue.properties
extracted: com/atlassian/jira/web/action/issue/IssueNavigator.properties
extracted: com/atlassian/jira/web/action/issue/LinkExistingIssue.properties
extracted: com/atlassian/jira/web/action/issue/LogWork.properties
extracted: com/atlassian/jira/web/action/issue/MoveIssue.properties
extracted: com/atlassian/jira/web/action/issue/ResolveIssue.properties
extracted: com/atlassian/jira/web/action/issue/ManageAttachments.properties
extracted: com/atlassian/jira/web/action/portal/AbstractConfigurePortal.properties
extracted: com/atlassian/jira/web/action/setup/AbstractSetupAction.properties
extracted: com/atlassian/jira/web/action/setup/Setup.properties
extracted: com/atlassian/jira/web/action/setup/Setup2.properties
extracted: com/atlassian/jira/web/action/setup/Setup3.properties
extracted: com/atlassian/jira/web/action/setup/Setup2Existing.properties
extracted: com/atlassian/jira/web/action/user/AddPortlet.properties
extracted: com/atlassian/jira/web/action/user/ChangePassword.properties
extracted: com/atlassian/jira/web/action/user/ConfigurePortal.properties
extracted: com/atlassian/jira/web/action/user/EditProfile.properties
extracted: com/atlassian/jira/web/action/user/UserVotes.properties
extracted: com/atlassian/jira/web/action/user/UserWatches.properties
extracted: com/atlassian/jira/web/action/user/ViewProfile.properties
[atlassian-jira-professional-2.6.1-standalone-tomcat ~/atlassian-jira/WEB-INF/classes]$
```

* If you are using the **WAR/Webapp distribution** of JIRA, you should extract the jar from `webapp/WEB-INF/lib` to `edit-webapp/WEB-INF/classes` and **remove** the `webapp/WEB-INF/lib/language_default.jar` and (if using English) `webapp/WEB-INF/lib/language_en_UK.jar` (using the correct jar for your language).

- If you are using the **standalone distribution** of JIRA and you are running JIRA in an **English locale**, you should extract the jar from `webapp/WEB-INF/lib` to `edit-webapp/WEB-INF/classes` and **remove** the `webapp/WEB-INF/lib/language_default.jar` and `webapp/WEB-INF/lib/language_en_UK.jar`
- If you are using the **standalone distribution** of JIRA and you are running JIRA in an **non-English locale**, you should extract the jar from `webapp/WEB-INF/lib` to `edit-webapp/WEB-INF/classes` and **remove** the `webapp/WEB-INF/lib/language_default.jar` only. Do not remove `webapp/WEB-INF/lib/language_en_UK.jar`, otherwise your users will not be able to select English as a language (See JRA-8266 for further details). You will need to update the text you wish to replace in the properties files (described in the next step) for both the `language_default.jar` and the `language_en_UK.jar`, for your changes to be reflected in JIRA.

Now search for the text you wish to replace. You will find most strings defined in `com/atlassian/jira/web/action/JiraWebActionSupport.properties`. The ones we are interested in are:

```
# issue operations
issue.operations.assign = {0}Assign{1} this issue
issue.operations.assign.tome = to me
issue.operations.attach = {0}Attach file{1} to this issue
issue.operations.attachscreenshot = {0}Attach screenshot{1} to this issue
issue.operations.comment = {0}Comment{1} on this issue
issue.operations.delete = {0}Delete{1} this issue
issue.operations.edit = {0}Edit{1} this issue
issue.operations.link = {0}Link{1} this issue to another issue
issue.operations.move = {0}Move{1} this issue to another project
```

In the text, {0} and {1} indicate sections that will be replaced by JIRA (here, the beginning and end of links), and should not be removed. Edit the text to look like:

```
# issue operations
issue.operations.assign = {0}Assign{1}
issue.operations.assign.tome = to me
issue.operations.attach = {0}Attach file{1}
issue.operations.attachscreenshot = {0}Attach screenshot{1}
issue.operations.comment = {0}Comment{1}
issue.operations.delete = {0}Delete{1}
issue.operations.edit = {0}Edit{1}
issue.operations.link = {0}Link{1} to another issue
issue.operations.move = {0}Move{1} this issue to another project
```

Standalone - Restart JIRA.
Webapp edition - Rebuild JIRA as per your App Server using the build scripts and restart the application server.
If you are deploying a .war to Tomcat - please note that Tomcat unpacks .war files into the `webapps/` directory. **Delete this directory** (eg. `webapps/ROOT` between redeploys, or this old webapp will be deployed.

The changes should be visible when next you view the page.

Mapping custom events to new email templates

CustomWare, one of our development partners, has a page describing how to map custom events to new email templates.

Thanks Guys.

Tutorial - How to Create a Custom Release Notes Template Containing Release Comments

Many JIRA users want to expand the generated Release Notes to include release comments for each issue in the report. This tutorial shows how to do this using a custom field and some customized velocity templates. It assumes the reader understands JIRA's custom fields. For more information on custom fields see the JIRA documentation.

- Step 1: Creating a Custom Field
- Step 2: Creating Custom Velocity Templates for Release Notes Reports
- Step 3: Modifying JIRA to Use Custom Velocity Templates
- Step 4: Filtering Release Notes
- Conclusion
- Further Resources

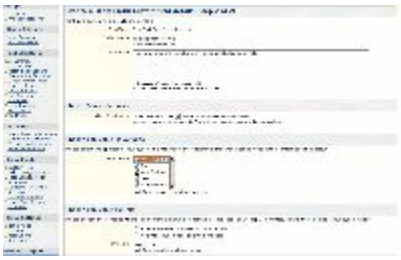
Step 1: Creating a Custom Field

The first thing to do is add a custom field for your release note comments. To add the custom field, click the *Administration* link in the top navigation bar then choose *Issue Fields* > *Custom Fields* and on the resulting page choose the *Add Custom Field* link. You should see this screen:

From this screen choose the *Free Text Field (unlimited text)* field type and click *next*. On the next page enter the name, description, search template, applicable issue types, and applicable context of our new custom field. The values should be:

- name: release notes comments
- description: This is a comment to include on the generated release notes.
- search template: Free Text Searcher (the default for this type of custom field)
- applicable issue typs: Any Issue Type (this can quite easily be changed if you only wish this custom field to apply to a certain issue type)
- applicable context: Global context. Apply to all issues in JIRA (you can choose here to suit your preference)

The input screen should look like this:



Click the Finish button. You will be taken to the 'Associate field release notes comments to screens' screen. Click the checkbox associated with the 'Resolve Issue Screen'. In this example it only really makes sense to show this custom field when resolving or closing an issue. The screen should look like this:

Click the Update button. You are now finished creating our custom field. When you choose to resolve or close an issue you should now have a text area where you can enter comments to appear on the release notes.

Step 2: Creating Custom Velocity Templates for Release Notes Reports

Velocity is the templating engine used in JIRA. Currently JIRA ships with a text and a html release notes report template. For this tutorial we will modify the existing templates to also display the custom field that we created in the previous step, for each issue in the report. The modified text template looks something like this:

```
#macro (getReleaseNoteComment $issue $customFieldManager)
#set ($customFields = $customFieldManager.getCustomFieldObjects($issue.project.getLong("id"),
$issue.issueType.getString("id")))
#foreach($customField in $customFields)
#if($customField.name.equals("release notes comments"))
#if($customField.getValue($issue)) - Release Comment:
$textUtils.htmlEncode($customField.getValue($issue))#end
#end
#end
#end

<title>$action.getText('release.notes.text.title', $project, $version) </title>
<body>
<table>
<tr>
<td>

#foreach ($issueType in $issueTypes)
#if($issueType.issues.size() > 0)
<h2>$textUtils.htmlEncode($issueType.name)</h2>
<ul>
#foreach ($issue in $issueType.issues)
<li>[<a xhref='$!appProps.getString("jira.baseurl")/browse/$issue.key'>$issue.key</a>] -
 $textUtils.htmlEncode($issue.summary)#getReleaseNoteComment($issue $customFieldManager)</li>
#end
</ul>
#end
#end
</td>
</tr>

<tr>
<td>

<hr width="100%">

<a name="editarea"><h2>$action.getText('release.notes.edit.copy')</h2></a>
<p>$action.getText('release.notes.description')<br></p>

<textarea rows="40" cols="120">

$action.getText('release.notes.heading', $project, $version)

#foreach ($issueType in $issueTypes)
#if($issueType.issues.size() > 0)
** $textUtils.htmlEncode($issueType.name)
#foreach ($issue in $issueType.issues)
* [$issue.key] - $textUtils.htmlEncode($issue.summary)#getReleaseNoteComment($issue
$customFieldManager)
#end
#end

#end
</textarea>
</td>
</tr>
</table>
</body>
```

The modified html template looks like this:

```
#macro (getReleaseNoteComment $issue $customFieldManager)
#set ($customFields = $customFieldManager.getCustomFieldObjects($issue.project.getLong("id"),
$issue.issueType.getString("id")))
#foreach($customField in $customFields)
#if($customField.name.equals("release notes comments"))
#if($customField.getValue($issue)) - <b>Release Comment</b>:
$textUtils.htmlEncode($customField.getValue($issue))#end
#end
#end
#end
<title>$action.getText('release.notes.html.title', $project, $version) </title>
<body>
<table>
<tr>
<td>

#foreach ($issueType in $issueTypes)
#if($issueType.issues.size() > 0)
<h2>$textUtils.htmlEncode($issueType.name)</h2>
<ul>
#foreach ($issue in $issueType.issues)
<li>[<a xhref='$!appProps.getString("jira.baseurl")/browse/$issue.key'>$issue.key</a>] -
 $textUtils.htmlEncode($issue.summary)#getReleaseNoteComment($issue $customFieldManager)</li>
#end
</ul>
#end
#end
</td>
</tr>

<tr>
<td>

<hr width="100%">

<a name="editarea"><h2>$action.getText('release.notes.edit.copy')</h2></a>
<p>$action.getText('release.notes.description')<br></p>

<textarea rows="40" cols="120">

$action.getText('release.notes.heading', $project, $version)
#foreach ($issueType in $issueTypes)
#if($issueType.issues.size() > 0)

<h2>$textUtils.htmlEncode($issueType.name)</h2>
<ul>
#foreach ($issue in $issueType.issues)
<li>[<a xhref='$!appProps.getString("jira.baseurl")/browse/$issue.key'>$issue.key</a>] -
 $textUtils.htmlEncode($issue.summary)#getReleaseNoteComment($issue $customFieldManager)</li>
#end
</ul>
#end
#end
</textarea>
</td>
</tr>
</table>
</body>
```

The only difference between the above templates and the originals is the definition of the velocity macro 'getReleaseNoteComment' and its use after the issue summary. In the macro we use the context variable $customFieldManager which is an instance of the com.atlassian.jira.issue.CustomFieldManager interface. We use the CustomFieldManager to get all CustomFieldObjects for the current issue and then iterate through them looking for the field we want. When we find the field named 'release notes comments' we get and display the value if it is not null. *NOTE: we do not use the getCustomFieldObjectByName method because it is deprecated and we can not be certain that the custom field name will be unique within the JIRA instance.*

These velocity templates will display the 'release notes comments' custom field in our generated Release Notes. Now you need to tell JIRA to use

the new templates.

Step 3: Modifying JIRA to Use Custom Velocity Templates

This is quite a simple step. Place the two velocity templates into your JIRA distribution* in the directory: WEB-INF/classes/templates/jira/project/releasenotes . Next you must modify the WEB-INF/classes/jira-application.properties file. There are two variables that are of importance to us:

{{jira.releasenotes.templatenames = Text, Html
jira.releasenotes.templates = releasenotes-text.vm, releasenotes-html.vm}}

We do not need to modify the first property since we still want a Text and Html option and have only changed the templates. We modify the second property to look like this:

```
jira.releasenotes.templates = releasenotes-with-releasecomment-text.vm,
releasenotes-with-releasecomment-html.vm
```

Save the properties file and restart Jira. Now we can test that what we have done works. You must make sure you can generate some Release Notes that will contain issues:

- Make sure that you have created a version for the project you are testing this under (release notes can only be generated for a version).
- Make sure that you have some issues that have the fix versions set to the version you created above.
- Make sure that you have entered some release comment text on the issues with the fix versions from above(in our example if there is not a release comment for an issue then nothing will display and it will look very much like the original Release Notes).

Browse to the release notes page, click the 'Browse Project' link in the top navigation, click the 'Release Notes' link on the resulting page, choose your 'Version' and 'Style', and click 'Create'. You should see the release note comments text after the summary text for each issue. The screen should look something like this:

Step 4: Filtering Release Notes

As an optional extra, you can filter the release notes based on various issue properties to see if they match your criteria. For example, to print only issues that have a resolution of Fixed and a Status, put an `#if` statement around the code that creates a bullet point for the issue:

```
#set ($resolution = $issue.getResolutionObject())
#set ($status = $issue.getStatusObject())
## check for resolved or closed and fixed
#if (($status.getId() == "5" || $status.getId() == "6") && $resolution.getId() == "1")
 <li>[<a href='$!appProps.getString("jira.baseurl")/browse/$issue.getString("key")'>$issue.getString(
"key")</a>] -
 $textUtils.htmlEncode($issue.getString("summary"))#getReleaseNoteComment($issue
$customFieldManager)</li>
#end
```

* The Jira Distribution is the package that you downloaded and installed from our website. The directory commonly has a name starting with **atlassian-jira-<edition>-<version>**.

Conclusion

This tutorial shows a very simple modification to the Release Notes reports but the concepts used within should show you how to customize Release Notes to fit many other needs. The use of custom fields and the mechanism for accessing their values through a velocity template can allow a great number of extensions beyond the scope of this example.

Have fun and good luck!

Further Resources

1) Available Fields - Please see the JIRA Remote API for details on what information you can retrieve. For example:

- Projects
- Issues

2) Template Syntax - The Velocity User Guide will help you choose the right syntax.

3) Help & Support - Please post to the JIRA Development Forums for get help from the user community with your templates.

Velocity Context for Email Templates

JIRA 3.6 introduced the TemplateContext and TemplateIssue objects into the Velocity context available for each email template.

The TemplateContext provides access to some general object references for use in the templates - e.g. Utilities, Change Log details, etc.

The TemplateIssue provides access to the Issue object - further extended by helper methods for displaying information in rendered HTML format (i.e. environment, description and comments) and users who may not exist in the system. All issue related details can be retrieved from the TemplateIssue.

The following tables lists other objects that are also available within the context:

Velocity Variable	JIRA Object/Value	Description
userutils	com.atlassian.core.user.UserUtils	collection of utility methods for querying, creating, and re users
velocityhelper	com.atlassian.jira.util.JiraVelocityHelper	a collection of miscellaneous methods (e.g. urlencode, indentTextBlock, printChangelog
textutils	com.opensymphony.util.TextUtils	collection of utility functions for manipulating text
constantsManager	com.atlassian.jira.config.ConstantsManager	JIRA's constants for priorities, resolutions, issue types, st
projectManager	com.atlassian.jira.project.ProjectManager	management of project entities (e.g. getProjectByName, getComponents, getCategories)
customFieldManager	com.atlassian.jira.issue.CustomFieldManager	functions for working with custom fields
applicationProperties	com.atlassian.jira.config.properties.ApplicationProperties	access JIRA's application properties
jirautils	com.atlassian.jira.util.JiraUtils	a handful of miscellaneous utility methods (e.g. isPublicM isSetup)
jirakeyutils	com.atlassian.jira.util.JiraKeyUtils	methods for working with project keys
buildutils	com.atlassian.jira.util.BuildUtils	JIRA build and edition information
jiraUserUtils	com.atlassian.jira.user.util.UserUtil	user level operations (e.g. getActiveUserCount, getDisplayableNameSafely, addUserToGroup)
mailPluginsHelper	com.atlassian.jira.mail.JiraMailPluginsHelperImpl	method for checking whether plugins are enabled
userformat	com.atlassian.jira.plugin.profile.UserFormatManager	formatting user names
context	com.atlassian.jira.mail.TemplateContext	The Template Context object itself
issue	com.atlassian.jira.mail.TemplateIssue	An Issue object with extended helper methods for display the issue for which the email notification event was trigge
params	java.util.Map	Parameters of the IssueEvent that triggered this email no
remoteUser	com.opensymphony.user.User	The user who has triggered this event
stringUtils	org.apache.commons.lang.StringUtils	Utility object providing formatting methods
padSize	java.lang.Integer	Used in text emails for padding the display
timeSpentFieldId	java.lang.String	The Id of the TimeSpentField - used in changelog for wor notifications
attachments	Collection(org.ofbiz.core.entity.GenericValue)	Collection of Generic Values that represents the attachm associated with the issue
htmlComment	java.lang.String	The comment associated with this event as a rendered H formatted string. A basic string is returned if the commen be rendered correctly - including links for any referenced issues.

comment	com.atlassian.jira.issue.comments.Comment	The comment associated with this event. The body of the comment will be the raw unprocessed comment as entered user (e.g. the raw wiki markup)
commentauthor	com.opensymphony.user.User	The author of the comment
roleVisibilityLevel	java.lang.String	The comment's project role level (overwritten by worklog)
groupVisibilityLevel	java.lang.String	The comment's group level (overwritten by worklog)
originalhtmlComment	java.lang.String	
originalcommentauthor	com.opensymphony.user.User	
originalroleVisibilityLevel		comment (overwritten by worklog)
originalgroupVisibilityLevel		comment (overwritten by worklog)
worklog	com.atlassian.jira.issue.worklog.Worklog	
timeSpentUpdated	java.lang.Boolean	
startDateUpdated	java.lang.Boolean	
commentUpdated	java.lang.Boolean	
visibilityUpdated	java.lang.Boolean	
originalworklog	com.atlassian.jira.issue.worklog.Worklog	
changelog	org.ofbiz.core.entity.GenericValue	The Generic value representing the changelog associate the change
changelogauthor	com.opensymphony.user.User	The change log author
security	org.ofbiz.core.entity.GenericValue	The Generic value representing the security level, if any, associated with this issue
rendererManager	com.atlassian.jira.issue.RendererManager	JIRA 3.13.3 onwards! allows use of renderers such as wi renderer. (e.g. $rendererManager.getRenderedContent("atlassian-wiki-r "**bold wiki**", $issue.getIssueRenderContext()))
recipient	com.opensymphony.user.User	The recipient of the email
i18n	com.atlassian.jira.utilI.I18nHelper	i18n translation helper for the current recipient
dateformatter	com.atlassian.jira.web.util.OutlookDate	Date and time formatter for the current recipient's locale

Services, Listeners and Handlers

- Adding your own email handling classes — In JIRA, emails may be periodically fetched (eg. via POP or IMAP) and processed (eg. create a new issue, create a comment). You can add new ways of processing emails by implementing the MessageHandler interface in JIRA (or subclassing one if its implementations).

Examples

- Service for updating select-list values

Adding your own email handling classes

In JIRA, emails may be periodically fetched (eg. via POP or IMAP) and processed (eg. create a new issue, create a comment). You can add new ways of processing emails by implementing the MessageHandler interface in JIRA (or subclassing one if its implementations).

> **MessageHandlers**
> For more information on MessageHandlers, including API documentation and source to the system handlers, see the documentation

After writing your MessageHandler implementation, this is how to get it into JIRA:

1. Compile your handler and add it to the JIRA classpath (either as a JAR in WEB-INF/lib or as a class in WEB-INF/classes). JIRA

Standalone has an `external-source` directory whose README describes a quick way to compile source (see JIRA Standalone quick source modifications).
2. Add your handler to the services/com/atlassian/jira/service/services/pop/popservice.xml
3. Restart JIRA

You should now be able to select your own MessageHandler when configuring a POP service.

3.6 Technical Improvements

Many things have changed under the covers in JIRA 3.6 to make JIRA more flexible. Many of these things do not yet have a proper interface, and are set either through `jira-application.properties`, through workflow properties or by writing code extending certain JIRA interfaces. These features are likely to change in future, either to make them more easily available or more flexible. While we will attempt to provide backwards-compatibility, this is not guaranteed. If you use these features, please read the upgrade notes of future JIRA releases, where we will note any changes.

This page guides you through these advanced customizations.

- Programmatically determining automatic assignee
- Resolutions per workflow step
- Subtask creation form enhancements

Programmatically determining automatic assignee

In JIRA 3.6, it is now possible to programmatically determine which assignee to set if the user chooses "Automatic". Previously it was always the project/component lead or "unassigned".

To do this, make your own implementation of the AssigneeResolver interface, and package it as a component plugin. For example, here is a simple implementation that makes Bugs unassigned:

```
package com.atlassian.jira.plugin.ext;

import com.atlassian.jira.plugin.assignee.impl.DefaultAssigneeResolver;
import com.atlassian.jira.issue.Issue;
import com.atlassian.jira.project.ProjectManager;
import com.atlassian.jira.security.JiraAuthenticationContext;
import com.opensymphony.user.User;

import java.util.Map;

/**
 * Leaves bugs unassigned by default; everything else follows the normal JIRA rules.
 */
public class CustomAssigneeResolver extends DefaultAssigneeResolver
{
    public CustomAssigneeResolver(ProjectManager projectManager, JiraAuthenticationContext
authenticationContext)
    {
        super(projectManager, authenticationContext);
    }

    public User getDefaultAssignee(Issue issue, Map fieldValuesHolder)
    {
        if (issue.getIssueTypeObject().getId().equals("1")) // 1 is the id for the Bug issue type.
{
            return null; // unassigned
}
        else return super.getDefaultAssignee(issue, fieldValuesHolder);
    }
}
```

See the attached plugin source which packages this, and can be used as a starting point for your own customizations.

Resolutions per workflow step

Resolutions per workflow step.

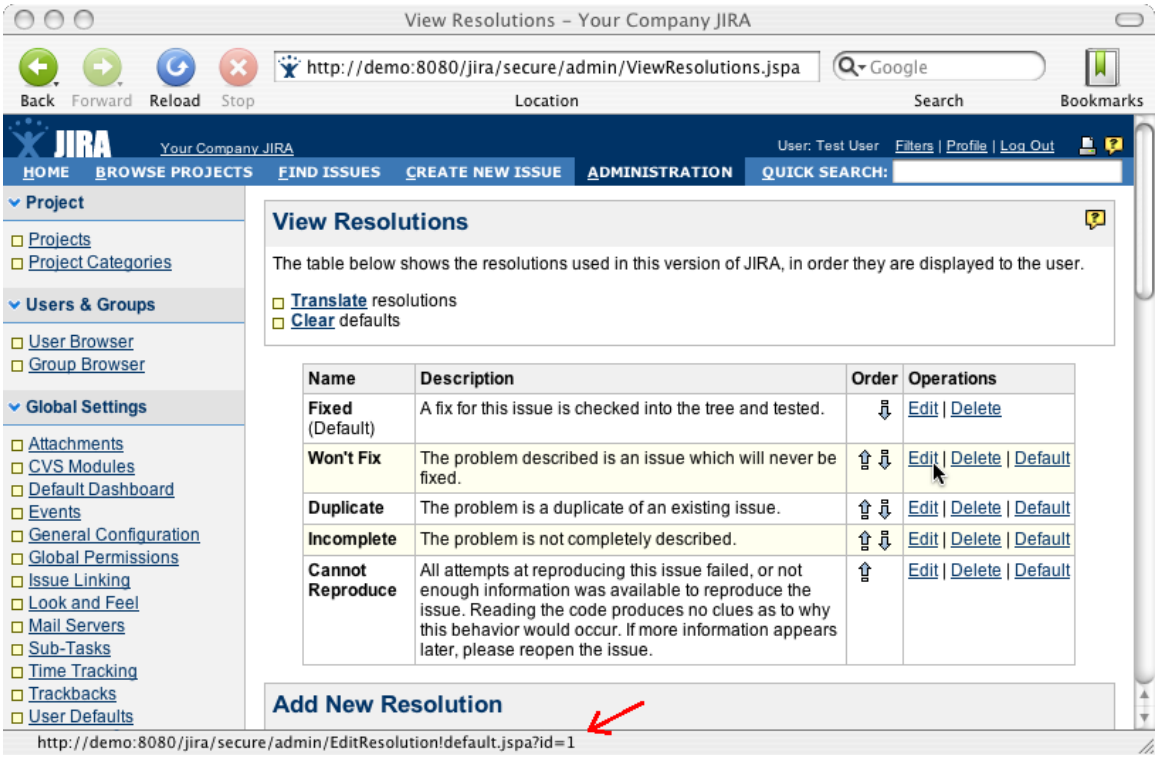
It is sometimes necessary to restrict what resolutions are shown on a particular "Resolve" screen. For instance, one might have a "Reject"

transition which omits resolutions like "Fixed", leaving only resolutions like "Invalid".

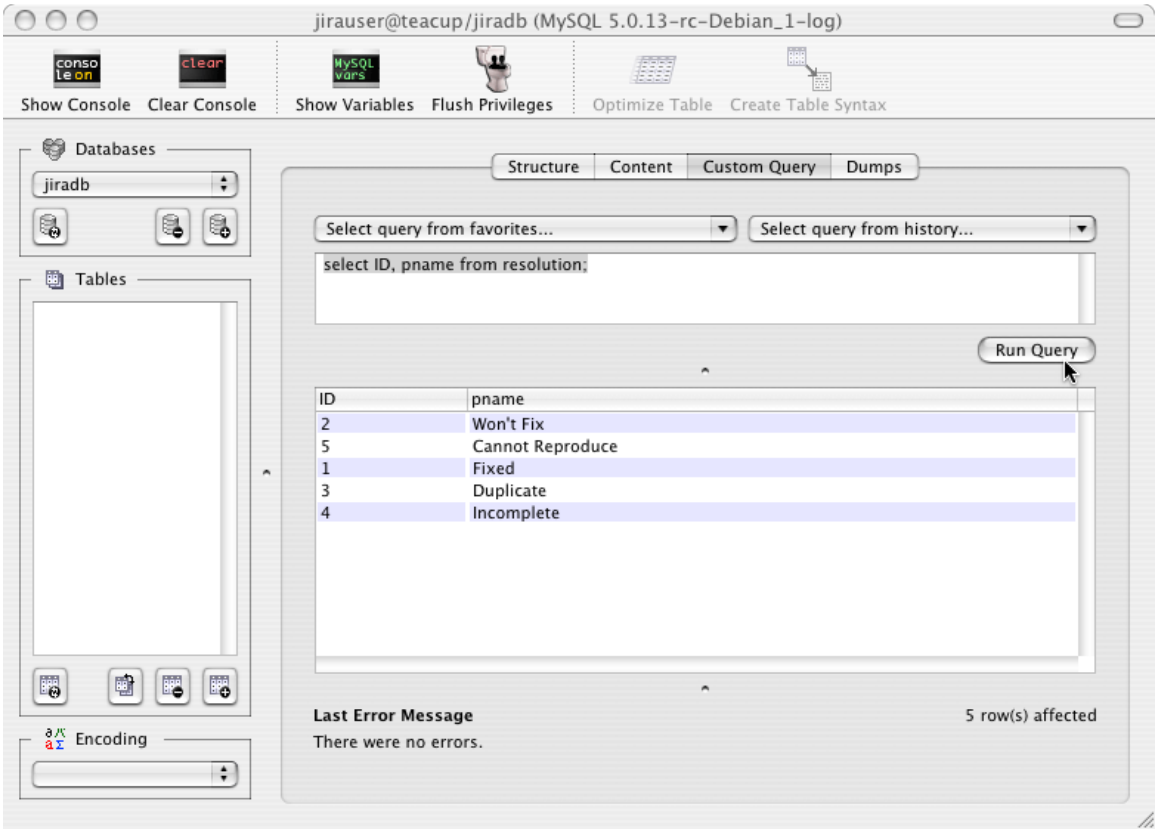
This can be done as follows:

Find out the IDs of the resolutions to be excluded.

The IDs of resolutions can be discovered by examining URLs of operations on resolutions:

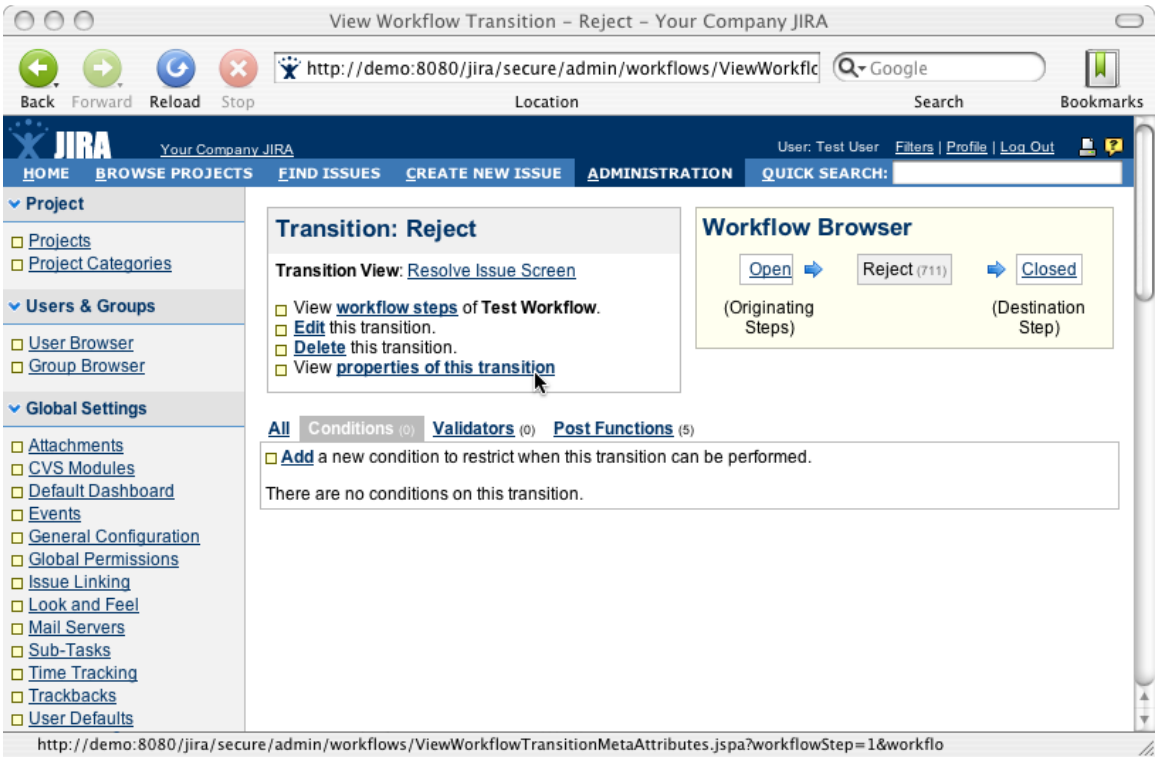


or more directly by querying the `resolutions` table in the database:

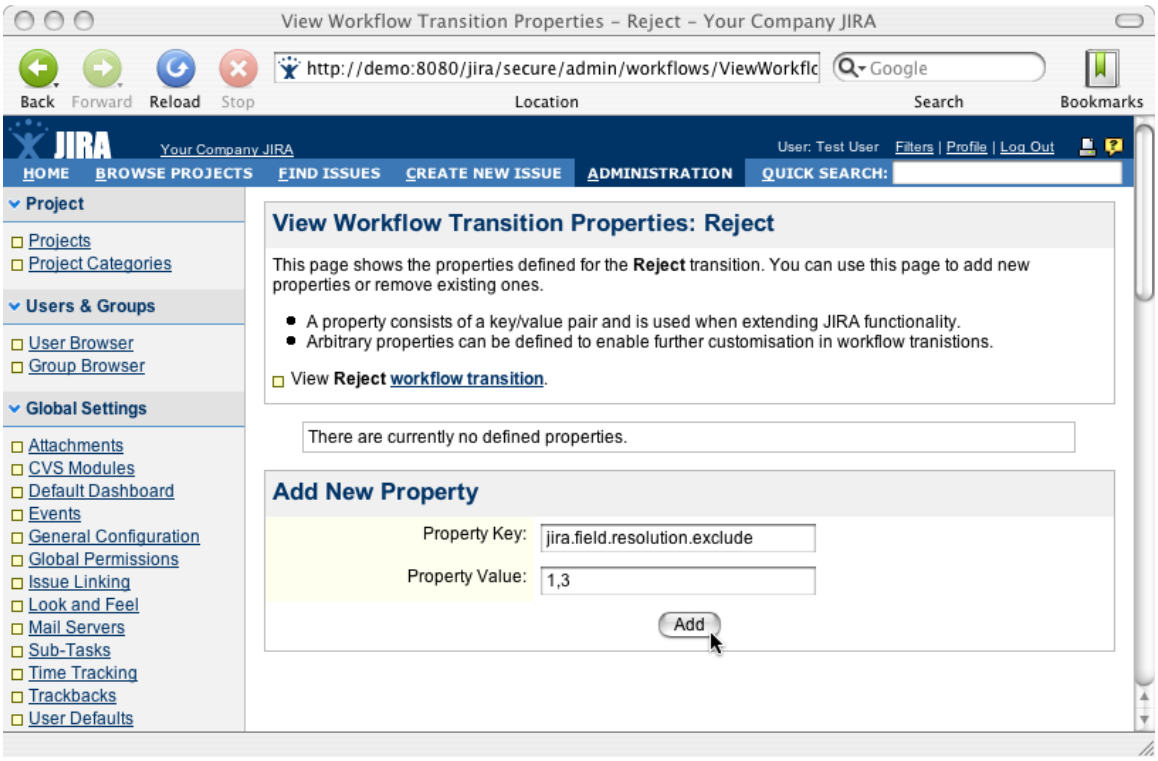


Exclude the resolutions in the workflow

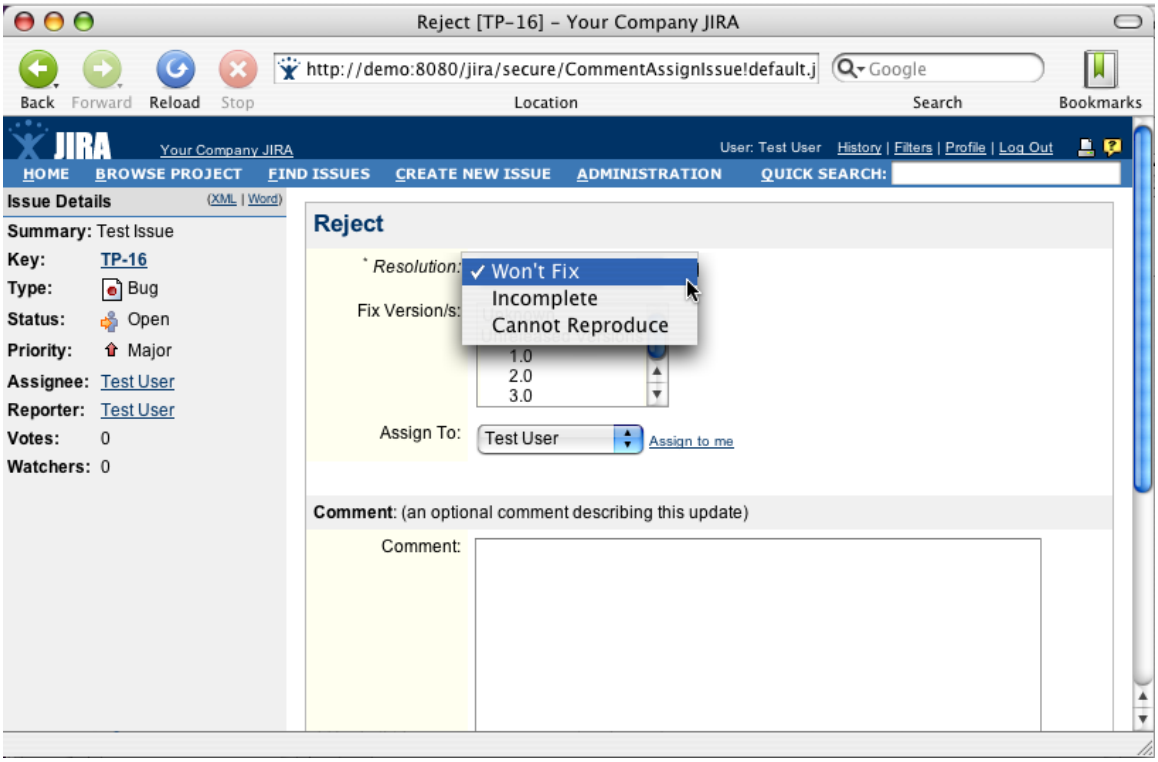
In the workflow editor, locate the workflow you wish to edit. If it is active, you will need to copy it and later associate it with projects with a new workflow scheme. Select the transition you wish to limit resolutions on, and click "properties of this transition":



Now set a property called `jira.field.resolution.exclude` to a comma-separated list of resolutions to hide for this transition:



Now activate the workflow, and create a test issue. The resolutions will be excluded for that transition:



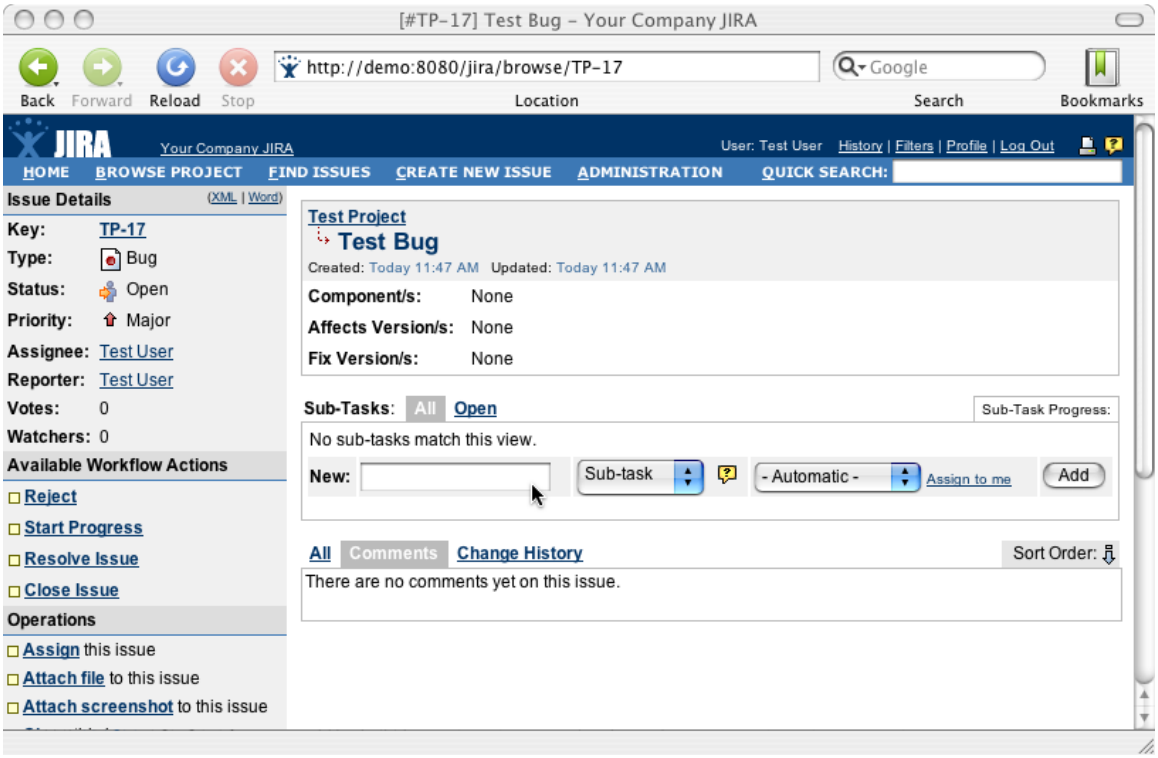
Subtask creation form enhancements

Subtask Creation form enhancements

If subtasks are enabled, a "quick subtask creation" form allows subtasks to be quickly created and manipulated from the parent issue. Various enhancements have been made to this.

Always show

By default, the subtask creation form only appears on the parent issue if one or more subtasks exists. It can now be made to always appear:



This is done by editing `jira-application.properties`, and setting the following property to `true`:

```
# Should we show the form for creating subtasks in the parent issue even when there are no subtasks
yet?
jira.view.issue.show.quicksubtask.when.no.subtasks=true
```

Display arbitrary fields, default field values

What fields to show can now be determined in `jira-application.properties`:

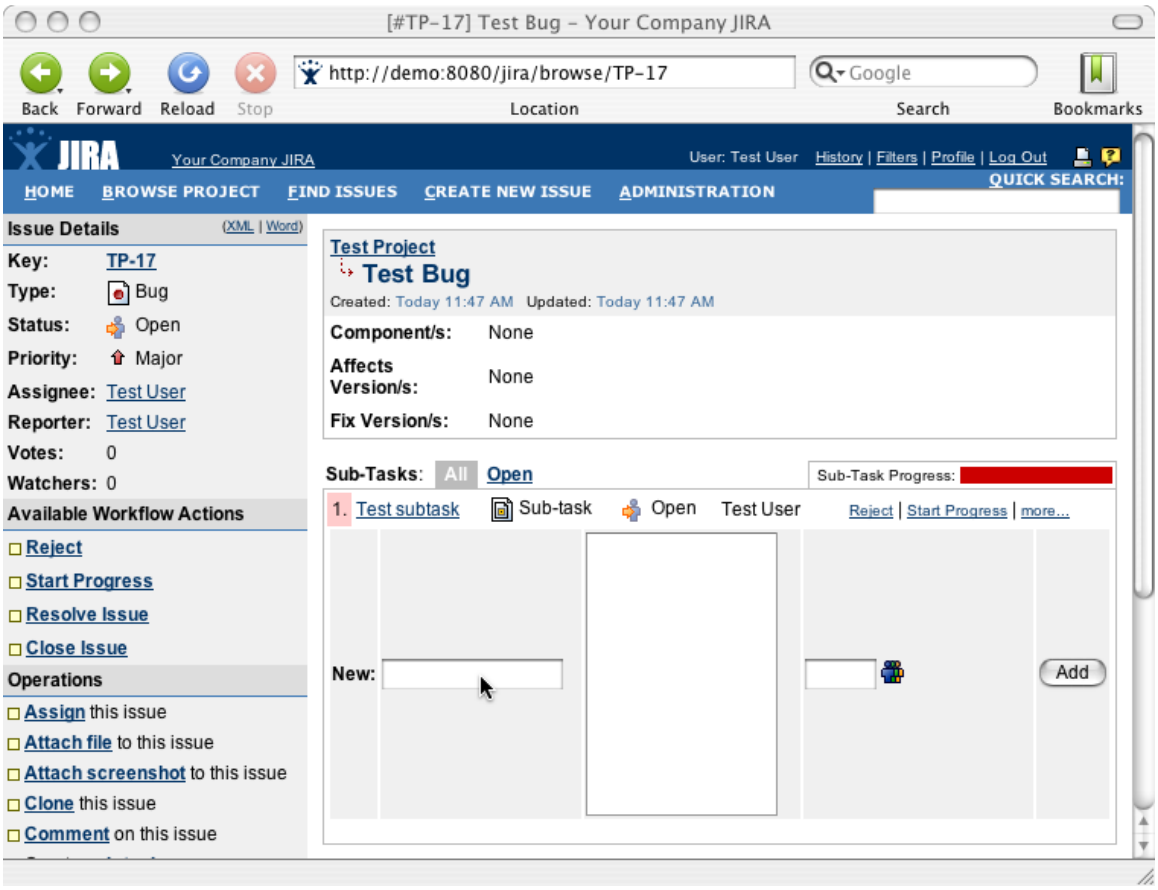
```
# The fields to show on the sub-task quick creation form that appears on the View Issue page
# The values must be valid issue field ids (see IssueFieldConstants class) and can contain custom
fields
# The format of the custom field is 'customfield_<id>' where <id> is the numeric id of the field in
the database
# Field values may be preset, eg. 'summary,issuetype=7,assignee=-1' where only 'summary' would show.
# Note: Leave the trailing comma as it marks an empty cell on the form.
jira.subtask.quickcreateform.fields = summary,issuetype,,assignee,
```

So for example, if we wish to prompt for the summary, description and a "Group Affected" custom field, while defaulting the issue type and leaving assignee to be automatically chosen, we would have:

```
jira.subtask.quickcreateform.fields = summary,description,issuetype=7,customfield_10002,assignee=-1
```

Where the issue type and custom field IDs can be discovered from the database ("select * from issuetype;", "select * from customfield;") or by examining URLs in the admin section as above. The "-1" means "automatic".

The result is the following:



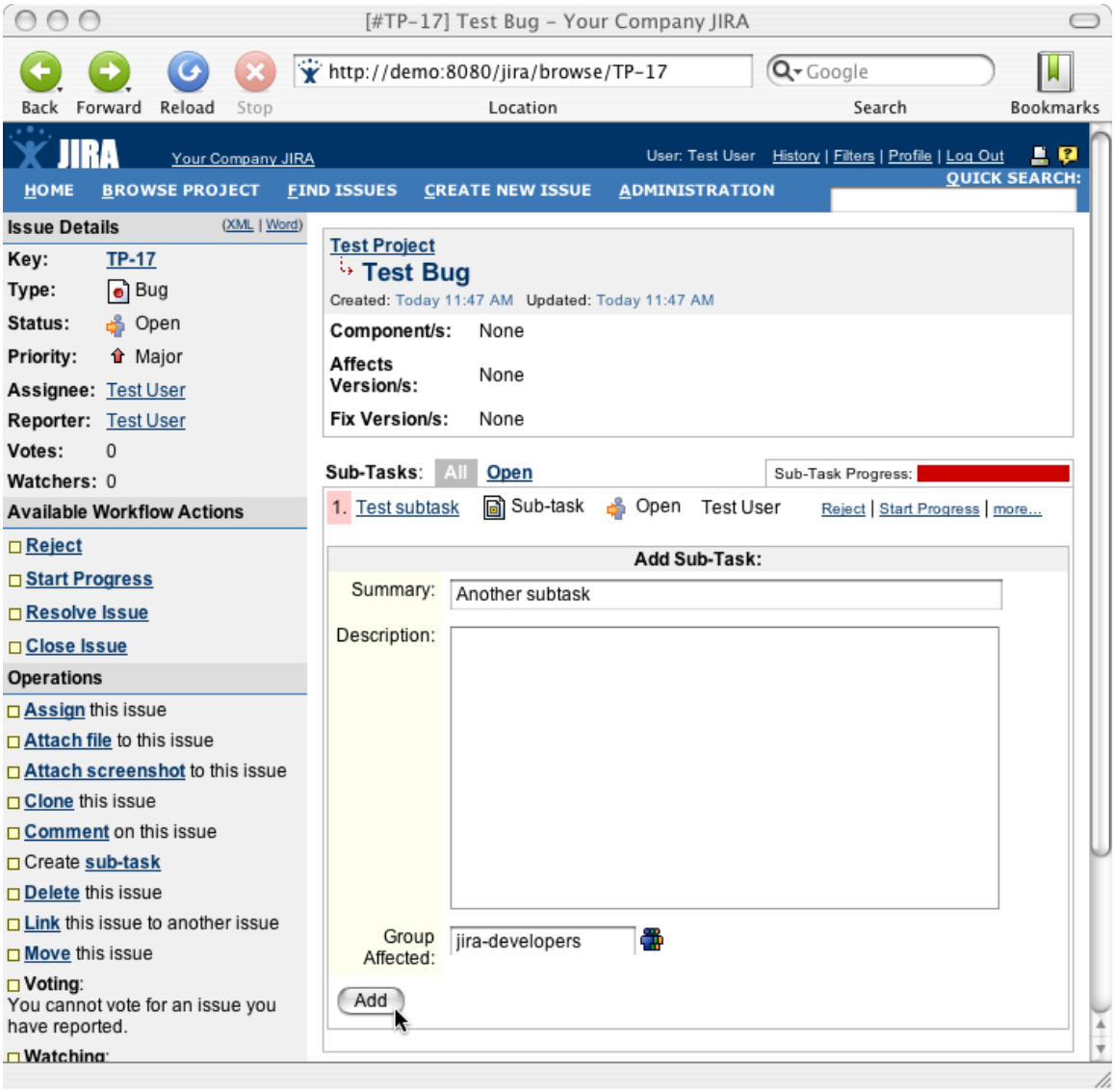
This does work, but doesn't look so good, which introduces the next enhancement:

Vertical alignment

Fields can now be displayed vertically instead of horizontally, by setting the Velocity template used. To switch to vertical layout, comment out the horizontal template and uncomment the vertical template:

```
# This template formats the sub-task quick creation form that appears on the View Issue page.
# Horizontal and vertical forms are provided (vertical works better if fields need labels or use
textareas)
jira.subtask.quickcreateform.template = templates/jira/issue/subtask/quickcreationform-vertical.vm
#jira.subtask.quickcreateform.template = templates/jira/issue/subtask/quickcreationform-horizontal.vm
```

After restarting, the form now looks like this:



Future enhancements

Although the entry form can be customized to include arbitrary fields, which fields to *display* from entered subtasks is still hardcoded. In future we hope to make this fully editable (probably as a screen type).

JIRA Developer FAQ

This is a constantly updated FAQ listing questions and answers asked by people developing JIRA plugins and working with the JIRA code base in general.

> If you have a question, please ask it on our Developer Forums and certain threads will be merged back into the FAQ.

Questions

How do I extract CVS commits for an issue

How to set a custom field value on an issue?

How do I get access to manager X?
How do I create a report with custom issue set using IssueNavigator
Can I have a different Excel template for each report?
Creating and Editing an Issue
How do I get a handle to object X (specifically IssueNavigator)?
How do I get determine which Issues a user is allowed to see?
How do I get access to the current request?
Is it possible to have a report plugin only appear for specific projects?
How to search for Issues based on CustomField
How do I get a handle to the model (issues) then iterate?
How do I find issues with field X?
How do I find all Versions in a Project?
How do I know what else lurks in the Velocity context?
Why am I getting 'Exception looking up public key' exception while building a plugin?
How to add searching to a Custom Field?
How do I set the value for default comment security level?
How to search for Issues from within a Plugin
How do I remove an issue operation?
How do I write a servlet for JIRA?
How do I get more help with SOAP?
How do I get a handle to the 'components' of a JIRA project?
I have a GenericValue issue. How do I convert this to an Issue object?

Can I have a different Excel template for each report?

Excel templates of reports are controlled by the file `\secure\views\browser\report-excel.jsp`. While you can't easily have different `report-excel.jsp` for each report, you can easily edit the `report-excel.jsp` file to behave differently for each report (or if you want, use a different include for each report).

The report-excel.jsp has full access to methods available in `com.atlassian.jira.web.action.browser.ConfigureReport`. This means that you can figure out what report it is from the `getReportKey()` and `getReport()` methods, and make the page behave accordingly.

How do I create a report with custom issue set using IssueNavigator

This code could be cleaned up, but it shows how to generate an IssueNavigator table from a List of issues. This will be very helpful to anyone trying to generate a list of issues who wants to reuse the excellent IssueNavigator table we all know and love.

```
    /**
     * Using the list of issues we've compiled generate a report using the IssueNavigator.
     * @param action
     * @param params
     * @param issues
     * @return
     * @throws Exception
     */
    protected String generateIssueTableslForReport(ProjectActionSupport action, Map params, List
issues) throws Exception {
        HttpServletRequest request = ActionContext.getRequest();
        IssueTableLayoutBean layoutBean =
issueNavigator.getTableLayoutFactory().getStandardLayout(issueNavigator.getSearchRequest(),
issueNavigator.getRemoteUser());
        IssueTableWebComponent itwc = new IssueTableWebComponent();
        StringBuffer sb = new StringBuffer();
        List tempList = new LinkedList();

        // break up what gets returned into sets of issues revolving around an issue
// of type feature
for (int i = 0; i < issues.size(); i++) {
            Issue issue = (Issue) issues.get(i);
            if (issue.getIssueType().get("name").equals(FLUID_ISSUE_TYPE_FEATURE)) {
                if (tempList.size() > 0) {
                    sb.append("<br/>");
                    sb.append(itwc.getHtml(layoutBean, tempList, null));
                    sb.append("<br/>");
                    tempList.clear();
                }
            }
            // add this feature to the list
tempList.add(issue);
        }
        // write out the rest of the issues
sb.append("<br/>");
        sb.append(itwc.getHtml(layoutBean, tempList, null));
        sb.append("<br/>");
        return sb.toString();
    }
```

How do I extract CVS commits for an issue

This code snippet was extracted from the DefaultActionManager class in JIRA's source code:

```
/**
* Retrieves all of the commits for this issue from ALL of the repositories associated with the issue's
project
*
* @param issue
* @param remoteUser
*/

    public List getCommits(GenericValue issue, User remoteUser) throws GenericEntityException,
RepositoryException
    {
        List commits = new ArrayList();

        if (issue == null)
            throw new IllegalArgumentException("Issue cannot be null.");
        if (!"Issue".equals(issue.getEntityName()))
            throw new IllegalArgumentException("Entity must be of type Issue");

        if (!hasPermission(issue, remoteUser))
        {
            // If the user does not have the reuqited permission, do not return any information.
return Collections.EMPTY_LIST;
        }

        Collection repositories = getRepositories(issue);
        for (Iterator iterator = repositories.iterator(); iterator.hasNext();)
        {
            Repository repository = (Repository) iterator.next();
            try
            {
                List coms = repository.getCommitsForIssue(issue.getString("key"));
                for (int i = 0; i < coms.size(); i++)
                {
                    commits.add(new Commit((VCSCommit) coms.get(i), remoteUser, repository.getName(),
repository.getRepositoryBrowser()));
                }
            }
            catch (OutOfMemoryError e)
            {
                // Add an issue action that represents OutOFMemeory error
commits.add(new OutOfMemoryCommitIssueAction(remoteUser, new Timestamp(System.currentTimeMillis()),
repository.getName()));
            }
        }

        // Sort by date
Collections.sort(commits);
        return commits;
    }
```

How do I find all Versions in a Project?

Given an issue, you can retrieve all versions of its Project.

```
GenericValue project = projectManager.getProject(myIssueGV);
List versions = versionManager.getVersions(project);
Collections.sort(versions);
```

How do I find issues with field X?

How do I find issues with field X?

You can find more information about searching in JIRA at How to search for Issues from within a Plugin and also in here.

How do I get access to manager X?

How do I get access to manager X?

You can get access to all Managers in JIRA through the Pico container's `ComponentManager`.

You can inject the component by registering it in your constructor as per PicoContainer and JIRA or you can statically load it from the `ComponentManager` itself. For example:

```
IssueLinkTypeManager issueLinkTypeManager =
 (IssueLinkTypeManager) ComponentManager.getComponentInstanceOfType(IssueLinkTypeManager.class);
```

How do I get access to the current request?

How do I get access to the current request?

Within your plugin code, you can perform the following static call to get access to the `HttpServletRequest` object. This then in turn gives you access to the session etc.

```
HttpServletRequest request = ActionContext.getRequest();
```

Please note!
The request object may be null in some contexts, so always do a null-test before using the returned `request`.

How do I get a handle to object X (specifically IssueNavigator)?

How do I get a handle to object X (specifically IssueNavigator)?

For JIRA plugins in many cases you can simply pass an object to your plugin's constructor. However some objects are not by default available. In these cases you have to add the <component> to your atlassian-plugin.xml file like so:

```
    <component key="issueNavigator" name="Issue Navigator" class=
"com.atlassian.jira.web.action.issue.IssueNavigator">      <interface
>com.atlassian.jira.web.action.issue.IssueNavigator</interface>
    </component>
```

There are many examples in the atlassian-plugin.xml files within the sample plugins found in the JIRA development kit.

How do I get a handle to the 'components' of a JIRA project?

```
    /**
     * get all the components for this project
     */
    public Collection getAllComponents() {
Long currentProjectId = (Long) ActionContext.getSession().get(SessionKeys.SELECTED_PROJECT);
GenericValue currentProject = (GenericValue) projectManager.getProject(currentProjectId);
        Collection components = new ArrayList(projectManager.getComponents(currentProject));
        return components;
    }
```

How do I get a handle to the model (issues) then iterate?

All of the objects defined externally to these methods are available to a JIRA plugin via [dependency injection].

```
    /**
     * Retrieve a list of all the issues in the current project.
     * Note that several of these objects are passed via dependency injection as
     * constructor parameters.
     * @return list of Issue objects
     */
    public List getAllIssuesInCurrentProject() {
        SearchRequest sr = new SearchRequest(authenticationContext.getUser());
        sr.addParameter(new ProjectParameter(currentProjectId));
        List issuesGenericValues = null;
        List issues = new ArrayList();
        try {
            issuesGenericValues = searchProvider.search(sr, authenticationContext.getUser());
            // convert to issue types
issues = convertGenericValuesToIssues(issuesGenericValues);
        } catch (SearchException e) {
            e.printStackTrace();  //To change body of catch statement use File | Settings | File
Templates.
errorMessage += "exception: " + e.getMessage();
        }
        return issues;
    }


    /**
     * GenericValue objects are the frequently used return type, but dealing with
     * Issue object is more helpful for this plugin. Hence this convenience method
     * converts a collection of GenericValues to Issue objects.
     *
     * @param issuesGenericValues
     * @return List of issues
     */
    public List convertGenericValuesToIssues
            (Collection issuesGenericValues) {
        List issues = new ArrayList();
        for (Iterator iterator = issuesGenericValues.iterator(); iterator.hasNext();) {
            GenericValue genericValue = (GenericValue) iterator.next();
            issues.add(issueFactory.getIssue(genericValue));
        }
        return issues;
    }
```

How do I get determine which Issues a user is allowed to see?

How do I get determine which Issues a user is allowed to see?

Given a list of issues, use

```
IssueUtils.filterIssues(issueGVs, new PermissionsParameter(user))
```

to filter the list of issues down to only those that the user has permission to view.

How do I get more help with SOAP?

How do I get more help with SOAP?

The best place to get help for SOAP is at Creating a SOAP Client. This page links to the example SOAP Client, which is an excellent source to see how things work.

The SOAP client has example usages to most common functionality available in the SOAP interface. Including update issue and progressWorkflow

For example:

```
private static void testUpdateIssue(JiraSoapService jiraSoapService, String token, final String
issueKey)
        throws RemoteException
{
    // Update the issue
RemoteFieldValue[] actionParams = new RemoteFieldValue[]{
        new RemoteFieldValue("summary", new String[] {NEW_SUMMARY}),
        new RemoteFieldValue(CUSTOM_FIELD_KEY_1, new String[] {CUSTOM_FIELD_VALUE_1}),
        new RemoteFieldValue(CUSTOM_FIELD_KEY_2, new String[] {CUSTOM_FIELD_VALUE_2})};

    jiraSoapService.updateIssue(token, issueKey, actionParams);
}
```

This updates the summary, and custom fields of an issue.

Need more help?

Try the jira-developer mailing list is your next stop. The mailing list archives a lot of helpful answers to most of the common SOAP problem, and there are helpful users and developers there to answer potential questions.

Willing to pay for help?

The Atlassian Partners site has a listing of partner specialities. You can also search for a specific keyword such as SOAP

How do I know what else lurks in the Velocity context?

Most people scratch their head to know what parameters and `Object`'s are in the Velocity context, the links below may come into help:

- Contents of the Velocity Context
- Velocity Context for Email Templates

For developers, who want to know more, the `DefaultVelocityManager` creates a self-reference back to the `VelocityContext` with these lines:

```
    protected VelocityContext createVelocityContext(Map params)
    {
        if (params != null)
        {
            params.put("ctx", params);
        }
    ...
```

Therefore, doing this in the template will display the available parameters:

```
#foreach($p in $ctx.keySet().toArray())
  $p.toString() - $ctx.get($p).getClass().getName().toString()
#end
```

e.g.

```
  textutils - com.opensymphony.util.TextUtils
  dateformatter - com.atlassian.jira.web.util.OutlookDate
  stringUtils - org.apache.commons.lang.StringUtils
  formatter - java.text.SimpleDateFormat
  constantsManager - com.atlassian.jira.config.DefaultConstantsManager
  buildutils - com.atlassian.jira.util.BuildUtils
  i - java.lang.Integer
  context - com.atlassian.jira.mail.TemplateContext
  jiraUserUtils - com.atlassian.jira.util.JiraUserUtils
  baseurl - java.lang.String
  security - $ctx.get($p).getClass().getName().toString()
  numdashes - java.lang.Integer
  ctx - java.util.HashMap
  jirakeyutils - com.atlassian.jira.util.JiraKeyUtils
  i18n - com.atlassian.jira.web.bean.I18nBean
  recipient - com.opensymphony.user.User
  velocityCount - java.lang.Integer
  attachments - java.util.ArrayList
  remoteUser - com.opensymphony.user.User
  userutils - com.atlassian.core.user.UserUtils
  applicationProperties - com.atlassian.jira.config.properties.ApplicationPropertiesImpl
  padSize - java.lang.Integer
  customFieldManager - com.atlassian.jira.issue.managers.DefaultCustomFieldManager
  issue - com.atlassian.jira.mail.TemplateIssue
  jirautils - com.atlassian.jira.util.JiraUtils
  params - java.util.HashMap
  projectManager - com.atlassian.jira.project.CachingProjectManager
  velocityhelper - com.atlassian.jira.util.JiraVelocityHelper
  timeSpentFieldId - java.lang.String
```

How do I remove an issue operation?

At the moment the easiest way to achieve this would be to edit the `includes/panels/issue/operations.jsp` file under the JIRA web application and remove the code that prints out the link to your operation. For example, if you want to remove the "Assign to" link, the code starts with:

```
<webwork:if test="hasIssuePermission('assign', issue) == true">

...

</webwork:if>
```

Simply remove that block and this will be removed from view. You may need to restart your application server for this change to be reflected.

How do I set the value for default comment security level?

While there is no nice way of setting the default comment security level through the admin section, a bit of creative javascripting can come to the rescue. It's not the best solution but it works.

First, you will need to examine the HTML code of the issue page, and find the user visibility select list. For example:

```
<select name="commentLevel" id="commentLevel">
...
    <option value="role:10002">Administrators</option>
...
    <option value="group:jira-administrators" >jira-administrators</option>
```

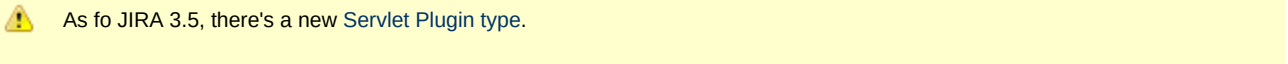
Take note of the value for the option you want to set. Then add this as the **Announcement Banner** under **Options and Settings**, and your field should come preselected.

```
<script language="JavaScript">
<!--
window.onload = function()
{
    var commentLevelSelect = document.getElementById("commentLevel");
    if (commentLevelSelect)
    {
        commentLevelSelect.value = 'group:jira-administrators';
    }
}
//-->
</script>

<style>
.alertHeader{display:none;}
</style>
```

Of course, you need to change the `commentLevelSelect.value` from `group:jira-administrators` to the one you want.

How do I write a servlet for JIRA?

How do I write a servlet for JIRA?



First, see the JIRA WebWork Plugin type. But if you can't use a plugin for some reason, you can follow these instructions:

JIRA uses Webwork, which manages things from a central servlet. To write a new .jspa page ('action'), you would:

- Edit WEB-INF/classes/actions.xml and a section for your page:

```
<action name="MyPage" alias="MyPage">
        <view name="success">/views/mypage.jsp</view>
    </action>
```

- Create the MyPage action class, eg. in 3.4-beta1 Standalone, create external-source/src/MyPage.java as shown below and run 'ant' in external-source/ to compile.

```
package com.atlassian.jira.web.action;

public class MyPage extends JiraWebActionSupport
{
        public String getName() {
                return "Fred";
        }
}
```

- Create the JSP, eg. in atlassian-jira/views/mypage.jsp:

```
<%@ taglib uri="webwork" prefix="webwork" %>
Hello there, <webwork:property value="name"/>
```

- Restart, and request /secure/MyPage.jspa

How to set a custom field value on an issue?

Confusingly, the method `setCustomFieldValue` on the `MutableIssue` does **not** actually set the custom field value of an issue. To update a custom field value on an issue, you'd need to use the method `updateValue` on the `CustomField` object itself.

Since JIRA 3.4.x the updateValue method signature has changed to:

```
/**
* Update the issue in the data store
*
* @param fieldLayoutItem for this field within this context
* @param issue Issue this field is part of
* @param modifiedValue new value to set field to. Cannot be null.
* @param issueChangeHolder an object to record any changes made to the issue by this method.
*/
void updateValue(FieldLayoutItem fieldLayoutItem, Issue issue, ModifiedValue modifiedValue,
                 IssueChangeHolder issueChangeHolder);
```

The fieldLayoutItem is used to determine whether the wiki renderer is used or not and can be null **for non-renderable fields only**. The `issueChangeHolder` is used to store and then possibly write the change history. You can simply replace this with `new DefaultIssueChangeHolder` and it should still work out.

For example, here is some code that updates a "Timestamp" non-wiki text custom field on issue "TP-14":

```
IssueManager issueManager = ComponentManager.getInstance().getIssueManager();
CustomFieldManager cfManager = ComponentManager.getInstance().getCustomFieldManager();
MutableIssue issue = issueManager.getIssueObject("TP-14");
CustomField cf = cfManager.getCustomFieldObjectByName("Timestamp");
IssueChangeHolder changeHolder = new DefaultIssueChangeHolder();
cf.updateValue(null, issue, new ModifiedValue(issue.getCustomFieldValue(cf), ""+new
java.util.Date()),changeHolder);
System.out.println(changeHolder.getChangeItems().get(0));
```

I have a GenericValue issue. How do I convert this to an Issue object?

I have a GenericValue issue. How do I convert this to an Issue object?

JIRA's API is moving slowly from representing issues as GenericValues (essentially untyped hashmaps) to the Issue interface). As such, you may find yourself having a GenericValue, but needing an Issue to pass to some more recent part of the API.

Note: often there are two methods available, one returning a GenericValue (eg. called `getIssue`) and another returning an Issue (eg. called `getIssueObject`). You may be able to avoid the conversion by picking the right method.

If you find yourself with a GenericValue, the best way to convert your issue is by declaring an IssueFactory in your object's constructor to set an `issueFactory` field in your object, and then call `issueFactory.getIssue(GenericValue)` to get the Issue.

If you're writing code that isn't managed by Pico container, and so won't have an IssueFactory passed in through the constructor, then you can call the static method IssueImpl.getIssueObject(GenericValue)

Is it possible to have a report plugin only appear for specific projects?

This isn't really possible at the moment. You could work around this by implementing the method

```
boolean showReport();
```

and utilise the fact that JIRA stores the current project in the session. For example, you can find the currently selected project ID by calling.

```
(Long) ActionContext.getSession().get(SessionKeys.SELECTED_PROJECT);
```

You can then use this in your `showReport` method to return true for false depending on the project.

Why am I getting 'Exception looking up public key' exception while building a plugin?

Why am I getting 'Exception looking up public key' exception while building a plugin?

Some plugins such as the RPC Plugin execute unit-testing while 'build' (e.g. `maven jar`) is under progress. As you may not require it sometimes, such error message may turn out to be unpleasant:

```
2006-11-02 16:41:31,375 ERROR [atlassian.license.decoder.LicenseDecoder] Exception looking up public
key: null
java.lang.NullPointerException
at com.atlassian.license.decoder.LicenseDecoder.getPublicKey(LicenseDecoder.java:51)
at com.atlassian.license.decoder.LicenseDecoder.getLicense(LicenseDecoder.java:26)
at com.atlassian.license.LicenseManager.setLicense(LicenseManager.java:127)
at com.atlassian.jira.rpc.soap.service.TestProjectServiceImpl.setUp(TestProjectServiceImpl.java:108)
at org.jmock.core.VerifyingTestCase.runBare(Unknown Source)
at junit.framework.TestResult$1.protect(TestResult.java:106)
at junit.framework.TestResult.runProtected(TestResult.java:124)
at junit.framework.TestResult.run(TestResult.java:109)
at junit.framework.TestCase.run(TestCase.java:118)
at junit.framework.TestSuite.runTest(TestSuite.java:208)
at junit.framework.TestSuite.run(TestSuite.java:203)
at org.apache.tools.ant.taskdefs.optional.junit.JUnitTestRunner.run(JUnitTestRunner.java:325)
at org.apache.tools.ant.taskdefs.optional.junit.JUnitTask.executeInVM(JUnitTask.java:848)
at org.apache.tools.ant.taskdefs.optional.junit.JUnitTask.execute(JUnitTask.java:556)
at org.apache.tools.ant.taskdefs.optional.junit.JUnitTask.execute(JUnitTask.java:532)
at org.apache.tools.ant.Task.perform(Task.java:341)
at org.apache.commons.jelly.tags.ant.AntTag.doTag(AntTag.java:185)
at org.apache.commons.jelly.impl.TagScript.run(TagScript.java:279)
at org.apache.commons.jelly.impl.ScriptBlock.run(ScriptBlock.java:135)
at org.apache.commons.jelly.TagSupport.invokeBody(TagSupport.java:233)
at org.apache.commons.jelly.tags.core.IfTag.doTag(IfTag.java:88)
at org.apache.commons.jelly.impl.TagScript.run(TagScript.java:279)
at org.apache.commons.jelly.impl.ScriptBlock.run(ScriptBlock.java:135)
at org.apache.maven.jelly.tags.werkz.MavenGoalTag.runBodyTag(MavenGoalTag.java:79)
at org.apache.maven.jelly.tags.werkz.MavenGoalTag$MavenGoalAction.performAction(MavenGoalTag.java:110)
at com.werken.werkz.Goal.fire(Goal.java:639)
at com.werken.werkz.Goal.attain(Goal.java:575)
at com.werken.werkz.Goal.attainPrecursors(Goal.java:488)
at com.werken.werkz.Goal.attain(Goal.java:573)
at com.werken.werkz.Goal.attainPrecursors(Goal.java:488)
at com.werken.werkz.Goal.attain(Goal.java:573)
at org.apache.maven.plugin.PluginManager.attainGoals(PluginManager.java:671)
at org.apache.maven.MavenSession.attainGoals(MavenSession.java:263)
at org.apache.maven.cli.App.doMain(App.java:488)
at org.apache.maven.cli.App.main(App.java:1239)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:324)
at com.werken.forehead.Forehead.run(Forehead.java:551)
at com.werken.forehead.Forehead.main(Forehead.java:581)
2006-11-02 16:41:31,375 WARN [com.atlassian.license.LicenseManager] Attempt to set invalid license.
Ensure that you are calling setLicense(license, appName) - not (appName, license)
java.lang.RuntimeException: Exception getting verification from file - possible classloader problem,
or corrupt JIRA installation
```

To have a smooth build, you may like to turn unit-testing off:

```
maven jar:install -Dmaven.test.skip=true
```

How to make a custom field "importable" for project imports

Summary

As of JIRA 3.13, you can now import single projects from an XML backup. The import includes the projects issues and all related issue data.

In order to import custom field data the CustomFieldType class that defines the custom field must implement a new interface, ProjectImportableCustomField. All the existing JIRA system custom fields and the custom fields included in supported JIRA plugins have been modified to implement this interface.

As of July 18th, 2008, the following plugins have been modified to support the new interface:

FishEye Plugin for JIRA	✗
JIRA Charting Plugin	✓
JIRA Labels Plugin	✓
JIRA Linker Plugin	✓
JIRA Toolkit	✓
JIRA Perforce Plugin	✓

You only need to be concerned about this document if you are using a custom field that may have been developed before 3.13, or without taking into account the ProjectImportableCustomField interface AND you would like this custom fields data to be imported when performing a project import.

Making a field project importable

ProjectImportableCustomField

The ProjectImportableCustomField interface is an optional interface that can be implemented when creating a JIRA CustomFieldType. If this interface is not implemented then any custom fields of this type will be ignored when performing a project import. This will in no way stop the import from proceeding but it will stop the custom field values from being included in the projects issue data.

The interface looks like this:

```
public interface ProjectImportableCustomField
{
    ProjectCustomFieldImporter getProjectImporter();
}
```

ProjectCustomFieldImporter

As you can see implementing the interface commits you to creating an instance of a ProjectCustomFieldImporter. The ProjectCustomFieldImporter is the class that does the actual work for a custom field when being imported.

A ProjectCustomFieldImporter has two jobs:

1. Validate whether or not the custom field values are relevant for the running instance of JIRA.
2. Transform the custom field values (if required) so that the values will be valid in the running instance of JIRA.

The ProjectCustomFieldImporter has two methods, canMapImportValue and getMappedImportValue:

```
MessageSet canMapImportValue(ProjectImportMapper projectImportMapper, ExternalCustomFieldValue
customFieldValue, FieldConfig fieldConfig, final I18nHelper i18n);

MappedCustomFieldValue getMappedImportValue(ProjectImportMapper projectImportMapper,
ExternalCustomFieldValue customFieldValue, FieldConfig fieldConfig);
```

canMapImportValue

The project import will run through all the custom field values that are relevant to a custom field and, if the custom field type is project importable, will first invoke the canMapImportValue. At this time the custom field needs to decide if there is a problem with the provided custom field value.

The ProjectCustomFieldImporter implementation can communicate two levels of messages, warnings and errors. An error will make it so the project import can not continue. The error message will be shown on the pre-import summary screen associated with the custom field. Error messages should be descriptive to the extent that a user can, hopefully, correct the error so that the import will eventually proceed. A warning will NOT cause the project import to stop. A warning message is used to alert the user to some aspect of the data that you may want them to know BEFORE they decide if they want to perform the import or not. The warning message will be shown on the pre-import screen.

Both warning and error messages are added to a MessageSet. One characteristic of the MessageSet is that if you add 2 or more of the exact same textual errors, they will only be reported as a single errors. For example:

Your select custom field in your running instance of JIRA does not have an option configured for the value "test option 1".

In your backup data you have 10 references to the custom field value "test option 1"

canMapImportValue will be called 10 times with this value. Each time you add an error message of "Field 'My Select Field' can not import value 'test option 1', the field does not have this option."

In the pre-import UI, this message will be shown one time under a sub-heading of the custom fields name.

If you add an error message to the message set when the canMapImportValue method is called then the getMappedImportValue will never be called.

If you are a custom field that is storing system values (e.g. a group custom field). You can also get the system to perform validation for you by alerting the system that your "system" value is required. This is done through the mappers that are passed in via the ProjectImportMapper.

Each individual mapper has a flagValueAsRequired method. Calling this records the fact that the value is a system value that **must** be present for the import to proceed. If the value is not present then the project import will report an error on the pre-import summary screen under the sub-heading of the system field (e.g. group).

For example if you were a group custom field:

```
projectImportMapper.getGroupMapper().flagValueAsRequired(groupname);
```

getMappedImportValue

This method is called once the actual import is being performed.

The main objective of this method is to "transform" the existing custom field value in any way that the ProjectCustomFieldImporter implementation deems necessary.

If, for some reason, you do not want the value to be stored then you can return a MappedCustomFieldValue with a null value.

Otherwise you should fill the MappedCustomFieldValue with the value you want stored in the running JIRA instance.

It should almost always be the case that if this method is called (i.e. canMapImportValue did not generate an error) that a MappedCustomFieldValue can be created.

ProjectImportMapper

Many custom fields rely on system information. The ProjectImportMapper gives you access to system values (statues, priorities, project roles, etc.) that the Importer has already mapped and validated. This allows you to find the ID of a system value in the running JIRA instance corresponding to the "old" ID (usually your custom field value).

Custom fields are therefore provided with a ProjectImportMapper when the canMapImportValue and getMappedImportValue methods are called. This mapper contains individual mappers for different system fields. The mapper is filled with information about the values from the backup XML data and mapped values in the running JIRA instance. These mapped values can be useful in deciding if a custom field type can map a value and what that mapping would be.

Sometimes you may want to not perform a validation check yourself, instead you want to communicate to the Project Import that the system value is "in use" or "required" and then the Project Import can validate if that system value is correct or not.

One example of where you want to let the Project Importer do validation is Users and the UserCFType.

The project importer will try to create any users that are required but do not exist in the running JIRA instance. Therefore, we want to let the system know about any users that a custom field may reference.

The UserCustomFieldImporter looks like this:

```
public MessageSet canMapImportValue(final ProjectImportMapper projectImportMapper, final
ExternalCustomFieldValue customFieldValue, final FieldConfig fieldConfig, final I18nHelper i18n)
{
    String username = customFieldValue.getValue();
    // ignore empty username including null and empty String.
if (username != null && username.length() > 0)
    {
        // Flag the username as required
projectImportMapper.getUserMapper().flagValueAsRequired(username);
        // We don't check the Mapper directly if the username can be mapped, because Users can
sometimes be automatically imported
// during the Project Import.
}
    return null;
}

public MappedCustomFieldValue getMappedImportValue(final ProjectImportMapper projectImportMapper,
final ExternalCustomFieldValue customFieldValue, final FieldConfig fieldConfig)
{
    // We don't actually map Users, we just use the same username.
return new MappedCustomFieldValue(customFieldValue.getValue());
}
```

Notice the call to projectImportMapper.getUserMapper().flagValueAsRequired(username). If you want to alert a mapper that a field value is required for the import to continue you can call this method on the provided mapper.

Most of the time, you are not storing system field values so, you should just add an error message yourself. Users are a very special case since the importer will create them if they are missing. Do not raise an error if you are a user custom field, instead flag the value as required in the UserMapper and the import will handle the creation/validation of the user for you.

One example of just using a mapper and doing the validation yourself is Project information and the ProjectCFType.

For example:

The Project custom field stores the id of a project within JIRA. When importing a project from a backup there is no guarantee that the custom fields value will make any sense in the running JIRA instance.

In the project custom fields ProjectCustomFieldImporter implementation of canMapImportValue we lookup the projectMapper and see if there is a mapped value for the custom field value (old project id):

```
projectImportMapper.getProjectMapper()
final String mappedId = projectMapper.getMappedId(valueAsIntString);
```

The ProjectImport has already done the trouble to validate and map the values found in the system mappers so the custom fields can use this information to decide if their values are valid. In the case of the project custom field if there is not a mapped ID we choose to add a warning to the message set to let the users know that the unmapped values will be dropped on import.

Not all ProjectCustomFieldImporter's will need to use the ProjectImportMapper. It may be the case that the custom field has no need of system information.

NoTransformationCustomFieldImporter

The majority of CustomFieldType's store values that have no dependency on the running JIRA instances configuration (e.g. text custom fields, date custom fields, number custom fields, etc.).

In these cases we just want to pass the existing value strait through the importer.

We have created an implementation of ProjectCustomFieldImporter that does exactly this, its called NoTransformationCustomFieldImporter. If you custom field simply wants its existing value imported into the new system then you should use this implementation.

Example: Select custom field ProjectCustomFieldImporter

The SelectCFType in JIRA stores a string which relates to a configured custom field option in JIRA. A select custom field that references a value that does not have a corresponding option in the custom field configurations will not be shown on the JIRA view issue screen.

THEREFORE, when importing select custom field values you DO NOT want to import the value if there is not a corresponding option for that select custom field.

The code for the select custom field ProjectCustomFieldImporter looks like this:

```
public class SelectCustomFieldImporter implements ProjectCustomFieldImporter
{
    private final OptionsManager optionsManager;

    public SelectCustomFieldImporter(OptionsManager optionsManager)
    {
        this.optionsManager = optionsManager;
    }

    public MessageSet canMapImportValue(final ProjectImportMapper projectImportMapper, final
ExternalCustomFieldValue customFieldValue, final FieldConfig fieldConfig, final I18nHelper i18n)
    {
        final String value = customFieldValue.getValue();
        // Get this custom field's "valid" options and see if the value is one of them
final Options options = optionsManager.getOptions(fieldConfig);
        if (options.getOptionForValue(value, null) == null)
        {
            // If an option does not exist for the value we are looking at then log an error and stop
the import
MessageSet messageSet = new MessageSetImpl();
            messageSet.addErrorMessage(i18n.getText("admin.errors.project.import
.custom.field.option.does.not.exist", fieldConfig.getCustomField().getName(), value));
            return messageSet;
        }
        return null;
    }

    public MappedCustomFieldValue getMappedImportValue(final ProjectImportMapper projectImportMapper,
final ExternalCustomFieldValue customFieldValue, final FieldConfig fieldConfig)
    {
        // Since this method will never be called without a successful call to canMap we can safely
just pass the
// value back out.
return new MappedCustomFieldValue(customFieldValue.getValue());
    }
}
```

Take careful notice of how in the canMapImportValue method we use the provided FieldConfig (this provides the "context" that the custom field is being used in) to find the configured Options for this custom field.

We next try to find the option that corresponds to the string custom field value we are provided. If we do not find this option in the running JIRA instance we will add an internationalized error message (please note the I18nHelper provided is the I18nHelper configured for your custom field, so if you want to provide an I18n'ed message you can do so via normal methods in your plugin descriptor).

Notice too that once getMappedImportValue is called all we have to do is pass back the existing value since we can be sure that the canMapImportValue has already succeeded.

Now we need to wire our ProjectCustomFieldImporter with our CustomFieldType:

```
public class SelectCFType extends TextCFType implements MultipleSettableCustomFieldType,
MultipleCustomFieldType,
                                                        SortableCustomField, GroupSelectorField,
ProjectImportableCustomField
{

    private final SelectConverter selectConverter;
    private final OptionsManager optionsManager;
    private final ProjectCustomFieldImporter projectCustomFieldImporter;

    private static final Logger log = Logger.getLogger(SelectCFType.class);

    public SelectCFType(CustomFieldValuePersister customFieldValuePersister, StringConverter
stringConverter,
                        SelectConverter selectConverter, OptionsManager optionsManager,
GenericConfigManager genericConfigManager)
    {
        super(customFieldValuePersister, stringConverter, genericConfigManager);
        this.selectConverter = selectConverter;
        this.optionsManager = optionsManager;
        this.projectCustomFieldImporter = new SelectCustomFieldImporter(this.optionsManager);
    }

    public ProjectCustomFieldImporter getProjectImporter()
    {
        return this.projectCustomFieldImporter;
    }
```

Existing ProjectCustomFieldImporter Implementations

These implementations have been created and are used by various JIRA system custom fields:

- NoTransformationCustomFieldImporter
- SelectCustomFieldImporter
- CascadingSelectCustomFieldImporter
- GroupCustomFieldImporter
- ProjectPickerCustomFieldImporter
- UserCustomFieldImporter
- VersionCustomFieldImporter

Conclusion

If you want your custom field to participate in project imports you will need to modify your existing custom field code, hopefully this document has provided enough information to help you do this.

If your custom field type is extending a JIRA system custom field type you may already inherit an implementation of ProjectImportableCustomField.

Please make sure that this is the right implementation for your custom field type.

JIRA Developer Forums

The JIRA Developer Forum is a place for the discussion of extending and customising JIRA. There are two ways to join the discussion:

- Read the web-based forum
- Join the mailing-list

Forum Guidelines

The developer forum is set up for the discussion of:

- JIRA development: Plugins, themes or JIRA source customisation

- JIRA's internal and remote APIs
- Automation of tasks in JIRA
- Announcement of new JIRA developer releases
- Announcement of new plugin releases
- Requests for plugins or customisation services
- Complaints that JIRA have broken the plugin APIs *again*

The following are discouraged:

- Requests for support setting up or running JIRA should be directed to http://support.atlassian.com
- Bug-reports should be submitted to http://jira.atlassian.com (If the bug is specific to plugin development or JIRA's internal APIs, you can discuss it on the developer forum, but **PLEASE** submit a bug as well!)
- Questions about using, running or administering JIRA should be directed to The general JIRA forum
- It's OK to respond to requests for professional services on the forum, or to plug your business in plugin announcements or your signature, but please don't just post advertisements.

The Developer FAQ

Some questions come up on the forum a lot. Make sure you've checked the JIRA Developer FAQ first.

About the Participants

When taking part in the forum, please keep in mind that JIRA licenses do **not** include a guaranteed level of developer support. The JIRA development team follows and contributes to the forum because it's important to us to have a healthy ecosystem of third-party developers, and because we love seeing the cool stuff that our customers and partners come up with. That said, we can't respond to every request, and developers are also expected to help themselves by reading the available documentation, the API, and occasionally even looking at the JIRA source-code.

Also keep in mind that a lot of the people on the list don't work for Atlassian at all, and are answering questions because they're nice people.

JIRA Tutorial Videos

This page contains tutorial videos on a number of basic JIRA functions. The videos are intended to **supplement, not replace,** the online JIRA documentation.

> **Tutorial Videos — Version and Edition**
> The JIRA tutorial videos are not updated with every JIRA release and edition. Please note the **JIRA version** and **JIRA edition** of the tutorial video before watching.
>
> While there are commonalities between different versions and different editions, you may find that the example scenarios in the tutorials differ from your JIRA instance, if you are using a different version or edition.

The videos are generally recorded in Quicktime (.mov) format. If any tutorials are available in other formats, they will be listed below.

On this page:

- Installing JIRA Standalone (Windows)
- Installing JIRA Standalone (Mac)
- Adding a Project
- Adding a Custom Field
- Creating Comments and Issues via Email
- Issue Security Scheme Overview
- Permission Scheme Overview
- Workflow Scheme Overview
- Creating SLA Issue Filters

Installing JIRA Standalone (Windows)

JIRA Version	**3.13**
JIRA Edition	**Enterprise**

Video (Flash)	 *(click image to watch video)*
Related Documentation	* Installing JIRA Standalone

*Please note the **version** and **edition** of the tutorial video before watching.*

Installing JIRA Standalone (Mac)

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (Flash)	 *(click image to watch video)*
Related Documentation	* Installing JIRA Standalone

*Please note the **version** and **edition** of the tutorial video before watching.*

Adding a Project

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (.mov)	 *(click image to watch video)*
Related Documentation	* What is a Project? * Defining a Project

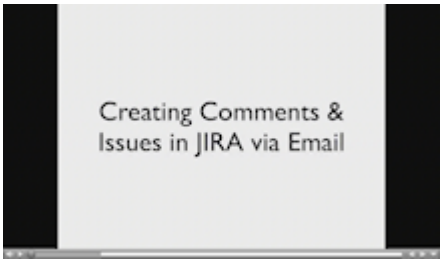
*Please note the **version** and **edition** of the tutorial video before watching.*

Adding a Custom Field

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (.mov)	 *(click image to watch video)*
Related Documentation	* Adding a Custom Field

*Please note the **version** and **edition** of the tutorial video before watching.*

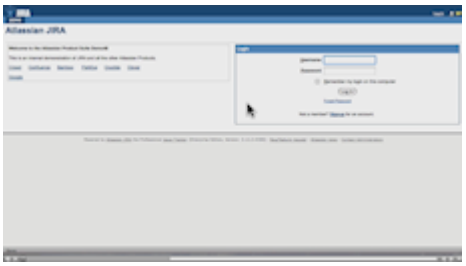
Creating Comments and Issues via Email

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (.avi)	 *(click image to watch video)*
Video (.m4v)	 *(click image to watch video)*
Related Documentation	* Creating Issues and Comments from Email

*Please note the **version** and **edition** of the tutorial video before watching.*

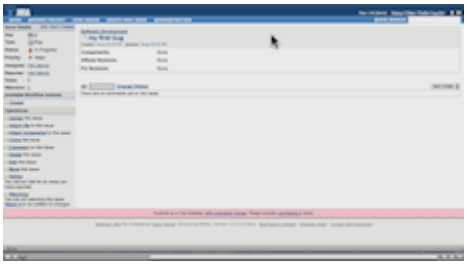
Issue Security Scheme Overview

JIRA Version	**3.13**
JIRA Edition	**Enterprise**

Video (.mov)	 *(click image to watch video)*
Related Documentation	* Configuring Issue Level Security

*Please note the **version** and **edition** of the tutorial video before watching.*

Permission Scheme Overview

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (.mov)	 *(click image to watch video)*
Related Documentation	* Managing Project Permissions

*Please note the **version** and **edition** of the tutorial video before watching.*

Workflow Scheme Overview

JIRA Version	**3.13**
JIRA Edition	**Enterprise**
Video (.mov)	 *(click image to watch video)*
Related Documentation	* Activating Workflow

*Please note the **version** and **edition** of the tutorial video before watching.*

Creating SLA Issue Filters

JIRA Version	**3.13**

JIRA Edition	Enterprise
Video (.mov)	 *(click image to watch video)*
Related Documentation	* Saving Searches ('Issue Filters') * JIRA as a Support System

*Please note the **version** and **edition** of the tutorial video before watching.*

Local JIRA documentation

On this page:

- Why would I set up local online documentation?
- How to set up local online documentation
  - For JIRA 4.0.x and later:
  - For JIRA 3.13.x and earlier:
- Local field documentation

Why would I set up local online documentation?

You may wish to run the documentation locally, and have JIRA link to it. There are a few reasons you may wish to do this:

- JIRA's interface contains links ( ) to help pages, some to pages within JIRA, but many to the online documentation on www.atlassian.com. For deployments in environments without an internet connection, a local copy of the documentation is desirable.
- If you have customised JIRA, you may wish to update the documentation to reflect your changes, or add new pages.
- You can change the look and feel of the documentation to integrate into your company's intranet.

How to set up local online documentation

For JIRA 4.0.x and later:

1. Install Atlassian Confluence. (If you don't already have Confluence, ask for a free Evaluation License. You can use 'Anonymous' access to allow your users to view the documentation.)
2. Download the JIRA Documentation's XML source. Note that the Confluence version of the XML source needs to be the same major Confluence version as your local Confluence site.
3. Import the XML file into your Confluence site. (Note: if there is already a 'JIRA' space in your Confluence site, it will be overwritten.) For detailed instructions, see the Confluence documentation on Restoring a Space.
4. *(Optional)* If you want JIRA's help links ( ) to point to your local documentation, you will need to:
   a. edit JIRA's `/WEB-INF/classes/help-paths.properties` file and change the `url-prefix` line so that it points to the 'JIRA' space in your local Confluence site, e.g.:

      ```
      url-prefix=http://confluence.mycompany.com/display/JIRA/
      ```

   b. restart JIRA.

For JIRA 3.13.x and earlier:

JIRA licensees can download the XML source for the documentation.

To build JIRA's docs locally:

1. Download Apache Forrest 0.5.1 (zip, tar.gz), used to render the docs.
2. Download the JIRA Documentation's XML source (6.2Mb). Note: the download is restricted to JIRA license holders.

3. Follow the JIRA_DOCUMENTATION.txt instructions in the package.

Local field documentation

If you just want to document usage of a custom field, please see Creating Help for a Custom Field.