# Bamboo 101

## Bamboo Administrator's Guide

### 3.2 Configuring Projects and Plans

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1.1 About Projects, Plans and Builds</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1.2 Creating a Plan</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1.3 Specifying a Plan's Details</td>
<td>24</td>
</tr>
<tr>
<td>3.2.1.2.2 Specifying a Plan's Source Repository</td>
<td>25</td>
</tr>
<tr>
<td>3.2.1.2.3 Specifying a Plan's Builder</td>
<td>34</td>
</tr>
<tr>
<td>3.2.1.2.4 Specifying a Plan's Capability Requirements</td>
<td>53</td>
</tr>
<tr>
<td>3.2.1.2.5 Specifying a Plan's Build Artifacts</td>
<td>54</td>
</tr>
<tr>
<td>3.2.1.2.6 Specifying a Plan's Notifications</td>
<td>55</td>
</tr>
<tr>
<td>3.2.1.2.7 Specifying a Plan's Post Actions</td>
<td>58</td>
</tr>
<tr>
<td>3.2.1.2.8 Specifying a Plan's Permissions</td>
<td>62</td>
</tr>
<tr>
<td>3.2.1.3.1 Editing a Plan</td>
<td>65</td>
</tr>
<tr>
<td>3.2.1.4 Deleting or deleting a Plan</td>
<td>66</td>
</tr>
<tr>
<td>3.2.1.4.1 Deleting a Build Result</td>
<td>67</td>
</tr>
<tr>
<td>3.2.1.4.2 Deleting a Plan's Working Files</td>
<td>67</td>
</tr>
<tr>
<td>3.2.1.4.3 Stopping an Active Build</td>
<td>67</td>
</tr>
<tr>
<td>3.2.1.5 Moving a Plan to a different Project</td>
<td>68</td>
</tr>
<tr>
<td>3.2.1.6 Renaming a Plan or Project</td>
<td>69</td>
</tr>
<tr>
<td>3.2.1.7 Modifying Multiple (Bulk) Plans</td>
<td>70</td>
</tr>
</tbody>
</table>

### 3.2.2 Configuring Agents and Capabilities

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2.1 About Agents and Capabilities</td>
<td>73</td>
</tr>
<tr>
<td>3.2.2.2 Configuring Agents</td>
<td>76</td>
</tr>
<tr>
<td>3.2.2.2.1 Creating a Local Agent</td>
<td>77</td>
</tr>
<tr>
<td>3.2.2.2.2 Creating a Remote Agent</td>
<td>78</td>
</tr>
<tr>
<td>3.2.2.2.3 Editing an Agent's Details</td>
<td>82</td>
</tr>
<tr>
<td>3.2.2.2.4 Disabling or Deleting an Agent</td>
<td>83</td>
</tr>
<tr>
<td>3.2.2.5 Viewing an Agent</td>
<td>83</td>
</tr>
<tr>
<td>3.2.2.6 Monitoring Agent Status</td>
<td>88</td>
</tr>
</tbody>
</table>

### 3.2.3 Configuring Capabilities

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2.3.1 Configuring a new Builder</td>
<td>89</td>
</tr>
<tr>
<td>3.2.2.3.2 Configuring a new JDK</td>
<td>89</td>
</tr>
<tr>
<td>3.2.2.3.3 Configuring a new Custom Capability</td>
<td>96</td>
</tr>
<tr>
<td>3.2.2.3.4 Configuring a new Perforce Capability</td>
<td>97</td>
</tr>
<tr>
<td>3.2.2.3.5 Editing a Capability</td>
<td>98</td>
</tr>
<tr>
<td>3.2.2.3.6 Renaming a Capability</td>
<td>101</td>
</tr>
<tr>
<td>3.2.2.3.7 Deleting a Capability</td>
<td>102</td>
</tr>
<tr>
<td>3.2.2.3.8 Viewing the Agents &amp; Plans Related to a Capability</td>
<td>102</td>
</tr>
</tbody>
</table>

### 3.2.4 Working with Builds

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.4.1 About Elastic Bamboo</td>
<td>104</td>
</tr>
<tr>
<td>3.2.4.1.1 Elastic Bamboo Costs</td>
<td>104</td>
</tr>
<tr>
<td>3.2.4.2 Getting Started with Elastic Bamboo</td>
<td>105</td>
</tr>
<tr>
<td>3.2.4.2.1 Generating your AWS Private Key File and Certificate File</td>
<td>106</td>
</tr>
<tr>
<td>3.2.4.2.2 Configuring Elastic Instances to use EBS</td>
<td>107</td>
</tr>
<tr>
<td>3.2.4.4 Managing Elastic Bamboo</td>
<td>108</td>
</tr>
<tr>
<td>3.2.4.4.1 Managing your Elastic Images</td>
<td>109</td>
</tr>
<tr>
<td>3.2.4.4.2 Managing your Elastic Instances</td>
<td>109</td>
</tr>
<tr>
<td>3.2.4.4.3 Managing your Elastic Agents</td>
<td>109</td>
</tr>
<tr>
<td>3.2.4.5 Running Builds using Elastic Bamboo</td>
<td>110</td>
</tr>
<tr>
<td>3.2.4.6 Disabling Elastic Bamboo</td>
<td>110</td>
</tr>
<tr>
<td>3.2.5 Configuring Email and Instant Messaging Notifications</td>
<td>111</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Adding or Removing Notifications for a Plan</td>
</tr>
<tr>
<td></td>
<td>3.2.5.2 Modifying Notification Templates</td>
</tr>
<tr>
<td></td>
<td>3.2.5.2.1 Freemarker and Notification Templates</td>
</tr>
<tr>
<td></td>
<td>3.2.5.3 Configuring Bamboo to send SMTP Email</td>
</tr>
<tr>
<td></td>
<td>3.2.5.4 Configuring Bamboo to use Instant Messaging (IM)</td>
</tr>
<tr>
<td></td>
<td>3.2.5.4.1 Configuring Bamboo to use Google Talk for Instant Messaging</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Managing Users and Permissions</td>
</tr>
<tr>
<td></td>
<td>3.2.6.1 Managing Users</td>
</tr>
<tr>
<td></td>
<td>3.2.6.1.1 Creating a User</td>
</tr>
<tr>
<td></td>
<td>3.2.6.1.2 Changing a User’s Password or Details</td>
</tr>
<tr>
<td></td>
<td>3.2.6.1.3 Deleting or deactivating a User</td>
</tr>
<tr>
<td></td>
<td>3.2.6.1.4 Granting Administration Rights to a User</td>
</tr>
<tr>
<td></td>
<td>3.2.6.2 Managing Groups</td>
</tr>
<tr>
<td></td>
<td>3.2.6.2.1 Creating a Group</td>
</tr>
<tr>
<td></td>
<td>3.2.6.2.2 Deleting a Group</td>
</tr>
<tr>
<td></td>
<td>3.2.6.3 Adding Users to and removing them from Groups</td>
</tr>
<tr>
<td></td>
<td>3.2.6.3.1 Granting Plan Permissions to Users or Groups</td>
</tr>
<tr>
<td></td>
<td>3.2.6.3.2 Granting Global Permissions to Users or Groups</td>
</tr>
<tr>
<td></td>
<td>3.2.6.3.3 Allowing Anonymous Users to access Bamboo</td>
</tr>
<tr>
<td></td>
<td>3.2.6.4 Enabling or disabling Contact Details Display</td>
</tr>
<tr>
<td></td>
<td>3.2.6.5 Enabling or disabling Public Group</td>
</tr>
<tr>
<td></td>
<td>3.2.6.6 Working with External User Repositories</td>
</tr>
<tr>
<td></td>
<td>3.2.6.6.1 Integrating Bamboo with Crowd</td>
</tr>
<tr>
<td></td>
<td>3.2.6.6.2 Integrating Bamboo with LDAP</td>
</tr>
<tr>
<td>3.2.7</td>
<td>Managing Data and Backups</td>
</tr>
<tr>
<td></td>
<td>3.2.7.1 Locating Important Directories and Files</td>
</tr>
<tr>
<td></td>
<td>3.2.7.2 Specifying Bamboo’s Working Directory</td>
</tr>
<tr>
<td></td>
<td>3.2.7.3 Viewing your Database Connection Details</td>
</tr>
<tr>
<td></td>
<td>3.2.7.4 Moving your Bamboo Databases to a different Database</td>
</tr>
<tr>
<td></td>
<td>3.2.7.5 Optimising or Re-indexing Data</td>
</tr>
<tr>
<td></td>
<td>3.2.7.6 Enabling Expiry of Build Results</td>
</tr>
<tr>
<td></td>
<td>3.2.7.7 Specifying a Backup Schedule</td>
</tr>
<tr>
<td></td>
<td>3.2.7.8 Exporting Data for Backup</td>
</tr>
<tr>
<td></td>
<td>3.2.7.9 Importing Data from Backup</td>
</tr>
<tr>
<td>3.2.8</td>
<td>Configuring System Settings</td>
</tr>
<tr>
<td></td>
<td>3.2.8.1 Viewing Bamboo’s System Information</td>
</tr>
<tr>
<td></td>
<td>3.2.8.2 Updating your Bamboo License Details</td>
</tr>
<tr>
<td></td>
<td>3.2.8.3 Specifying Bamboo’s Title</td>
</tr>
<tr>
<td></td>
<td>3.2.8.4 Specifying Bamboo’s URL</td>
</tr>
<tr>
<td></td>
<td>3.2.8.5 Configuring Logging in Bamboo</td>
</tr>
<tr>
<td></td>
<td>3.2.8.6 Enabling GZIP Compression</td>
</tr>
<tr>
<td></td>
<td>3.2.8.7 Enabling Bamboo’s Remote API</td>
</tr>
<tr>
<td></td>
<td>3.2.8.8 Configuring System Properties</td>
</tr>
<tr>
<td>3.2.9</td>
<td>Configuring Bamboo Security</td>
</tr>
<tr>
<td></td>
<td>3.2.9.1 Configuring OAuth Consumers</td>
</tr>
<tr>
<td></td>
<td>3.2.9.2 Elastic Bamboo Security</td>
</tr>
<tr>
<td>3.2.10</td>
<td>Configuring Plugins</td>
</tr>
<tr>
<td></td>
<td>3.2.10.1 About Bamboo Plugins</td>
</tr>
<tr>
<td></td>
<td>3.2.10.2 Enabling the ‘Auto-Favourites’ Plugin</td>
</tr>
<tr>
<td></td>
<td>3.2.10.3 Enabling the Clover Plugin</td>
</tr>
<tr>
<td></td>
<td>3.2.11 Using Bamboo with Other Applications</td>
</tr>
<tr>
<td></td>
<td>3.2.11.1 Embedding Bamboo into Other Applications</td>
</tr>
<tr>
<td></td>
<td>3.2.11.1.1 Javascript Widgets</td>
</tr>
<tr>
<td></td>
<td>3.2.11.2 Integrating Bamboo with Other Applications</td>
</tr>
<tr>
<td></td>
<td>3.2.11.2.1 Integrating Bamboo with JIRA</td>
</tr>
<tr>
<td>3.3</td>
<td>Bamboo Installation &amp; Upgrade Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.1 Installing Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.1.1 Bamboo EAR-WAR Installation Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.1.2 Bamboo Remote Agent Installation Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.1.2.1 Legacy Remote Agent Installation Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.1.3 Bamboo Standalone Installation Guide (Linux)</td>
</tr>
<tr>
<td></td>
<td>3.3.1.4 Bamboo Standalone Installation Guide (Mac)</td>
</tr>
<tr>
<td></td>
<td>3.3.1.5 Bamboo Standalone Installation Guide (Windows)</td>
</tr>
<tr>
<td></td>
<td>3.3.1.6 Running the Setup Wizard</td>
</tr>
<tr>
<td></td>
<td>3.3.1.6.1 Connecting Bamboo to an external database</td>
</tr>
<tr>
<td></td>
<td>3.3.1.6.2 Using Bamboo’s embedded HSQL database</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Bamboo Release Notes</td>
</tr>
<tr>
<td></td>
<td>3.3.2.1 Bamboo 2.4 Release Notes</td>
</tr>
<tr>
<td></td>
<td>3.3.2.1.1 Bamboo 2.4 Upgrade Guide</td>
</tr>
<tr>
<td></td>
<td>3.3.2.2 Bamboo 2.3 Release Notes</td>
</tr>
<tr>
<td></td>
<td>3.3.2.2.1 Bamboo 2.3.1 Release Notes</td>
</tr>
<tr>
<td></td>
<td>3.3.2.2.2 Bamboo 2.3 Upgrade Guide</td>
</tr>
</tbody>
</table>
3.3.2.3 Bamboo 2.2 Release Notes ................................................................. 256
  3.3.2.3.1 Bamboo 2.2 Upgrade Guide ....................................................... 264
  3.3.2.3.2 Bamboo 2.2.4 Release Notes ...................................................... 265
  3.3.2.3.3 Bamboo 2.2.3 Release Notes ...................................................... 266
  3.3.2.3.4 Bamboo 2.2.2 Release Notes ...................................................... 267
  3.3.2.3.5 Bamboo 2.2.1 Release Notes ...................................................... 268
3.3.2.4 Bamboo 2.1 Release Notes ................................................................. 270
  3.3.2.4.1 Bamboo 2.1 Upgrade Guide ....................................................... 275
  3.3.2.4.2 Bamboo 2.1.5 Release Notes ...................................................... 276
  3.3.2.4.3 Bamboo 2.1.4 Release Notes ...................................................... 278
  3.3.2.4.4 Bamboo 2.1.3 Release Notes ...................................................... 279
  3.3.2.4.5 Bamboo 2.1.2 Release Notes ...................................................... 280
  3.3.2.4.6 Bamboo 2.1.1 Release Notes ...................................................... 282
3.3.2.5 Bamboo 2.0 Release Notes ................................................................. 283
  3.3.2.5.1 Bamboo 2.0 Upgrade Guide ....................................................... 287
  3.3.2.5.2 Bamboo 2.0.6 Release Notes ...................................................... 290
  3.3.2.5.3 Bamboo 2.0.5 Release Notes ...................................................... 292
  3.3.2.5.4 Bamboo 2.0.4 Release Notes ...................................................... 293
  3.3.2.5.5 Bamboo 2.0.3 Release Notes ...................................................... 294
  3.3.2.5.6 Bamboo 2.0.2 Release Notes ...................................................... 295
  3.3.2.5.7 Bamboo 2.0.1 Release Notes ...................................................... 296
3.3.2.6 Bamboo 2.0 Beta Release Notes ......................................................... 298
  3.3.2.6.1 Bamboo 2.0 Beta Upgrade Guide ................................................. 301
  3.3.2.6.2 Bamboo 2.0 Beta 9 Release Notes .............................................. 304
  3.3.2.6.3 Bamboo 2.0 Beta 8 Release Notes .............................................. 306
  3.3.2.6.4 Bamboo 2.0 Beta 6 Release Notes .............................................. 308
  3.3.2.6.5 Bamboo 2.0 Beta 5 Release Notes .............................................. 310
  3.3.2.6.6 Bamboo 2.0 Beta 4 Release Notes .............................................. 311
  3.3.2.6.7 Bamboo 2.0 Beta 3 Release Notes .............................................. 313
  3.3.2.6.8 Bamboo 2.0 Beta 2 Release Notes .............................................. 315
  3.3.2.6.9 Bamboo 2.0 Beta 1 Release Notes .............................................. 316
3.3.2.7 Bamboo 1.2 Release Notes ................................................................. 327
  3.3.2.7.1 Bamboo 1.2 Upgrade Guide ....................................................... 331
  3.3.2.7.2 Bamboo 1.2 Plugin Interface Changes ......................................... 332
  3.3.2.7.3 Bamboo 1.2.4 Release Notes ...................................................... 333
  3.3.2.7.4 Bamboo 1.2.3 Release Notes ...................................................... 335
  3.3.2.7.5 Bamboo 1.2.2 Release Notes ...................................................... 337
  3.3.2.7.6 Bamboo 1.2.1 Release Notes ...................................................... 338
3.3.2.8 Bamboo 1.1 Release Notes ................................................................. 338
  3.3.2.8.1 Bamboo 1.1 Upgrade Guide ....................................................... 343
  3.3.2.8.2 Bamboo 1.1.2 Release Notes ...................................................... 344
  3.3.2.8.3 Bamboo 1.1.1 Release Notes ...................................................... 345
3.3.2.9 Bamboo 1.0 Release Notes ................................................................. 346
  3.3.2.9.1 Bamboo 1.0 Upgrade Guide ....................................................... 347
  3.3.2.9.2 Bamboo 1.0.5 Release Notes ...................................................... 347
  3.3.2.9.3 Bamboo 1.0.4 Release Notes ...................................................... 348
  3.3.2.9.4 Bamboo 1.0.3 Release Notes ...................................................... 349
  3.3.2.9.5 Bamboo 1.0.2 Release Notes ...................................................... 350
  3.3.2.9.6 Bamboo 1.0.1 Release Notes ...................................................... 351
3.3.2.10 Bamboo 1.0-Beta Release Notes ....................................................... 352
  3.3.2.10.1 Bamboo 1.0-Beta Upgrade Guide .............................................. 354
  3.3.2.10.2 Bamboo Release Summary ....................................................... 354
3.3.3 Bamboo Upgrade Guides ................................................................. 356
  3.3.3.1 Bamboo Generic Upgrade Guide ..................................................... 357
3.3.4 Bamboo Security Advisories ................................................................. 359
  3.3.4.1 Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta) .................. 361
  3.3.4.1.1 Securing your Remote Agents .................................................. 361
  3.3.4.2 Bamboo Security Advisory 2009-03-09 ........................................... 364
3.4 Bamboos User’s Guide ................................................................. 366
  3.4.1 Getting Started ................................................................. 369
  3.4.1.1 Using the Bamboo Dashboard ..................................................... 369
  3.4.1.2 Viewing Bamboos’s Current Activity ............................................ 370
  3.4.1.3 Viewing your Latest Build Results .............................................. 371
  3.4.1.4 Working with Favourites .............................................................. 372
  3.4.1.4.1 Adding a Plan to your Favourites ............................................ 372
  3.4.1.4.2 Removing a Plan from your Favourites ....................................... 373
  3.4.1.5 Displaying a Build Monitor .......................................................... 374
  3.4.1.6 Viewing Bamboos’s Agents ........................................................... 375
3.4.2 Working with Projects and Plans ................................................... 377
  3.4.2.1 Viewing a Plan’s Activity Log ..................................................... 377
  3.4.2.2 About Projects and Plans ............................................................. 378
  3.4.2.3 Viewing a Plan’s Details .............................................................. 379
  3.4.2.4 Viewing the JIRA Issues linked to the Builds in a Build Plan .......... 382
  3.4.2.5 Viewing the Clover Coverage Summary for a Build Plan ............... 383
3.4.3 Working with Build Results ................................................................. 384
3.4.3.1 About Builds and Build Results ......................................................... 384
3.4.3.2 Viewing a Build Result ......................................................................... 385
3.4.3.3 Viewing the Code Changes that triggered a Build .............................. 386
3.4.3.4 Viewing a Build's Artifacts .................................................................. 386
3.4.3.5 Viewing a Build Log ............................................................................ 387
3.4.3.6 Viewing the Metadata for a Build Result ........................................... 388
3.4.3.7 Viewing the Clover Code-Coverage for a Build Result ....................... 389
3.4.3.8 Viewing the JIRA Issues for a Build Result .......................................... 390
3.4.4 Working with Tests .................................................................................. 391
3.4.4.1 Viewing Test Results for a Build ....................................................... 393
3.4.4.2 Viewing a Test's History ........................................................................ 394
3.4.4.3 Viewing Test Statistics for a Plan ....................................................... 395
3.4.5 Reporting on Plan Trends ......................................................................... 398
3.4.5.1 Viewing Build Statistics for a Plan .................................................... 398
3.4.5.2 Generating Reports across multiple Plans ......................................... 401
3.4.5.2.1 'Build Activity' Report ................................................................. 402
3.4.5.2.2 'Build Duration' Report ................................................................ 403
3.4.5.2.3 'Build Queue Duration' Report ..................................................... 404
3.4.5.2.4 'Clover Code Coverage' Report .................................................... 405
3.4.5.2.5 'Clover Lines of Code' Report ....................................................... 406
3.4.5.2.6 'Number of Build Failures' Report ............................................... 407
3.4.5.2.7 'Number of Tests' Report ............................................................. 408
3.4.5.2.8 'Percentage of Successful Builds' Report ....................................... 409
3.4.5.2.9 'Time to Fix' Report ....................................................................... 410
3.4.6 Reporting on Author Trends ................................................................. 411
3.4.6.1 Viewing Build Statistics for all Authors ............................................ 411
3.4.6.2 Viewing Build Results for an Author ................................................. 412
3.4.6.3 Generating Reports on selected Authors ............................................ 414
3.4.6.3.1 'Build Activity per Author' Report ............................................... 415
3.4.6.3.2 'Number of Build Failures per Author' Report .............................. 416
3.4.6.3.3 'Number of Builds Broken per Author' Report ............................. 417
3.4.6.3.4 'Number of Builds Fixed per Author' Report ............................... 418
3.4.6.3.5 'Percentage of Successful Builds per Author' Report.................... 419
3.4.7 Working with Comments ......................................................................... 419
3.4.7.1 About Comments ................................................................................. 419
3.4.7.2 Commenting about a Build Result .................................................... 420
3.4.7.3 Viewing Comments about a Build Result .......................................... 420
3.4.7.4 Viewing Code Check-in Comments .................................................. 422
3.4.8 Working with Labels ................................................................................ 422
3.4.8.1 About Labels ......................................................................................... 422
3.4.8.2 Labelling a Build Result ....................................................................... 422
3.4.8.3 Removing a Label from a Build Result .............................................. 423
3.4.8.4 Viewing Labelled Build Results ....................................................... 423
3.4.8.5 Viewing Popular Labels ...................................................................... 424
3.4.9 Subscribing to RSS Feeds ........................................................................ 424
3.4.9.1 Subscribing to an RSS Feed for All Build Results for All Plans ........ 425
3.4.9.2 Subscribing to an RSS Feed for All Build Results for a Particular Plan .. 425
3.4.9.3 Subscribing to an RSS Feed for Failed Builds for All Plans ................ 425
3.4.9.4 Subscribing to an RSS Feed for Failed Builds for a Particular Plan .... 425
3.4.9.5 Subscribing to an RSS Feed for Labelled Build Results .................... 425
3.4.10 Working with Instant Messenger (IM) Notifications .............................. 426
3.4.10.1 About Instant Messenger (IM) Notifications ...................................... 426
3.4.10.2 Labelling a Build Result via IM .......................................................... 426
3.4.10.3 Commenting about a Build Result via IM ......................................... 427
3.4.11 Managing your User Profile .................................................................... 428
3.4.11.1 Changing your Personal Details ....................................................... 428
3.4.11.2 Viewing your Notifications ............................................................... 428
3.4.11.3 Changing your Password ................................................................. 429
3.4.11.4 Changing your Notification Preferences ......................................... 429
3.4.11.5 Associating your Author Name with your User Profile .................... 430
3.5 Bamboo Development Hub ......................................................................... 430
3.5.1 Bamboo Plugin Guide ............................................................................. 432
3.5.1.1 Bamboo Plugin Module Types ............................................................ 435
3.5.1.1.1 Bamboo Event Listener Module .................................................. 436
3.5.1.1.2 Build Agent Requirement Filter Module ....................................... 437
3.5.1.1.3 Command Decorator Module ....................................................... 438
3.5.1.1.4 Notification Recipient Module ...................................................... 439
3.5.1.1.5 Notification Type Module ............................................................. 440
3.5.1.1.6 Builder Plugin Module .................................................................. 443
3.5.1.1.7 Source Repository Module ......................................................... 444
3.5.1.1.8 Trigger Reason Module ............................................................... 446
3.5.1.1.9 Pre Build Queued Action .............................................................. 447
3.5.1.1.10 Pre Build Action Module ............................................................ 448
3.5.1.1.11 Build Processor Module ............................................................. 450
3.8.34 queue ................................................................. 579
3.8.35 reason ................................................................. 579
3.8.36 remote agent ......................................................... 580
3.8.37 remote agent supervisor ......................................... 580
3.8.38 requirement .......................................................... 580
3.8.39 shared capability .................................................. 580
3.8.40 triggering ............................................................. 580
3.8.41 watcher ............................................................... 581

4. Diagrams ................................................................. 581
  4.1 How are builds distributed to agents? ........................................ 581
  4.2 How do capabilities affect the distribution of builds to agents? ............. 582

5. TreeNavigation .......................................................... 583

6. TreeNavigationVersions ................................................... 584
InclusionsLibrary

The children of this page contain information which is included in other pages and sometimes even in other spaces. This is a library of re-usable information chunks.

If you want to change any of these pages, be aware that:

- Changing page names is problematic — you will need to change all the {include} and {excerpt-include} macros manually.
- The content is used in many places — make sure your change is generic enough to fit the contexts in which the pages are used.

IDE Connectors

Use the Atlassian Connector for Eclipse or the Atlassian Connector for IntelliJ IDEA to work with your Bamboo builds right there in your development environment. Do you use JIRA, Crucible or FishEye too? With the connector you can manage your issues 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.

newreleaseBamboo

_Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide._

_Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!_

downloadBambooButton

_about Integrating Bamboo with JIRA

Integrating Bamboo with Atlassian's JIRA combines Bamboo's continuous integration capabilities with your issue tracker to give you a unified view of your software development project. Using JIRA and Bamboo together, you can see which issues are being actively coded, which builds have run for an issue, find the build that fixed the issue, download your distribution and much more.

Flagship Image
Flagship image encapsulates functionality

The 'flagship' image should be the best representation of a particular set of functionality. This image will be used on an important page in the doc space which describes that functionality. The image will also be used in other spaces, such as the ATLAS space. This is a way to ensure that the images displayed in other spaces are kept up to date. The assumption is that the flagship image will be updated whenever necessary, as part of the normal documentation updates at each software release. (That's why the image should be used in a prominent place.

To use an image from this page:

!Flagship Image^ImageName.png!

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Creator (Last Modifier)</th>
<th>Creation Date</th>
<th>Last Mod Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo-BuildResults-Issues.png</td>
<td>52 kB</td>
<td>Andrew Lui [Atlassian Technical Writer] (modified by Andrew Lui [Atlassian Technical Writer])</td>
<td>Sep 12, 2008</td>
<td>Sep 12, 2008</td>
<td></td>
</tr>
</tbody>
</table>

_planBuilderCommonOptions

Update the following general build parameters:

- **System Environment Variables** *(Optional)* — Specify any additional* operating system environment variables you want to pass to your build; Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).

- **Working Sub Directory** *(Optional)* — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.

- **The build will produce test results** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
  - **Look in the standard test results directory** — Select this option if Bamboo should look in the Builder's standard test results directory.
  - **Specify custom results directories** — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
    - **Specify custom results directories** — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as */test-reports/.*.xml. Please specify file path relative to your plan's root directory (e.g /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.

- **NCover output will be produced** — Do not check this option. NCover is not relevant to Ant builders.

- **Override default build hanging detection** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
  - **Build Time Multiplier** — This setting is used to calculate the 'Expected Build Time' for the build, i.e. ‘Expected Build Time’ = ‘Build Time Multiplier’ times ‘Average Build Time’ (note, the ‘Average Build Time’ is calculated by Bamboo by using an average of previous build times).
  - **Log Quiet Time** — This is the amount of time since Bamboo last recorded an entry in the build log for a build.

  The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.

- **Use Clover to collect Code Coverage for this build** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see Viewing the Clover Code-Coverage for a Build Result). The following fields will be displayed:
  - **Integration Options** — Select one of the following options:
    - **Automatically integrate Clover into this build** — For this option, you have two sub options; ‘Generate a Clover Historical Report’ and ‘Generate a JSON report’. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views. You will also need to insert a Clover license (evaluation licenses are available) into the field provided.

The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.
A issue currently exists preventing this option working on Windows. Please see Enabling the Clover Plugin for more information.

- 'Clover is already integrated into this build and a clover.xml file will be produced' — Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report. You will also need to specify where the Clover XML report is being generated, under 'Clover XML Location'. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:

  target/site/clover/clover.xml

Do not specify an absolute path.

- 'Clover XML Location' — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.
Bamboo Plugins

Did you know that Bamboo has an extensive plugin library? We've implemented a number of popular features as plugins and many of them are free to use. Interested in:

- Integrating Atlassian's JIRA with Bamboo?
- Using Bamboo with Clearcase?
- Specifying post-build commands for your Bamboo builds?

Browse our plugin library or the Atlassian Plugin Exchange for all of these features and more!

Bamboo Builds inside your IDE

Use the Atlassian Connector for Eclipse or the Atlassian Connector for IntelliJ IDEA to work with your Bamboo builds right there in your development environment. Do you use JIRA, Crucible or FishEye too? With the connector you can manage your issues 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.

About

Bamboo is a continuous integration (CI) server. Bamboo assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/failed builds.
- reporting tools for statistical analysis.

Resources

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

- Bamboo Announcements
- Bamboo General Forum
- Bamboo Developers Forum

Other handy links:

- Bamboo Knowledge Base
- Bamboo FAQ
- JIRA Issue Tracker & Feature Requests for Bamboo

Download

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

Previous Versions

Bamboo 2.4.x
Bamboo 2.3.x
Bamboo 2.2.x
Bamboo 2.1.x
Bamboo 2.0.x
Bamboo 1.2.x
Bamboo 1.1.x
Bamboo 1.0.x

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

**Getting Started**

1. **Installing Bamboo**

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

- **For Windows:** (click to expand)
  1. Go to the Atlassian Download Center.
  2. Download the Bamboo Standalone Windows Installer (.EXE) file.
  3. Run the Windows Installer .EXE file, choose an installation directory and a home directory.
  4. Start the Bamboo server by selecting the 'Start in Console' option from the new 'Bamboo' folder that has been created in your Windows 'Start' menu.
  5. To access Bamboo, go to your web browser and type this address: http://localhost:8085/.
  6. Follow the Setup Wizard. This will guide you through the process of setting up your Bamboo server, database and creating an Admin user.

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

If you need assistance, please create a support ticket.

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

- **For Mac:** (click to expand)
  1. Go to the Atlassian Download Center.
  2. Download the Bamboo Standalone (Mac OS X Installer) file.
  3. Run the Mac OS X Installer .DMG file, choose an installation directory and a home directory.
  4. Start the Bamboo server by running `./bamboo.sh console` in the installation root directory.
  5. To access Bamboo, go to your web browser and type this address: http://localhost:8085/.
  6. Follow the Setup Wizard. This will guide you through the process of setting up your Bamboo server, database and creating an Admin user.

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

If you need assistance, please create a support ticket.

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

- **For Unix or Linux:** (click to expand)
  1. Go to the Atlassian Download Center.
  2. Download the Bamboo Standalone (TAR.GZ Archive) file.
  3. Extract the files to a directory of your choice. This directory will be referred to as your 'Bamboo installation directory'.
  4. Set up your Bamboo home directory by opening the file named `bamboo-init.properties` in the `<Bamboo installation directory>/webapp/WEB-INF/classes` directory. In this file, insert the property "bamboo.home", with an absolute path to your desired Bamboo home directory. e.g. `bamboo.home=/test/bamboo-home`
    
    Please ensure that your Bamboo home directory is not located inside the Bamboo installation directory.
  5. Start the Bamboo server by running `./bamboo.sh console` in the installation root directory.
  6. To access Bamboo, go to your web browser and type this address: http://localhost:8085/.
  7. Follow the Setup Wizard. This will guide you through the process of setting up your Bamboo server, database and creating an Admin user.

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

If you need assistance, please create a support ticket.

⚠ Before using Bamboo as a production system, you need to switch from the embedded HSQL database, which is provided for evaluation purposes only. Please see the documentation for details.
2. Configuring the Default Agent

Why would I do this? (click to expand)

Your default local agent will inherit all *local server capabilities* that are defined in your Bamboo system. If you had system environment variables set up for builders and JDKs on your system, then the Bamboo installation process will have set them up as local server capabilities. However, if they were not automatically configured, you can configure them yourself. You may also want to set up additional custom capabilities to match specific types of builds to the agent (e.g. `functional.tests=true`, `fast.builds=true`, `operating.system=linux`, etc.

To configure the default local agent: (click to expand)

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column. The 'Agents' screen will display.
3. Click the 'Default Agent' in the list of agents.
4. Click the 'Add Capability' link to add *agent-specific capabilities* that override shared capabilities (e.g. custom capability: `operating.system=linux`). See the following pages for detailed instructions:
   - Configuring an Agent-specific Builder Capability
   - Configuring an Agent-specific JDK Capability
   - Configuring an Agent-specific Custom Capability

Running your First Build

3. Creating a Plan

To create a new plan in Bamboo: (click to expand)

To run your build on your default agent, you will need to ensure that the capability requirements of the plan can be met by the capabilities of your default agent.

1. Click the 'Create Plan' link in the top navigation bar.
2. Enter the plan details as described on the following pages:
   - Specifying a Plan's Details
   - Specifying a Plan's Source Repository
   - Specifying a Plan's Builder
   - Specifying a Plan's Capability Requirements
   - Specifying a Plan's Build Artifacts
   - Specifying a Plan's Notifications (you need to configure Bamboo's SMTP and/or instant messaging notification capabilities for this feature — see 9. Configuring Notifications' below)
   - Specifying a Plan's Post Actions
   - Specifying a Plan's Permissions

If you are using Atlassian's FishEye, you can link your Bamboo plan to your FishEye project when specifying your plan's source repository. Enter the URL of your FishEye project in the 'Web Repository URL' (e.g. https://svn.myrepo.com/fisheye/browse/myproject) and the Code Changes section of a build result will display links to your files via FishEye.

You can extend Bamboo to support additional repositories, builders and build tools for your plans by installing plugins. Check out our plugin library for more information.

4. Running a Build

To run your build: (click to expand)

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. Locate the relevant plan and click the 'Check Out and Build' icon.
3. You can view the build's activity on the 'Current Activity' tab.
   (if your plan does not build, check whether all of its capability requirements can be met by the remote agent's capabilities)

You can configure your build plan to be triggered by code changes or on a schedule, rather than manually triggering it. See About Build Triggering for more information.
To view the build results: (click to expand)

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. Locate the relevant plan and click the plan name. The plan summary will display, showing the latest build result. You can also click the 'Completed Builds' tab to see a summary list of build results.
   - To view the test results, click the 'Tests' tab.
   - To view the build artifacts, click the 'Artifacts' tab.
   - To view the build logs, click the 'Logs' tab.
   - If you have Atlassian's Clover installed, you can also view the Clover code coverage by clicking the 'Clover' tab. See '8. Using Bamboo with Clover' below for details.

Using Bamboo with your Development Tools

5. Using Bamboo with your IDE

Why would I do this? (click to expand)

Integrating Bamboo with your IDE (via an Atlassian IDE Connector) allows you to work with Bamboo from within your IDE. You won’t have to switch between applications to see what's happening with your builds. You can start them, monitor them and view the results without leaving your IDE.

The Atlassian IDE Connectors are currently available for IntelliJ IDEA and Eclipse.

To use Bamboo with Eclipse: (click to expand)

1. Install the Atlassian Eclipse Connector by following the instructions in this document: Installing the Eclipse Connector
2. Configure Bamboo options in Eclipse by following the instructions in this document: Configuring your Bamboo Options in Eclipse
3. Read the following document for instructions on how to use Bamboo from within Eclipse: Working with Bamboo Builds in Eclipse

To use Bamboo with IntelliJ IDEA: (click to expand)

1. Install the Atlassian IntelliJ Connector by following the instructions in this document: Installing the IntelliJ Connector
2. Configure Bamboo options in IntelliJ IDEA by following the instructions in this document: Configuring your Bamboo Options in IDEA
3. Read the following document for instructions on how to use Bamboo from within IntelliJ: Working with Bamboo Builds in IDEA

6. Using Bamboo with JIRA

Why would I do this? (click to expand)

Integrating Bamboo with Atlassian's JIRA allows you to access Bamboo information from within JIRA and vice versa. Your team will be able to see which issues are being actively coded, which builds have run for an issue, find the build that fixed the issue, download your distribution and much more.

To use Bamboo with JIRA: (click to expand)

1. Configure the JIRA plugin for Bamboo and the Bamboo plugin for JIRA on your Bamboo and JIRA servers by following the instructions in this document: Integrating Bamboo with JIRA
2. Read the following documents for instructions on how to use Bamboo with JIRA:
   - View the Bamboo builds that relate to a particular JIRA issue.
   - View the Bamboo builds that relate to a JIRA project or version.
   - View the JIRA issues for a build result.
   - Add JIRA portlets to display the status of your builds or a graphical summary of each build plan.

7. Using Bamboo with Confluence

Why would I do this? (click to expand)

Integrating Bamboo with Atlassian's Confluence allows you to easily access Bamboo information from within Confluence. You will be able to view the status of builds, build history and charts from within your wiki and share it with team members.
To use Bamboo with Confluence: (click to expand)

1. Configure the Bamboo plugin for Confluence by following the instructions in this document: Bamboo Plugin
2. The Bamboo Plugin document also contains information on how to use the various Confluence macros made available by the Bamboo Plugin.

8. Using Bamboo with Clover

Why would I do this? (click to expand)

If your organisation uses the Atlassian Clover code-coverage tool, Bamboo can record code-coverage details (i.e. the percentage of code covered by tests) for each build result. This can help you to monitor and improve code-coverage over your builds.

To use Bamboo with Clover: (click to expand)

1. Edit the plan that you wish to record code-coverage details for.
2. Click the ‘Builder’ tab.
3. Configure the Clover details for your plan, as described on Enabling the Clover Plugin.

You will now be able to view Clover code-coverage for any subsequent builds of this plan. Please see Viewing the Clover Code-Coverage for a Build Result for further details.

Collaborating with your Team in Bamboo

9. Adding User Groups and Users

Before you create additional users, you may wish to create a non-administrator user group. Here’s how (click to expand):

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Groups’ link in the left navigation column. The ‘Manage Groups’ screen will be displayed.
3. In the ‘Group Name’ field (in the ‘Create Group’ section), type a name for your new group. You can add users to this group when you create them in the next section.
4. Click ‘Save’ to save your new group.

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

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Users’ link in the left navigation column.
3. Enter the ‘Username’, ‘Password’, ‘Full Name’, ‘Email’ and optionally the instant messaging ‘Jabber Address’ for the user.
4. Select a user group for the user.
5. Click the ‘Save’ button.

10. Configuring Notifications

Why would I do this? (click to expand)

Bamboo can automatically notify users when certain events occur in a build. For example, if the build has hung you can ensure that an email is sent to the appropriate people. Setting up the appropriate notifications can help your users to keep in touch with build activity on Bamboo. You can also help foster collaboration between team members by configuring notifications to be sent whenever someone comments on a build result.

To set up SMTP email and/or instant messages for Bamboo: (click to expand)

Bamboo can send notifications based on build plan configuration. For example, you may wish to send an email to all users when a build has hung. To take advantage of this feature, you will first need to configure Bamboo to send SMTP email and/or instant messages.

To configure notifications for a build plan: (click to expand)
1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Notifications' sub-tab.
5. Add a new notification in the 'Add Build Notification' section by doing the following:
   - Specify the 'Event' and the 'Recipient Type' (this may include a child field such as 'User') for the notification.
   - Click the 'Add' button to add the notification.
6. Repeat the last step if you want to add more notifications for the plan.
7. Click the 'Done' button

11. Labelling Builds

Why would I do this? (click to expand)

A label is a convenient way to tag and group build results that are logically related to each other. For example, it might not be practical for your QA team to review every build, and you need to know which builds they have reviewed. By using labels such as "qa_passed" and "qa_failed", Bamboo allows them to simply indicate which builds have passed and failed QA.

Labels can be applied to build results automatically by specifying the label(s) in a build plan (note that only Bamboo administrators can do this) or applied ad hoc to build results by Bamboo users. Labels can also be used to define RSS feeds and to control build expiry.

To add a label to a build result: (click to expand)

1. Go to the build result.
2. Locate the 'Labels' link at the top of the screen (above the 'Summary' tab).
3. Click the 'Add' link. The 'Enter labels to add to this build' textbox will display.
4. Type the relevant label (or multiple labels, separated by commas).
5. Click the 'Done' button.

To view all build results with a particular label: (click to expand)

1. Go to the build result.
2. Locate the 'Labels' link at the top of the screen (above the 'Summary' tab).
3. Click the label that you want to view the build results for. All build results with that label will be displayed.

12. Commenting on Build Results

Why would I do this? (click to expand)

Comments are a useful way to record and share information about builds. Together with notifications, they can help your team to collaborate more effectively. Comments made when code was committed are automatically copied from the repository. Comments can also be manually added to a particular build result.

To add a comment to a build result: (click to expand)

1. From within the 'Build Result' screen, click the 'Comments' tab. A list of existing comments about this build result will be displayed.
2. Type your comment into the 'Add Comment' box, then click the 'Save' button.

Scaling your Build System

13. Setting up Distributed Builds

Why would I do this? (click to expand)

You can set up remote agents to distribute builds across multiple build servers. This is particularly useful if you need to run your builds in different geographic locations, or on different platforms. Installing a Bamboo agent on a new build server is a simple process and once installed, your main Bamboo server will be able to manage them.

To set up a remote agent on a machine: (click to expand)
1. Create a directory on the agent machine (e.g. `bamboo-agent-home`) to serve as the Bamboo agent home for the remote agent.
2. On your Bamboo server, click the 'Administration' link in the top navigation bar.
3. Click the 'Agents' link in the left navigation column. The 'Agents' screen will display.
4. Enable remote agent support by clicking the 'Enable Remote Agent Support' link.
5. Click the 'Install Remote Agent' link. The 'Installing a Remote Agent' screen will display.
6. Click the 'DOWNLOAD Remote Agent JAR' button and save the JAR file to the Bamboo agent home directory on your agent machine.
7. Copy the command under 'Running a Remote Agent' to your clipboard, e.g. `java -jar atlassian-bamboo-agent-installer-2.2.4.jar http://172.20.5.83:8085/agentServer/`, and execute it on your agent machine to start the remote agent.
8. Follow the instructions on the pages linked below to set up agent-specific capabilities for your remote agent:
   - Configuring an Agent-specific Builder Capability
   - Configuring an Agent-specific JDK Capability
   - Configuring an Agent-specific Custom Capability

Once you have set up a remote agent, builds will be assigned to it, just as they are assigned to any other agent.

ℹ️ For information on more remote agent configuration options, see the full Bamboo Remote Agent Installation Guide.

14. Building in the Cloud

(Please note, you need to have an Amazon Web Services (AWS) account to use Elastic Bamboo. If you don't have one, you will need to register for one on the AWS registration page.)

Why would I do this? (click to expand)

The Elastic Bamboo feature allows you to run your builds in the Amazon Elastic Compute Cloud (EC2). This provides you with instant scalability for your builds, allowing you to keep your build queues short while minimising your hardware costs.

To set up Elastic Bamboo: (click to expand)

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Configuration' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Elastic Bamboo Configuration' screen will display.
3. Click the 'Enable' button to enable Elastic Bamboo for your Bamboo installation. The 'Elastic Bamboo Configuration' screen will display.
4. Enter your 'AWS Access Key ID' and 'AWS Secret Access Key' (what are these?)
5. Click the 'Save' button.

✅ You can configure advanced features of Elastic Bamboo on this screen, such as using the Amazon Elastic Block Store (EBS) to improve build times and customise elastic agents. Read Configuring Elastic Bamboo for more information.

To start an elastic agent: (click to expand)

An elastic agent is essentially a remote agent that runs in the Amazon EC2. You can create elastic agents on the fly when you need them to run builds and shut them down when not required. This allows you to dynamically scale your computing resources to meet build demand.

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the 'Start New Elastic Instances' link. The 'Start New Elastic Instances' screen will display.
   - Enter '1' in the 'Number of instances' field.
   - Select the 'Default' elastic image configuration in the 'Elastic Image Configuration Name' dropdown.
   - Your elastic agent will inherit its capabilities from this elastic image. It is possible to set up additional elastic images for use with Elastic Bamboo. Read Managing your Elastic Image Configurations for more information.
4. Click the 'Submit' button. The 'Manage Elastic Instances' page will display, showing the startup of your new instance.

Once you have set up an elastic agent, builds will be assigned to it, just as they are assigned to any other agent.

⚠️ Amazon EC2 charges for the period of time that you have an instance running. Please ensure that you remember to shut down any elastic instances you start, to minimise your costs.

Analysing and Improving your Builds
15. Generating Reports

Why would I do this? (click to expand)

You can generate reports in Bamboo to help you identify problem areas to improve upon for your builds, based on the most common failures, tests that take longest to fix, long-running tests and more.

To generate a report on selected authors: (click to expand)

1. Click the 'Authors' link in the top navigation bar. The 'User and Author Statistics' screen will display.
2. Click the 'Statistics' tab. The 'Report Parameters' screen will display.
3. Select the parameters for your report, as follows:
   - 'Report' — choose from the available reports. Available reports include:
     - 'Build Activity per Author' Report
     - 'Number of Build Failures per Author' Report
     - 'Number of Builds Broken per Author' Report
     - 'Number of Builds Fixed per Author' Report
     - 'Percentage of Successful Builds per Author' Report
   - 'Authors' — choose the author(s) on whom you want to report. You can use the <Ctrl> key to select multiple authors.
   - 'Group By' — choose whether your report's horizontal axis should show days, months or weeks. You can also specify 'Auto' (generally defaults to 'month').
4. Click the 'Submit' button to generate your report.

To report on build statistics per plan: (click to expand)

1. Click the 'Reports' link in the top navigation bar. The 'Report Parameters' screen will display.
2. Select the parameters for your report, as follows:
   - 'Report' — choose from the available reports. Available reports include:
     - 'Build Activity' Report
     - 'Build Duration' Report
     - 'Build Queue Duration' Report
     - 'Clover Code Coverage' Report
     - 'Clover Lines of Code' Report
     - 'Number of Build Failures' Report
     - 'Number of Tests' Report
     - 'Percentage of Successful Builds' Report
     - 'Time to Fix' Report
   - 'Build plans' — choose the plan(s) on which you want to report. You can use the <Ctrl> key to select multiple plans.
   - 'Group By' — choose whether your report's horizontal axis should show days, months or weeks. You can also specify 'Auto' (generally defaults to 'month').
3. 'Date Filter' — date range to report on.
4. Click the 'Submit' button to generate your report.

Important Next Steps

(Please note, you need to be an Administrator to do the tasks in this section.)

16. Connecting to an External Database

Before using Bamboo 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.

17. Backing up Data

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

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

Bamboo Administrator's Guide

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

About

Bamboo is a continuous integration (CI) server. Bamboo assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/tailed builds.
- reporting tools for statistical analysis.

The Bamboo Administrator's Guide provides information about configuring builds and administering your Bamboo system.

For help getting Bamboo up and running for the first time, check out the Bamboo 101 guide.

For more documentation, please visit Bamboo Documentation Home.

Download

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

Search the Administrator's Guide

Getting Help

Support Request | Feature Request | Forums | Knowledge Base

Table of Contents

Configuring Projects and Plans

- About Projects, Plans and Builds
- Creating a Plan
  - Specifying a Plan's Details
  - Specifying a Plan's Source Repository
    - CVS
    - Perforce
      - Using Perforce with Bamboo - Limitations and Workarounds
    - Subversion
  - Specifying a Plan's Builder
    - Ant
    - Custom Command Builder
deenv.com
  - Maven
  - NAnt
  - PHPUnit
  - Script Builder
    - Using Global or Build-specific Variables
      - Defining Global Variables
  - Specifying a Plan's Capability Requirements
  - Specifying a Plan's Build Artifacts
  - Specifying a Plan's Notifications
  - Specifying a Plan's Post Actions
    - Specifying Expiry for a Plan's Build Results
Specifying Labels for a Plan's Build Results
Specifying a Plan's Permissions
Editing a Plan
Disabling or deleting a Plan
Deleting a Build Result
Deleting a Plan's Working Files
Stopping an Active Build
Moving a Plan to a different Project
Renaming a Plan or Project
Modifying Multiple (Bulk) Plans

Configuring Agents and Capabilities

About Agents and Capabilities
Configuring Agents
Creating a Local Agent
Creating a Remote Agent
Disabling Remote Agents Support
Editing an Agent's Details
Disabling or Deleting an Agent
Viewing an Agent
Viewing an Agent's Capabilities
Viewing the Plans that an Agent can build
Determining which Agents can build which Plans
Viewing an Agent's System Properties
Monitoring Agent Status
Configuring Capabilities
Configuring a new Builder
Configuring an Agent-specific Builder Capability
Configuring a Shared Builder Capability
Viewing your Builder Capabilities
Configuring a new JDK
Configuring an Agent-specific JDK Capability
Configuring a Shared JDK Capability
Viewing your JDK Capabilities
Configuring a new Custom Capability
Configuring an Agent-specific Custom Capability
Configuring a Shared Custom Capability
Configuring a new Perforce Capability
Editing a Capability
Renaming a Capability
Deleting a Capability
Viewing the Agents & Plans Related to a Capability

Working with Builds

Triggering a Build
About Build Triggering
Triggering a Build when Code is Updated
Polling the Repository for Code Changes
Triggering a Build on Code Check-in
Triggering a Build on Schedule
Scheduling a Single Daily Build
Specifying a Cron-based Schedule
Triggering a Build Manually
Setting up Build Dependencies
Dependency Blocking Strategies
Reordering your Build Queue
Monitoring Builds
Configuring the Hanging Build Event
Configuring the Build Queue Timeout Event
Disabling Build Monitoring

Working with Elastic Bamboo

About Elastic Bamboo
Elastic Bamboo Costs
Getting Started with Elastic Bamboo
Configuring Elastic Bamboo
Generating your AWS Private Key File and Certificate File
Configuring Elastic Instances to use EBS
Populating your EBS volume
Managing Elastic Bamboo
• Managing your Elastic Images
  • Viewing an Elastic Image
  • Creating a Custom Elastic Image
  • Managing your Elastic Image Configurations
• Managing your Elastic Instances
  • Viewing an Elastic Instance
  • Accessing an Elastic Instance
  • Starting an Elastic Instance
  • Scheduling your Elastic Instances
  • Shutting Down an Elastic Instance
• Managing your Elastic Agents
  • Viewing your Elastic Agents
  • Viewing your Elastic Agent Usage History
  • Configuring Elastic Agent Capabilities
  • Disabling an Elastic Agent
• Running Builds using Elastic Bamboo
• Disabling Elastic Bamboo

Configuring Email and Instant Messaging Notifications

• Adding or Removing Notifications for a Plan
• Modifying Notification Templates
  • Freemarker and Notification Templates
• Configuring Bamboo to send SMTP Email
• Configuring Bamboo to use Instant Messaging (IM)
  • Configuring Bamboo to use Google Talk for Instant Messaging

Managing Users and Permissions

• Managing Users
  • Creating a User
  • Changing a User's Password or Details
  • Deleting or deactivating a User
  • Granting Administration Rights to a User
• Managing Groups
  • Creating a Group
  • Deleting a Group
  • Adding Users to and removing them from Groups
• Managing Permissions for Users and Groups
  • Granting Plan Permissions to Users or Groups
  • Granting Global Permissions to Users or Groups
  • Allowing Anonymous Users to access Bamboo
• Enabling or disabling Contact Details Display
• Enabling or disabling Public Signup
• Working with External User Repositories
  • Integrating Bamboo with Crowd
  • Integrating Bamboo with LDAP
  • Configuring the Caching of your LDAP Repository

Managing Data and Backups

• Locating Important Directories and Files
• Specifying Bamboo's Working Directory
• Viewing your Database Connection Details
• Moving your Bamboo Data to a different Database
• Optimising or Re-indexing Data
• Enabling Expiry of Build Results
• Specifying a Backup Schedule
• Exporting Data for Backup
• Importing Data from Backup

Configuring System Settings

• Viewing Bamboo's System Information
• Updating your Bamboo License Details
• Specifying Bamboo's Title
• Specifying Bamboo's URL
• Configuring Logging in Bamboo
• Enabling GZIP Compression
• Enabling Bamboo's Remote API
• Configuring System Properties
Managing Bamboo Security

- Configuring OAuth Consumers
- Elastic Bamboo Security

Configuring Plugins

- About Bamboo Plugins
- Enabling the ‘Auto-Favourites’ Plugin
- Enabling the Clover Plugin
- Installing a new Plugin

Using Bamboo with Other Applications

- Embedding Bamboo into Other Applications
  - Javascript Widgets
    - All Plans & My Favourite Plans
    - Latest Builds
    - Latest Status of a Plan
    - My Latest Changes
    - Plan Summary Graphs
  - Integrating Bamboo with Other Applications
    - Integrating Bamboo with JIRA

Configuring Projects and Plans

Configuring Projects and Plans

- About Projects, Plans and Builds
- Creating a Plan
  - Specifying a Plan's Build Artifacts
  - Specifying a Plan's Builder
    - Ant
    - Custom Command Builder
devenv.com
  - Maven
  - NAnt
  - PHPUnit
  - Script Builder
- Using Global or Build-specific Variables
  - Defining Global Variables
  - Specifying a Plan's Capability Requirements
  - Specifying a Plan's Details
  - Specifying a Plan's Notifications
  - Specifying a Plan's Permissions
  - Specifying a Plan's Post Actions
    - Specifying Expiry for a Plan's Build Results
    - Specifying Labels for a Plan's Build Results
  - Specifying a Plan's Source Repository
    - CVS
    - Perforce
    - Using Perforce with Bamboo - Limitations and Workarounds
    - Subversion
- Disabling or deleting a Plan
  - Deleting a Build Result
  - Deleting a Plan's Working Files
  - Stopping an Active Build
- Editing a Plan
- Modifying Multiple (Bulk) Plans
- Moving a Plan to a different Project
- Renaming a Plan or Project
About Projects, Plans and Builds

- A Bamboo plan (or build plan) is the "recipe" for a build.

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project.

- A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans.

Each project has a Name (e.g. "CRM System") and a Key (e.g. "CRM"). The Project Key is prefixed to the relevant Plan Keys, e.g. the "CRM" project could have plans "CRM-TRUNK" and "CRM-BRANCH".

Note that creating a new project only requires defining the Project Name and Project Key, which is (optionally) done as part of the process of creating a new plan.

- A build is one execution of a plan.

Every build has a Build Number, which is appended to the relevant Plan Key to form the Build Key. For example, if a plan with the key "CRM-BRANCH" is executed for the seventeenth time, the build key will be "CRM-BRANCH-17".

Each plan's build results are stored in a subdirectory under your 'Build Directory' (see Locating Important Directories and Files).

Creating a Plan

Only people with the 'Create Plan' global permission can create a new build plan.

There are two ways of creating a new plan:

To create a new plan,

1. Click the 'Create Plan' link in the top navigation bar.
2. The 'Create a new plan' wizard will appear (see screenshot below). Click the nine sub-tabs to edit the following:
   - 1. Plan Details
   - 2. Source Repository
   - 3. Builder Configuration
   - 4. Capability Requirements
   - 5. Build Artifacts (optional; can be completed now or later)
   - 6. Build Notifications (optional; can be completed now or later)
   - 7. Post Actions (optional; can be completed now or later)
   - 8. Permissions (optional; can be completed now or later)
3. When you return to the Dashboard, your new plan (and new project, if applicable) will be displayed in the 'All Projects' list.

To copy an existing plan,

1. Click the 'Create Plan' link in the top navigation bar.
2. On the 'Plan Details' screen, select the check-box 'Clone an existing build plan?'
3. A list called 'Plan to clone' will be displayed, containing all existing plans for which you have the 'Clone' and/or 'Admin' plan permission. Select the plan you wish to copy.
4. Enter the required information in the screens listed above. On screens 2-8, appropriate information will be copied from the plan you selected; but you will need to complete all fields on screen 1.
Specifying a Plan’s Details

When creating a new plan, the first step is to specify the plan’s details and the project to which it will belong.

A project is a collection of plans.

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans.

Each project has a Name (e.g. “CRM System”) and a Key (e.g. “CRM”). The Project Key is prefixed to the relevant Plan Keys, e.g. the “CRM” project could have plans “CRM-TRUNK” and “CRM-BRANCH”.

To specify a plan’s details,

1. **Project** — When you create a new plan, you can either add it to an existing project or create a new project. Either:
   - Select the appropriate project from the drop-down list;
   - or:
     - Select ‘New Project’ and complete the following two fields:
       a. **Project Name** — Type a descriptive name (e.g. ‘Issue Tracking Application’) that will identify your project on the Dashboard and in reports.
       b. **Project Key** --- Type a logical contraction of the Project Name (e.g. ‘ITA’). The Project Key will be included in the plan’s Build Results keys (e.g. ‘ITA-MAIN-179’), so you may want to make it no longer than 3 or 4 characters. The Project Key must be unique within your Bamboo system.

2. **Build Plan Name** --- Type a name that will identify the plan within its project (e.g. ‘Main Build’, ‘Branch’, ‘Unit Tests’, ‘Acceptance Tests’). Note that the Build Plan Name, which is displayed throughout Bamboo, is always accompanied by its Project Name.

3. **Build Plan Key** — Type a logical contraction of the Build Plan Name. The Build Plan Key (e.g. ‘MAIN’) will be included in the plan’s Build Results keys (e.g. ‘ITA-MAIN-179’), so you may want to make it no longer than 3 or 4 characters.
   - Note that the Build Plan Key only has to be unique within the project, that is, you could have a ‘MAIN’ plan in lots of different projects.

4. Click the ‘Next’ button to go to Specifying a Plan’s Source Repository.

*Screenshot: ‘Plan Details’*
Specifying a Plan’s Source Repository

For each plan, you need to specify what type of source-code repository the plan will use, where the repository is located, and what type of build strategy the plan will use. Please see:

- CVS
- Perforce
- Subversion

CVS, Subversion and Perforce are supported out-of-the-box. If you need to use a different type of repository, a number of third-party Source Repository plugin modules are available (e.g. ClearCase plugin). You can also write a Source Repository Module plugin to enable Bamboo to connect to your repository.

Screenshot: ‘Source Repository --- CVS’
CVS

For each plan, you need to specify what type of source-code repository the plan will use, where the repository is located, and what type of build strategy the plan will use.

To specify a CVS repository,
If you are creating a new plan, start at step 5.

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:  📊
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. The 'Source Repository' sub-tab will be displayed (see screenshot below). Complete the fields as follows:

**CVS configuration**

- 'Repository' — select 'CVS'.
- 'CVS Root' — Type the full path to your CVS repository root (e.g. `:pserver:me@cvs.atlassian.com:/cvsroot/atlassian`). Bamboo supports pserver, ext (ssh) and local repository access methods. Note that you can use global variables in this field (see Using Global or Build-specific Variables).
- 'Authentication Type' — Select either 'Password' or 'SSH'.
  - If you select 'Password', the following fields will appear:
    - 'Password' — (Optional) Type the password for your CVS repository.
    - 'Change Password' — (Will only appear after you have saved the plan) Select this check-box if you want to change the password that is used to access the CVS repository.
  - If you select 'SSH', the following fields will appear:
    - 'Private Key' — Type the absolute path of your SSH private key.
    - 'Passphrase' — Type the passphrase for your SSH private key.
    - 'Change Passphrase' — (Will only appear after you have saved the plan) Select this check-box if you want to change the password for your SSH private key.
- 'Quiet Period' — This setting is used to avoid starting a build while someone is in mid-checkin. Bamboo will only initiate a build for this plan when no more changes are detected within the Quiet Period following the last known change. Type the number of seconds Bamboo should wait. Please note that this parameter is mandatory for CVS, as CVS allows partial checkouts.
- 'Module' — Type the name of the CVS module that contains the source-code.
  - Currently Bamboo has limited support for CVS ampersand modules. To use an ampersand module, you will need to define a regular module with the same name as the ampersand module (since Bamboo expects there to be a directory with the specified checkout module name). For example:
    1. Create a module (e.g. allbuilds).
    2. Define an ampersand module with the same name. (The ampersand module can be empty.)
    3. In the 'Module' field, enter the following: `allbuilds allbuilds &project2 &project2 &project3`
- 'Version of Module' — Select either 'HEAD' or 'Branch/Tag'. If you select 'Branch/Tag', the following field will appear:
  - 'Branch name' — Type the relevant branch name or tag. Note that you can use global variables in this field (see Using Global or Build-specific Variables).

**Common Repository Configuration**

- 'Force Clean Build' (Optional) — You can force Bamboo to remove the source directory and check it out again prior to each build by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
- 'Include/Exclude Files' — (Optional) You can specify a particular inclusion or exclusion pattern for file changes to be detected.
- 'File Pattern' — (Optional) The regular expression for file changes which you wish to include/exclude.
- 'Web Repository URL' — (Optional) You can specify the URL of the plan's browsable repository. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.

**Linking to your FishEye project**

If you are using Atlassian's FishEye, you can link your Bamboo plan to your FishEye project. Enter the URL of your FishEye project in the 'Web Repository URL' (e.g. `https://svn.myrepo.com/fisheye/browse/myproject`) and the Code Changes section of a build result will display links to your files via FishEye.

- 'Web Repository Module' — (Optional) The plan's repository name, if the above Web Repository URL points to multiple repositories.
- 'Build Strategy' — The default value, 'Poll the repository for changes', is a convenient option that requires no additional configuration. A number of other options are available; for details, please see Triggering a Build. You can change the Build Strategy over time as required. The rest of the fields on this tab will vary depending on which Build Strategy you select.
- Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Builder.
For each plan, you need to specify what type of source-code repository the plan will use, where the repository is located, and what type of build strategy the plan will use.

If you wish to build plans on your server and remote agents using a Perforce repository, you need to specify the location of the Perforce P4 client application for your server and each remote agent using Perforce. These locations are set by specifying a mandatory local server Perforce capability for your server and agent-specific remote Perforce capabilities for each of your remote agents using Perforce. You will not be able to create plans that use a Perforce repository without specifying the shared local Perforce capability first.

Read more about configuring a Perforce capability.

Keeping your Perforce configuration up to date
Please note, that if you are using Perforce as your repository, you will need to ensure that you keep the Perforce configuration in Bamboo in sync with any changes to your Perforce repository (e.g. change to client, depot, user credentials). Otherwise, these changes can cause unexpected behaviour in Bamboo when it tries to access the repository. See the notes in the configuration instructions below for further details.

To specify a Perforce repository,

If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Source Repository’ sub-tab.
5. The ‘Source Repository’ sub-tab will be displayed. Complete the fields as follows:
- **Repository** — select ‘Perforce’.
- **Port** — Type either the port to which the Perforce client will connect, or the Perforce server itself. This is the Perforce P4PORT environment variable that tells Bamboo which p4d (Perforce server) to use.
- **Client** — The name of the Perforce Client Workspace which Bamboo will use. The Client Workspace determines which portions of the depot are visible in your Workspace Tree.

  Do not create two plans that use the same client (e.g. one client set to manage, the other client set to not manage). This setup will create major issues in your builds.

- **Depot** — The client view of the depot that contains the source-code files for this plan. This is typically in the form //<clientname>/<workspace_mapping>/... For details please see the Perforce User’s Guide.
  - Bamboo sets the client root to its working directory, which means that code will be checked out to the working directory/<workspace_mapping> location. Please take note of this, when specifying the 'Artifact Copy Pattern' for your Build Artifacts.
- **Username** — (Optional) The Perforce username that Bamboo will use when it accesses the server ('Port'). Leave this field blank if you want Bamboo to use the default Perforce user (i.e. the OS username).
- **Password** — (Optional) Type the password required by the Perforce username (if applicable).
- **Change Password** — (This field will only appear after you have saved the plan) Select this check-box if you want to change the password that is used to access the Perforce repository.
- **Let Bamboo manage your workspace** — This field indicates whether you want Bamboo to manage your workspace or not.

Please be aware of the following implications of letting/preventing Bamboo from managing your workspace:

**If you let Bamboo manage your workspace,**
- We recommend this configuration if your plans will be running on many different machines or different operating systems, as Bamboo sets the client root for you.
- Bamboo will make configuration changes to the Client Workspace to manage builds (e.g. Bamboo will modify the host and root). You need to ensure that you enter a Client Workspace host root in the 'Client' field that will be used solely for Bamboo.
- Under this configuration, you should configure one client per plan to avoid conflicts when updating the client root.

**If you do not let Bamboo manage your workspace,**
- We recommend this configuration if you wish to reuse your client for several plans, as Bamboo will retrieve the client root directory from Perforce and use it to run builds.

  **Setting the client root in Perforce:** We strongly recommend that you choose a directory that is dedicated for Bamboo’s use only, when you are specifying the client root in your Perforce repository. This directory may get cleaned (i.e. files and sub-directories deleted) if you choose to force clean builds.
- Under this configuration, you need to ensure that the client root directory exists on all machines that the plan will be built on.
- Please note that alternate roots does not currently work in Bamboo. See issue BAM-2377 for further details.

- **Enable Quiet Period:** — Select this setting to set Quiet Period parameters for the build plan.
- **Quiet Period:** — This field will only display if 'Enable Quiet Period:' has been selected. This setting is used to avoid starting a build while someone is in mid-checkin. Bamboo will only initiate a build for this plan when no more changes are detected within the Quiet Period following the last known change. Type the number of seconds Bamboo should wait.
- **Maximum Retries:** — This field will only display if 'Enable Quiet Period:' has been selected. You can specify how many times Bamboo should check for new changes using the Quiet Period parameter before initiating a build. For example, if you have set the 'Quiet Period' to 10' seconds then Bamboo will check if a checkout has occurred in the last 10 seconds. If you have then specified 'Maximum Retries:' as '5', then Bamboo will perform this check five times before initiating the build, regardless of any activity during the Quiet Period of the last check.
## Common Repository Configuration

- **'Force Clean Build'** *(Optional)* — You can force Bamboo to remove the source directory and check it out again prior to each build by selecting this option. Please note that this will greatly increase the time it takes to complete a build.

- **'Include/Exclude Files'** *(Optional)* — You can specify a particular inclusion or exclusion pattern for file changes to be detected.

- **'File Pattern'** *(Optional)* — The regular expression for file changes which you wish to include/exclude.

- **'Web Repository URL'** *(Optional)* — You can specify the URL of the plan's browsable repository. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.

### Linking to your FishEye project

If you are using Atlassian's FishEye, you can link your Bamboo plan to your FishEye project. Enter the URL of your FishEye project in the 'Web Repository URL' *(e.g.)*

```plaintext
https://svn.myrepo.com/fisheye/browse/myproject
```

and the Code Changes' section of a build result will display links to your files via FishEye.

- **'Web Repository Module'** *(Optional)* — The plan's repository name, if the above Web Repository URL points to multiple repositories.

- **'Build Strategy'** — The default value, 'Poll the repository for changes', is a convenient option that requires no additional configuration. A number of other options are available; for details, please see Triggering a Build. You can change the Build Strategy over time as required. The rest of the fields on this tab will vary depending on which Build Strategy you select.

- Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Builder.

---

*Screenshot: 'Source Repository --- Perforce'*
Using Perforce with Bamboo - Limitations and Workarounds

There are some limitations to using Bamboo with Perforce. Please read the following information carefully before setting up a build plan to use Perforce.

On this page:

- 1. Running Builds on Multiple Remote Agents/Machines
  - Limitation
  - Background
  - Workarounds
- 2. Using Perforce Overlay and Exclusionary Mappings in Bamboo
  - Limitation
  - Background
  - Workarounds

1. Running Builds on Multiple Remote Agents/Machines

Limitation

You will not be able to run builds on multiple remote agents and/or multiple remote machines using a Perforce repository, without using one of the workarounds described below. If you try to do so, you will run into problems with change detection that could cause your agents to build incorrect code. This problem does not affect the running of builds on multiple local agents.

Background
Perforce is a client/server SCM (software configuration management) system that manages your changes/files by storing the change information on its server. However, storing change information on the Perforce server can cause problems when you have clients on multiple agents/machines. If you have downloaded a particular change with a Perforce client, the change will be marked as downloaded by the Perforce server. If you use the same Perforce client on another machine, the Perforce server will incorrectly assume that you have already downloaded that particular change and will not download it. Hence, your agents may not pick up changes correctly and could build incorrect code.

Workarounds

There are a few workarounds available for this issue, if you are using Perforce with Bamboo:

1. **Restrict your plan to use a single machine** — you can use one or more remote agents to build a plan, if they are running on the same machine and you set the client root yourself (i.e. do not let Bamboo manage your workspace) so that your agents will build to the same directory.
2. **Make Bamboo force a clean build every time it builds** — this will ensure that your agents are always building the correct code. However, it can be an inefficient setup for big projects.
3. **Use alternate roots for different machines** — specifying alternate roots for different machines will allow you to work around the change detection issue, as long as the roots on each machine are unique. Please note however, you will be restricted to three machines (with three different roots) due to Perforce limitations.

Please note, we are aware of these problems and are working to address them — see the following JIRA issues for further information, BAM-2843 and BAM-2774.

2. Using Perforce Overlay and Exclusionary Mappings in Bamboo

Limitation

You will not be able to control how Bamboo detects changes using exclusionary mappings or overlay mappings. Please note, this issue does not affect you if you only trigger your builds on a schedule or manually, as Bamboo agents still build the correct code when triggered.

Background

Bamboo currently uses the depot view, not the client view, when detecting changes. Hence, any exclusionary and overlay mappings will not be available during change detection.

For example, if a p4 client uses an overlay mapping like this one;

```plaintext
//depot/Prj/... /clientName/depot/Prj/...
+//depot/Dep/... /clientName/depot/Prj/Dep/...
```

and the 'Depot' specified in a plan's repository configuration is:

```plaintext
/clientName/depot/Prj/...
```

then Bamboo will lookup the corresponding depot view and detect changes by running the following command:

```plaintext
p4 changes //depot/Prj/...
```

Consequently, no changes to files in //clientName/depot/Prj/Dep/... will be picked up by change detection, despite the overlay mapping.

Hence, if you set up your build to trigger when code is updated it will not trigger correctly.

Workarounds

A partial workaround is available in Bamboo, if you wish to use exclusionary mappings for your client workspace. Specify your build plan to exclude files that match a specified pattern by choosing 'Exclude all changes that match the following pattern' from the 'Include / Exclude Files' dropdown (under the 'Common repository configuration' section). See this document for further details. Please note, this will only exclude one pattern whereas multiple exclusions can be specified in an exclusionary mapping.

Unfortunately, there is no workaround for overlay mappings in Bamboo.

Please note, we are aware of these problems and are working to address them — see the following JIRA issue for further information, BAM-3323.

Subversion
For each plan, you need to specify what type of source-code repository the plan will use, where the repository is located, and what type of build strategy the plan will use.

To specify a Subversion repository,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. The 'Source Repository' sub-tab will be displayed. Complete the fields as follows:

**Subversion Configuration**

- 'Repository' — select 'Subversion'.
- 'Repository URL' — The location of your Subversion repository (e.g. http://svn.collab.net/repos/svn/trunk). Note that you can use global variables in this field (see Using Global or Build-specific Variables).
- 'Username' — (Optional) The Subversion username (if any) required to access the repository.
- 'Authentication Type' — Select either 'Password' or 'SSH'.
  - If you select 'Password', the following fields will appear:
    - 'Password' — (Optional) Type the password required by the Subversion username (if applicable).
    - 'Change Password' — (Will only appear after you have saved the plan) Select this check-box if you want to change the password that is used to access the Subversion repository.
  - If you select 'SSH', the following fields will appear:
    - 'Private Key' — Type the absolute path of your SSH private key.
    - 'Passphrase' — Type the passphrase for your SSH private key.
    - 'Change Passphrase' — (Will only appear after you have saved the plan) Select this check-box if you want to change the password for your SSH private key.'

**Advanced Options**

(The following fields will only display if you select the 'Enable Advanced Options' check-box.)

- 'Enable Quiet Period:' — Select this setting to set Quiet Period parameters for the build plan.
- 'Quiet Period:' — This field will only display if 'Enable Quiet Period:' has been selected. This setting is used to avoid starting a build while someone is in mid-checkin. Bamboo will only initiate a build for this plan when no more changes are detected within the Quiet Period following the last known change. Type the number of seconds Bamboo should wait.
- 'Maximum Retries:' — This field will only display if 'Enable Quiet Period:' has been selected. You can specify how many times Bamboo should check for new changes using the Quiet Period parameter before initiating a build. For example, if you have set the 'Quiet Period' to '10' seconds then Bamboo will check if a checkout has occurred in the last 10 seconds. If you have then specified 'Maximum Retries:' as '5', then Bamboo will perform this check five times before initiating the build, regardless of any activity during the Quiet Period of the last check.
- 'Detect Changes in Externals:' — (Optional) Select this check-box if your Subversion repository uses svn:externals to link to other repositories (note that your externals must be in the root of the checkout directory, not in a subdirectory). Please note that you only need to select this check box if you require Bamboo to detect changes in the externals. If your externals references a particular (static) revision, you do not need to check this box.

**Common Repository Configuration**

- 'Force Clean Build' (Optional) — You can force Bamboo to remove the source directory and check it out again prior to each build by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
- 'Include/Exclude Files' — (Optional) You can specify a particular inclusion or exclusion pattern for file changes to be detected.
- 'File Pattern' — (Optional) The regular expression for file changes which you wish to include/exclude.
- 'Web Repository URL' --- (Optional) You can specify the URL of the plan's browsable repository. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
Linking to your FishEye project

If you are using Atlassian's FishEye, you can link your Bamboo plan to your FishEye project. Enter the URL of your FishEye project in the 'Web Repository URL' (e.g. https://svn.myrepo.com/fisheye/browse/myproject) and the Code Changes section of a build result will display links to your files via FishEye.

- 'Web Repository Module' — (Optional) The plan's repository name, if the above Web Repository URL points to multiple repositories.
- 'Build Strategy' — The default value, 'Poll the repository for changes', is a convenient option that requires no additional configuration. A number of other options are available; for details, please see Triggering a Build. You can change the Build Strategy over time as required. The rest of the fields on this tab will vary depending on which Build Strategy you select.
- Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Builder.

If you use pre-1.5 Subversion client to access code checked out by Bamboo, you may encounter problems with your builds. This is due to the SVNKit upgrade in Bamboo 2.1.4. Please read this knowledge base article for further details.

Screenshot: 'Source Repository --- SVN'

Specifying a Plan's Builder
When you configure a plan, you need to specify which builder should be used for the plan's builds. If you specify an Ant or Maven builder, you will also need to choose a JDK.

At least one builder and one JDK were configured automatically when you installed Bamboo. You can add more builders of different types as described in Configuring a new Builder, and you can add more JDKs as described in Configuring a new JDK. They will then appear in the 'Builder' drop-down list and the 'Build JDK' drop-down list as described in the pages linked below:

- Ant
- Custom Command Builder
- devenv.com
- Maven
- NAnt
- PHPUnit
- Script Builder

Screenshot: ‘Specifying a Plan's Builder’
When you configure a plan, you need to specify which **builder** should be used for the plan's builds. If you specify an Ant builder, you will also need to choose a **JDK**.

To configure an Ant builder for a plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:

Ant

If you are creating a new plan, start at step 5.
4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.
5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).
6. In the 'Builder' dropdown, select the Ant builder that you wish to configure for this plan (e.g. "Ant").

The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

7. The screen will refresh to display the builder settings specific to Ant:
   - 'Build File' — Type the relevant filename (e.g. build.xml). You can include variables (see Using Global or Build-specific Variables).
   - 'Target' — Specify the Ant target that you want Bamboo to execute each time the source code changes. For example: test (this will run the Ant target 'test'). You can also use '-D' to define one or more JVM parameters, e.g.: -Djava.awt.headless=true (this will pass the parameter 'java.awt.headless' with a value of 'true'). You can also include variables (see Using Global or Build-specific Variables).
   - 'Build JDK' — Choose a JDK from the list. The JDK that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

---

**Specifying Generic JDK Requirements**

If you want to indicate that an agent is capable of running builds for a set of related JDKs (e.g. all point versions of JDK 1.5), you set up generic JDK capabilities to encompass these JDKs. For example, you can set up the following JDK capabilities for your Bamboo agent(s):

- **JDK** (where Label='JDK', Java Home='/usr/java/jdk1.5.0_07')) — this JDK capability indicates that an agent(s) is capable of running builds with any JDK requirement.
- **JDK 1.5** (where Label='JDK 1.5', Java Home='/usr/java/jdk1.5.0_07')) — this JDK capability indicates that an agent(s) is capable of running builds with a JDK 1.5 requirement or any point version of JDK 1.5, e.g. 1.5.0_07, 1.5.0_08, etc.
- **JDK 1.5.0_07** (where Label='JDK 1.5.0_07', Java Home='/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is only capable of running builds with a JDK 1.5.0_07 requirement.

(If you have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.)

You can then specify these generic JDK capabilities as requirements for your plan.
8. Update the following general build parameters:

- **'System Environment Variables' (Optional)** — Specify any additional operating system environment variables you want to pass to your build; Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).

- **'Working Sub Directory' (Optional)** — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.

- **'The build will produce test results'** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
  - 'Look in the standard test results directory' — Select this option if Bamboo should look in the Builder's standard test results directory.
  - 'Specify custom results directories' — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
    - 'Specify custom results directories' — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as '
  - '/test-reports/*.xml'. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.

- **'NCover output will be produced'** — Do not check this option. NCover is not relevant to Ant builders.

- **'Override default build hanging detection'** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
  - 'Build Time Multiplier' — This setting is used to calculate the 'Expected Build Time' for the build, i.e. 'Expected Build Time' = 'Build Time Multiplier' times 'Average Build Time' (note, the 'Average Build Time' is calculated by Bamboo by using an average of previous build times).
  - 'Log Quiet Time' — This is the amount of time since Bamboo last recorded an entry in the build log for a build. The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.

- **'Build Queue Timeout'** — This is the amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the Build Queue Timeout Event).

- **'Use Clover to collect Code Coverage for this build.'** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see Viewing Bamboo's System Information for a list). Note that existing environment variables are automatically available to the builder, thus you don't need to specify them in the 'System Environment Variables' field.
  - **Automatically integrate Clover into this build** — For this option, you have two sub options; 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views.
    - You will also need to insert a Clover license (evaluation licenses are available) into the field provided.
  - 'Clover XML Location' — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory, for example:
    - target/site/clover/clover.xml
  - Do not specify an absolute path.

- **'Clover is already integrated into this build and a clover.xml file will be produced'** — Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report. You will also need to specify where the Clover XML report is being generated, under 'Clover XML Location'. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:

  - target/site/clover/clover.xml

9. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.
Custom Command Builder

When you configure a plan, you need to specify which builder should be used for the plan's builds.

To a custom command builder for a plan,

If you are creating a new plan, start at step 5.

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 📘
4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.
5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).
6. In the 'Builder' field, select the custom command builder that you wish to configure for this plan (e.g. "CustomCommandBuilder1").

The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

7. The screen will refresh to display the builder settings specific to custom command builders:
   - 'Argument' — Specify the relevant argument to pass to the command. Note that arguments which contain spaces must be quoted. You can include variables (see Using Global or Build-specific Variables).
8. Update the following general build parameters:
   - **System Environment Variables** *(Optional)* — Specify any additional operating system environment variables you want to pass to your build; Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).
   - **Working Sub Directory** *(Optional)* — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.
   - **The build will produce test results** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
     - 'Look in the standard test results directory' — Select this option if Bamboo should look in the Builder's standard test results directory.
     - 'Specify custom results directories' — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
       - 'Specify custom results directories' — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as * /test-reports/.*.xml. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser.bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.
   - **NCover output will be produced** — Do not check this option. NCover is not relevant to Ant builders.
   - **Override default build hanging detection** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
     - **Build Time Multiplier** — This setting is used to calculate the 'Expected Build Time' for the build, i.e. 'Expected Build Time' = 'Build Time Multiplier' times 'Average Build Time' (note, the 'Average Build Time' is calculated by Bamboo by using an average of previous build times)
     - **Log Quiet Time** — This is the amount of time since Bamboo last recorded an entry in the build log for a build.
       - The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.
     - **Build Queue Timeout** — This is the amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the Build Queue Timeout Event).
   - **Use Clover to collect Code Coverage for this build** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see Viewing Clover Code-Coverage for a Build Result). The following fields will be displayed:
     - **Integration Options** — Select one of the following options:
       - **Automatically integrate Clover into this build** — For this option, you have two sub options; 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views.
       - **Clover XML Location** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory, for example:
         - `target/site/clover/clover.xml`
       - Do not specify an absolute path.
     - **Clover XML Location** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory, for example:
       - `target/site/clover/clover.xml`
   - **Build Queue Timeout Event** — This setting is used to calculate the 'Expected Build Time' for the build, i.e.
     - * Operating system environment variables
     - * System Environment Variables
     - * using Bamboo to throw the build hung event.
     - * configuring the build queue timeout setting (see Configuring the Build Queue Timeout Event).
   - **Use Clover to collect Code Coverage for this build** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see Viewing Clover Code-Coverage for a Build Result). The following fields will be displayed:
     - **Integration Options** — Select one of the following options:
       - **Automatically integrate Clover into this build** — For this option, you have two sub options; 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views.
       - **Clover XML Location** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory, for example:
         - `target/site/clover/clover.xml`
       - Do not specify an absolute path.
   - **Clover XML Location** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory, for example:
     - `target/site/clover/clover.xml`
   - **Do not specify an absolute path.**

9. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.
When you configure a plan, you need to specify which builder should be used for the plan's builds.

To configure a devenv.com builder for a plan,

1. If you are creating a new plan, start at step 5.
   1. Click 'Home' to go to the Dashboard.
   2. Click the 'All Plans' tab.
   3. Locate the plan in the list and click this icon: 🛠️
   4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.
   5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).
   6. In the 'Builder' field, select the devenv.com builder that you wish to configure for this plan (e.g. "devenvBuilder1").
   7. The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.
   8. The screen will refresh to display the builder settings specific to devenv.com:
      - 'Solution' — Type the name of the Visual Studio solution file that you want Bamboo to execute each time the source code changes. For example: `run`. You can also include variables (see Using Global or Build-specific Variables).
      - 'Options' — Specify any devenv command line options that you want to include (e.g. `@/build Debug`). You can also include variables (see Using Global or Build-specific Variables).
8. Update the following general build parameters:

- **‘System Environment Variables’ (Optional)** — Specify any additional* operating system environment variables you want to pass to your build: Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).
  * i.e. additional to the existing environment variables (see Viewing Bamboo's System Information for a list). Note that existing environment variables are automatically available to the builder, thus you don't need to specify them in the 'System Environment Variables' field.

- **‘Working Sub Directory’ (Optional)** — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.

- **‘The build will produce test results’** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
  - ‘Look in the standard test results directory’ — Select this option if Bamboo should look in the Builder's standard test results directory.
  - ‘Specify custom results directories’ — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
    - ‘Specify custom results directories’ — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as "* /test-reports/*.xml". Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN), i.e. please do not specify an absolute path.

- **‘NCover output will be produced’** — Do not check this option. NCover is not relevant to Ant builders.

- **‘Override default build hanging detection’** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
  - **‘Build Time Multiplier’** — This setting is used to calculate the 'Expected Build Time' for the build. A ‘Build Time Multiplier’ is calculated by Bamboo by using an average of previous build times.
  - **‘Log Quiet Time’** — This is the amount of time since Bamboo last recorded an entry in the build log for a build.

The ‘Expected Build Time’ and ‘Log Quiet Time’ must both be exceeded for Bamboo to throw the build hung event.

- **‘Build Queue Timeout’** — This is the amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the Build Queue Timeout Event).

- **‘Use Clover to collect Code Coverage for this build.’** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see Viewing the Clover Code-Coverage for a Build Result). The following fields will be displayed:
  - **‘Integration Options’** — Select one of the following options:
    - **‘Automatically integrate Clover into this build’** — For this option, you have two sub options: 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views.
      You will also need to insert a Clover license (evaluation licenses are available) into the field provided.
      ! A issue currently exists preventing this option working on Windows. Please see Enabling the Clover Plugin for more information.
    - **‘Clover is already integrated into this build and a clover.xml file will be produced’** — Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report. You will also need to specify where the Clover XML report is being generated, under 'Clover XML Location'. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:
      ```
target/site/clover/clover.xml
```  
      Do not specify an absolute path.
    - **‘Clover XML Location’** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN), i.e. please do not specify an absolute path.

9. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.
Maven

When you configure a plan, you need to specify which builder should be used for the plan's builds. If you specify a Maven builder, you will also need to choose a JDK.

To configure a Maven builder for a plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.
5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).
6. In the 'Builder' field, select the Maven builder that you wish to configure for this plan (e.g. "Maven 2").

The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

7. The screen will refresh to display the builder settings specific to Maven:
   - 'Goal' — Specify the Maven goal that you want Bamboo to execute each time the source code changes. For example: clean test (this will run the Maven goal 'clean' followed by the Maven goal 'test'). You can also use '-D' to define one or more JVM parameters, e.g.: -Djava.awt.headless=true (this will pass the parameter 'java.awt.headless' with a value of 'true'). You can also include variables (see Using Global or Build-specific Variables).
   - 'Build JDK' — Choose a JDK from the list. The JDK that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

Specifying Generic JDK Requirements

If you want to indicate that an agent is capable of running builds for a set of related JDKs (e.g. all point versions of JDK 1.5), you set up generic JDK capabilities to encompass these JDKs. For example, you can set up the following JDK capabilities for your Bamboo agent(s):

- **JDK** (where Label='JDK', Java Home='/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with **any** JDK 1.5 requirement.
- **JDK 1.5** (where Label='JDK 1.5', Java Home='/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with a JDK 1.5 requirement or any point version of JDK 1.5, e.g. 1.5.0_07, 1.5.0_08, etc.
- **JDK 1.5.0_07** (where Label='JDK 1.5.0_07', Java Home='/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is only capable of running builds with a JDK 1.5.0_07 requirement.

(If you have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.)

You can then specify these generic JDK capabilities as requirements for your plan.
8. Update the following general build parameters:

- **'System Environment Variables' (Optional)** — Specify any additional operating system environment variables you want to pass to your build: Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see [Using Global or Build-specific Variables](#)).

  - *i.e. additional to the existing environment variables (see Viewing Bamboo's System Information for a list). Note that existing environment variables are automatically available to the builder, thus you don't need to specify them in the 'System Environment Variables' field.*

- **'Working Sub Directory' (Optional)** — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in [Locating Important Directories and Files](#)). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.

- **'The build will produce test results'** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
  - 'Look in the standard test results directory' — Select this option if Bamboo should look in the Builder's standard test results directory.
  - 'Specify custom results directories' — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
    - 'Specify custom results directories' — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as *./test-reports/*.xml. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.

- **'NCover output will be produced'** — Do not check this option. NCover is not relevant to Ant builders.

- **'Override default build hanging detection'** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
  - **Build Time Multiplier** — This setting is used to calculate the 'Expected Build Time' for the build, i.e. 'Expected Build Time' = 'Build Time Multiplier' times 'Average Build Time' (note, the 'Average Build Time' is calculated by Bamboo by using an average of previous build times)
  - **Log Quiet Time** — This is the amount of time since Bamboo last recorded an entry in the build log for a build.

  ! The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.

  - **Build Queue Timeout** — This is the amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see [Configuring the Build Queue Timeout Event](#)).

- **'Use Clover to collect Code Coverage for this build.'** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view from within Bamboo (see [Viewing Bamboo's System Information](#)). For this, specify the name of the directory (including path) where Bamboo will look for the Clover XML report is being generated, under 'Clover XML Location'. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:

  ```shell
target/site/clover/clover.xml
  ```

  ! Do not specify an absolute path.

- **'Clover XML Location'** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.

  ! Do not specify an absolute path.

9. Click the **'Save'** button if you are editing an existing plan; or if you are creating a new plan, click the **'Next'** button and go to [Specifying a Plan's Capability Requirements](#).
When you configure a plan, you need to specify which builder should be used for the plan's builds.

To specify an NAnt builder for a plan,

ℹ️ If you are creating a new plan, start at step 5.
1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:
4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.
5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).
6. In the 'Builder' field, select the NAnt builder that you wish to configure for this plan (e.g. "NAnt").

The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.
7. The screen will refresh to display the builder settings specific to NAnt:
   - 'Build File' — Type the relevant filename (e.g. default.build). You can include variables (see Using Global or Build-specific Variables).
   - 'Target' — Specify the NAnt target that you want Bamboo to execute each time the source code changes. For example:RegexDemo/RegexDemo.sln. You can also include variables (see Using Global or Build-specific Variables).
   - 'Options' — Specify the NAnt command line options that you want to include (e.g. You can also include variables (see Using Global or Build-specific Variables).
   - 'System Environment Variables' (Optional) — Specify any additional operating system environment variables you want to pass to your build: Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xmx200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).
   - 'Working Sub Directory' (Optional) — If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.
   - 'The build will produce test results' — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
     - 'Look in the standard test results directory' — Select this option if Bamboo should look in the build's standard test results directory.
     - 'Specify custom results directories' — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
       - 'Specify custom results directories' — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as * /test-reports/*.xml. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.
   - 'NCover output will be produced' — Select this check-box if you are running the NCover code coverage tool and want to view its code-coverage data from within Bamboo. The following field will be displayed:
     - 'NCover XML Directory' — Specify the name of the directory (including path) where Bamboo will look for the XML report output file from NCover. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.
   - 'Override default build hanging detection' — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
     - 'Build Time Multiplier' — This setting is used to calculate the 'Expected Build Time' for the build, i.e. 'Expected Build Time' = 'Build Time Multiplier' times 'Average Build Time' (note, the 'Average Build Time' is calculated by Bamboo by using an average of previous build times)
     - 'Log Quiet Time' — This is the amount of time since Bamboo last recorded an entry in the build log for a build.
       The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.
   - 'Build Queue Timeout' — This is the amount of time that a build will wait in a build queue before a timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the Build Queue Timeout Event).
   - 'Clover output will be produced' — Select this check-box if you are running Atlassian Clover and want to view its code-coverage data from within Bamboo (see Viewing the Clover Code-Coverage for a Build Result). The following field will be displayed:
     - 'Clover XML Directory' — Specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.
8. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.
PHPUnit

When you configure a plan, you need to specify which builder should be used for the plan's builds.

⚠️ You will need to install PHPUnit and reference the path to your PHP command-line interpreter, (e.g. `/usr/bin/phpunit` on Ubuntu), to use this builder.

To configure a PHPUnit builder for a plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 🍀
4. The 'Configuration' tab will be displayed. Click the 'Builder' sub-tab.
5. The 'Builder' sub-tab will be displayed (see screenshot below).
6. In the 'Builder' dropdown, select the PHPUnit builder that you wish to configure for this plan (e.g. "PHPUnit 3").

The builder that you select will become one of the plan's capability requirements. For details please see Specifying a Plan's Capability Requirements.

7. The screen will refresh to display the builder settings specific to PHPUnit:

- **Arguments** — Type the name of the directory/files that will be recursively analysed by PHPUnit. The default value is "." (i.e. the working subdirectory, if specified). You must specify at least one argument.
- **System Environment Variables** (Optional) — Specify any additional operating system environment variables you want to pass to your build; Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. "ANT_OPTS=-Xms200m -Xmx700m"). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).
  - i.e. additional to the existing environment variables (see Viewing Bamboo's System Information for a list). Note that existing environment variables are automatically available to the builder, thus you don't need to specify them in the 'System Environment Variables' field.
- **Working Sub Directory** (Optional) — If you leave this field blank, PHPUnit will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.
- **Log test execution in XML format to file** — Select this check-box if you want PHPUnit to record test results in JUnit format. (Note, this format is also used by TestNG.)
  - **Test Result File** — Enter the location of the file to record JUnit test results.
- **Write code coverage data in HTML format** — Select this check-box if you want PHPUnit to generate code coverage data in HTML format (e.g. for PHPUnit HTML Code Coverage reports).
  - **HTML Code Coverage Directory** — Enter the location of the directory to record HTML coverage data.
- **Clover output will be produced** — Select this check-box if you are running Atlassian Clover and want to view its code-coverage data from within Bamboo (see Viewing the Clover Code-Coverage for a Build Result). The following field will be displayed:
  - **Clover XML Directory** — Specify the name of the directory (including path) where PHPUnit will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. `/home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/`), i.e. please do not specify an absolute path.

8. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.

Script Builder

When you configure a plan, you need to specify which builder should be used for the plan's builds.

To specify script builder for a plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:

4. The 'Configuration' tab will be displayed. Click the 'Builder Configuration' sub-tab.

5. The 'Builder Configuration' sub-tab will be displayed (see screenshot below).

6. In the 'Builder' field, select the script builder that you wish to configure for this plan (e.g. "Script1").

    The builder that you select will become one of the plan's **capability requirements**. For details please see *Specifying a Plan's Capability Requirements*.

7. The screen will refresh to display the builder settings specific to script builders:
  - 'Script' — Specify the location of the script file. This can be either relative to the repository root of the plan, or absolute. You can include variables (see *Using Global or Build-specific Variables*).
  - 'Argument' — Specify the relevant argument to pass to the script. Note that arguments which contain spaces must be quoted. You can include variables (see *Using Global or Build-specific Variables*). |
8. Update the following general build parameters:
   
   - **'System Environment Variables' (Optional)** — Specify any additional* operating system environment variables you want to pass to your build: Please note, multiple variables must be separated with spaces, and parameters with spaces must be quoted (e.g. 'ANT_OPTS=-Xms200m -Xmx700m'). You can also include Bamboo global or build-specific variables (see Using Global or Build-specific Variables).
   
   * i.e. additional to the existing environment variables (see Viewing Bamboo's System Information for a list). Note that existing environment variables are automatically available to the builder, thus you don't need to specify them in the **'System Environment Variables'** field.
   
   - **'Working Sub Directory' (Optional)** — if you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the build's Working Directory, as described in Locating Important Directories and Files). You can override this option by specifying an alternative working directory (which must be a subdirectory of the root directory). For example, if your plan has a build script in a subdirectory, and the script needs to be run from within that subdirectory, you would type the name of that subdirectory in the 'Working Sub Directory' field.
   
   - **'The build will produce test results'** — Select this check-box if you want Bamboo to gather test results data for each build result. (Note that Bamboo requires test results to be XML files that are compatible with JUnit XML format. This format is also used by TestNG):
      
      - **'Look in the standard test results directory'** — Select this option if Bamboo should look in the Builder's standard test results directory.
      
      - **'Specify custom results directories'** — Select this option if the Builder will place generated test results in an alternative directory. The following field will appear:
        
        - **'Specify custom results directories'** — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as */test-reports/*.xml. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN), i.e. please do not specify an absolute path.
      
   - **'NCover output will be produced'** — Do not check this option. NCover is not relevant to Ant builders.
   
   - **'Override default build hanging detection'** — Select this check-box if you want to override the default build hanging detection settings. These settings determine when a build hung event is thrown (e.g. you can configure your notifications to trigger from this event). The following settings will be displayed:
      
      - **'Build Time Multiplier'** — This setting is used to calculate the 'Expected Build Time' for the build, i.e. 'Expected Build Time' = 'Build Time Multiplier' times 'Average Build Time' (note, the 'Average Build Time' is calculated by Bamboo by using an average of previous build times)
      
      - **'Log Quiet Time'** — This is the amount of time since Bamboo last recorded an entry in the build log for a build.

The **'Expected Build Time'** and **'Log Quiet Time'** must both be exceeded for Bamboo to throw the build hung event.

   - **'Build Queue Timeout'** — This is the amount of time that a build will wait in a build queue before an timeout event is thrown. Setting this value will override the global build queue timeout setting (see Configuring the Build Queue Timeout Event).

   - **'Use Clover to collect Code Coverage for this build.'** — Select this check-box if you are running Atlassian Clover and want to collect code-coverage data to view within Bamboo (see Viewing the Clover Code-Coverage for a Build Result). The following fields will be displayed:
      
      - **'Integration Options'** — Select one of the following options:
        
        - **'Automatically integrate Clover into this build'** — For this option, you have two sub options; 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views. You will also need to insert a Clover license (evaluation licenses are available) into the field provided.

        Arr issue currently exists preventing this option working on Windows. Please see Enabling the Clover Plugin for more information.

        - **'Clover is already integrated into this build and a clover.xml file will be produced'** — Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report. You will also need to specify where the Clover XML report is being generated, under **'Clover XML Location'**. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:

        
        ```
target/site/clover/clover.xml
        ```

        Do not specify an absolute path.

        - **'Clover XML Location'** — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN), i.e. please do not specify an absolute path.

      
9. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Capability Requirements.
Using Global or Build-specific Variables

On this page:

- Specifying Global, Build-specific or System Variables
  - Where can I use Global, Build-specific or System Variables?
- Using Capabilities as Variables
  - What Capabilities Can I Use Where?

Specifying Global, Build-specific or System Variables

When configuring a plan, you may want to specify variables to be used in the build process. There are three types of variables available to you:

- **Global variables** are defined across your entire Bamboo instance, and have the same (static) value for every plan that is built by Bamboo. You can define as many global variables as you wish (see Defining Global Variables).
- **Build-specific variables** are evaluated by Bamboo dynamically at build time. The source of a build-specific variable can either be a Bamboo property or one of the default plugins (assuming they have been enabled). The following build-specific variables are available by default:

<table>
<thead>
<tr>
<th>Build-specific variable</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildKey</td>
<td>Bamboo property</td>
<td>The plan key for the build, e.g. BAM–MAIN</td>
</tr>
<tr>
<td>buildNumber</td>
<td>Bamboo property</td>
<td>The Bamboo build number, e.g. 123</td>
</tr>
<tr>
<td>custom.svn.revision.number</td>
<td>Plugin</td>
<td>(For Subversion only) The revision number</td>
</tr>
<tr>
<td>custom.cvs.last.update.time</td>
<td>Plugin</td>
<td>(For CVS only) The last updated timestamp</td>
</tr>
<tr>
<td>custom.p4.revision.number</td>
<td>Plugin</td>
<td>(For Perforce only) The change set number</td>
</tr>
</tbody>
</table>

- **System variables** also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

The usage format for all global and build-specific variables is:

```
${bamboo.<variable>}
```

The usage format for all system variables is:

```
${system.<variable>}
```

e.g. if you have a system variable MYPATH=C:\MyPath; you can use a Bamboo system variable system.MYPATH which will inherit the same value as the system variable.

Where can I use Global, Build-specific or System Variables?

Variables can be used in the following fields of your build plan:

<table>
<thead>
<tr>
<th>Field</th>
<th>Available variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong> (for Maven builders only)</td>
<td>Global variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>Build-specific variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>System variables</td>
</tr>
<tr>
<td><strong>Build File</strong> (for Ant and NAnt builders only)</td>
<td>Global variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>Build-specific variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>System variables</td>
</tr>
<tr>
<td><strong>Target</strong> (for Ant and NAnt builders only)</td>
<td>Global variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>Build-specific variables</td>
</tr>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td>System variables</td>
</tr>
</tbody>
</table>
**Options** *(for NAnt builders only)*  
— see Specifying a Plan’s Builder

<table>
<thead>
<tr>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Script</strong> <em>(for Scripts only)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Argument</strong> <em>(for Scripts and Custom Commands only)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>System Environment Variables</strong></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Builder</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Repository URL</strong> <em>(for Subversion repositories only)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Source Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Web Repository URL</strong> <em>(for Subversion, CVS and Perforce repositories)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Source Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CVS Root</strong> <em>(for CVS repositories only)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Source Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Branch name</strong> <em>(for CVS repositories only)</em></th>
<th>Global variables</th>
<th>Build-specific variables</th>
<th>System variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>— see Specifying a Plan’s Source Repository</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Maven example**

For example, you may want your Maven 2 version to be determined by Bamboo. In Maven 2 *pom.xml* you may have:

```xml
...  
<groupId>com.atlassian.boo</groupId>
<artifactId>boo-test</artifactId>
<version>1.1.1102-SNAPSHOT</version>
...  
```

You can then specify the following in the ‘**Goal**’ field of your build plan:

```bash
clean package -DbambooBuildNumber=${bamboo.buildNumber}
```

When the command runs, Bamboo will replace the **buildNumber** with the actual number (e.g. 1102), which will be passed to the underlying Maven build to use. The command will then produce a jar that looks like this: *boo-test-1.1.1102-SNAPSHOT.jar*.

**Using Capabilities as Variables**

You can also specify a capability to be used in a similar way to a global variable.

The format of the capability should be as follows:

```bash
${{bamboo.capability.<capability_key>}}
```

For example,
If you click on a capability, the specific capability key will be contained in the URL.

Please note, the space characters in the URL will be replaced with '+' characters. We recommend that you do not use capability labels with space characters, if you wish to use them as variables. A possible solution for space characters is to format them with '${}' symbols, however, this does not work in all cases.

**What Capabilities Can I Use Where?**

Global and Build-Specific Variables can be used in a specific fields of your build plan, as specified above. For capabilities,

- **System Capabilities** are available to all of these fields, (i.e. global and build-specific).
- **Agent Capabilities** (i.e. agent-specific and shared/server capabilities) are available only to the build-specific fields. (i.e. not available to Repository URL, CVS Root or Branch name.)

For example,

If you wanted to specify a system variable, but have it set to different values on each agent, do the following:

1. Set the following as a system environment variable field on the 'Builder' tab:

   ```
   ${bamboo.capability.thatsystemvariable}
   ```

2. Specify the system environment variable as a custom capability on each of your agents, and set to the capability to the different values, as desired.

**Defining Global Variables**

When configuring a plan, you may want to specify variables to be used in the build process. For details please see Using Global or Build-specific Variables.

Global variables are one type of variable that is available to you. Global variables are defined across your entire Bamboo instance, and have the same value for every plan that is built by Bamboo.

**To add a new global variable,**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Global Variables' link in the left navigation column.
3. This will display a list of variables that have been configured in Bamboo, with an empty field at the bottom of each column for adding a new variable.
4. In the 'Key' field, type the key that you will use to identify the variable.
5. In the 'Value' field, type the value of the variable.
6. Click the 'Save' button.
Specifying a Plan's Capability Requirements

A **requirement** is an agent capability required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements.

There are four types of capabilities in Bamboo:

- **Builder capability** — Every plan has one 'Builder' capability requirement.
- **JDK capability** — A plan has either one 'JDK' capability requirement or none, depending on the plan's 'Builder'.
- **Custom capability** — A plan can have multiple 'custom' capability requirements (or none).
- **Perforce capability** — A plan will have a 'Perforce' capability requirement **automatically** specified, if Perforce was selected as the source repository.

### What are custom capabilities?

Custom capabilities can be used to control which build plans will be built by a particular agent. For example, if the builds for a particular plan should only run in a Windows environment, you could create a custom capability `operating.system=WindowsXP` for the appropriate agent(s), and specify it as a requirement for this plan.

Note that before you can specify them in your build plan, you need to first define your custom capabilities in your Bamboo system (see Configuring a new Custom Capability).

To specify a plan's **Builder capability requirement**,

Choose a value for the 'Builder' field on the plan's 'Builder Configuration' tab. Please see Specifying a Plan's Builder for instructions.

To specify a plan's **JDK capability requirement**,

Choose a value for the 'Build JDK' field on the plan's 'Builder Configuration' tab. Please see Specifying a Plan's Builder for instructions.

To specify a plan's **custom capability requirements**,
If you are creating a new plan, start at step 5.

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 📌
4. The 'Configuration' tab will be displayed. Click the 'Plan Requirements' sub-tab.
5. The 'Requirements' sub-tab will be displayed (see screenshot below), showing a list of all the plan's current capability requirements and a list of 'Capable Agents' (i.e. agents which meet the plan's requirements and are therefore capable of running a build for this plan).
6. If you have previously set up your custom requirement (e.g. on another project), you can select it from the 'Requirement' dropdown in the 'Add Extra Requirements' section and continue from step 9. If you are setting up a new custom requirement, select 'New custom requirement' instead and complete the following steps.
7. In the 'Add Extra Requirement' section, in the 'Key' field, type the key of the capability which you want this plan to require.
8. In the 'Add Extra Requirement' section, in the drop-down box below the 'Key' field, select one of the following:
   - 'exists' — this plan can be built by any agent that has a custom capability with the same Key as the one you have just specified.
   - 'equals' — this will display a text field for you to enter an exact value. This plan can be built by any agent that has a custom capability with the same Key and value.
   - 'matches' — this will display a text field for you to enter a regular expression (for more information about regular expressions, please visit http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html#sum). This plan can be built by any agent that has a custom capability with the same Key as the one you have just specified, where the value matches this regular expression.
9. Click the 'Add' button to add your new custom capability to the plan's list of requirements.
10. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, click the 'Next' button and go to Specifying a Plan's Build Artifacts.

Screenshot: 'Edit Capability Requirements'

The 'Capability Requirements' tab (see screenshot above) shows all of a plan's capability requirements: Builder, JDK and custom capabilities. It also shows which agents meet the requirements (i.e. are capable of building the plan).

- To see what capabilities a particular agent has, see Viewing an Agent's Capabilities

Specifying a Plan's Build Artifacts

A plan's artifacts are any reports, websites or files (e.g. JAR files) which you wish to keep after each build. Artifacts are copied to a subdirectory (/PLAN_KEY/download_data/) under your 'Build Directory' folder (see Locating Important Directories and Files). Artifacts which you define in the plan are listed in each build result as User-defined artifacts (see Viewing a Build's Artifacts in the Bamboo User's Guide).

To specify a plan's build artifacts,
If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon:
4. The ‘Configuration’ tab will be displayed. Click the ‘Build Artifacts’ sub-tab.
5. The ‘Build Artifacts’ tab will be displayed (see screenshot below).
   - In the ‘Artifact Label’ field, specify the name which Bamboo will use to describe the artifact.
   - In the ‘Source Directory’ field, specify the directory (including path) where Bamboo will look for your artifact.
   - The source directory is relative to the build directory, do not use absolute path to refer to the ‘source directory’
   - In the ‘Artifact Copy Pattern’ field, specify the name (or Ant file copy pattern) of the artifact(s) you want to keep.
   - The artifact copy pattern is relative to the source directory specified.
6. Click the ‘Save’ button if you are editing an existing plan; or if you are creating a new plan, click the ‘Next’ button and go to Specifying a Plan’s Notifications.

For example, if you want to keep the latest version of a JAR you have built, you could specify Artifact Copy Pattern to be ‘*.jar’ and the Source Directory to be ‘target’.

Screenshot: ‘Build Artifacts’

Specifying a Plan’s Notifications

Notifications in Bamboo are triggered by a range of events for a plan, including build completion, build outcomes and comments being posted against build results. You can configure whether notifications are sent for a particular event and who they are sent to. Users can choose whether to receive their notifications via email, IM, both or neither.

You must have the ‘Edit’ permission for a plan, to add or remove notifications for it.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘All Completed Builds’</td>
<td>Bamboo will send a notification whenever a build finishes for this plan, regardless of the build result. This trigger is recommended for any plans for which it is critical that people are always informed about the latest build activity. Many organisations start with this trigger, then change it later as they get more confidence in the continuous build process.</td>
</tr>
</tbody>
</table>
| ‘Failed Builds And First Successful’ | This trigger is generally suitable for the majority of plans. Bamboo will send a notification whenever:
   - a build fails for this plan.
   - the plan is “fixed” (that is, the plan’s latest build is successful and the previous build failed). |
<p>| ‘After X Failed Builds’      | This trigger enables you to specify the ‘Number Of Failed Builds’ after which Bamboo will send a notification. This is a useful way of limiting the number of notifications, if you are concerned about people receiving too many. You can also use this event to escalate build problems, e.g. notify a manager if a build fails five times. |</p>
<table>
<thead>
<tr>
<th>Notification Trigger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Hung</td>
<td>Bamboo will send a notification whenever it believes that a build has hung, according to the hung build criteria (read more about configuring your Building Hanging Detection settings). You can use this event to ensure that the right people are informed, if a build becomes unresponsive.</td>
</tr>
<tr>
<td>Build Commented</td>
<td>Bamboo will send a notification whenever a comment is posted against a build result. The email notification will contain all comments against the build, whereas the IM notification will only contain the comment that triggered the notification. This feature can help improve collaboration between team members. Please note that you will not receive notifications for any comments that you post.</td>
</tr>
<tr>
<td>Build Error</td>
<td>Bamboo will send a notification whenever an error occurs with the build process (i.e. the activities that Bamboo performs to run a build). This event is not related to failures of the actual build itself (see the 'Failed Builds And First Successful' event above). For example, a notification will be sent if Bamboo encounters an error when connecting to the repository, or detecting changes.</td>
</tr>
<tr>
<td>Build Queued Without Capable Agents</td>
<td>Bamboo will send a notification whenever a build is queued and there are no agents capable of building it. You can use this event to ensure that people are notified when changes to the agents adversely affect your builds.</td>
</tr>
<tr>
<td>Build Queue Timeout</td>
<td>Bamboo will send a notification whenever a build has been waiting in a build queue for longer than the build queue timeout criteria (read more about configuring your Build Queue Timeout settings). You can use this event to ensure that the right people are informed, if a build is stuck in a queue for too long.</td>
</tr>
</tbody>
</table>

For each plan, you can specify different recipients for each Notification Trigger. Note also that recipients need not be people with Bamboo user accounts.

⚠️ **Before you begin**
You need to configure Bamboo's SMTP email and/or instant messaging capabilities before Bamboo can send notifications.

To add notifications for a plan,
If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 📚.
4. The ‘Configuration’ tab will be displayed. Click the ‘Notifications’ sub-tab.
5. The ‘Notifications’ tab will be displayed. Click the ‘Configuration’ sub-tab.
6. Select the ‘Event’:
   - ‘All Completed Builds’
   - ‘Failed Builds and First Successful’
   - ‘After X Failed Builds’
   - ‘Build Commented’
   - ‘Build Hung’
   - ‘Build Queued Without Capable Agents’
   - ‘Build Queue Timeout’
7. Specify the ‘Recipient’:
   - ‘Committers’ — A committer is the Bamboo user(s) who committed code to a particular build (i.e. someone who committed code after the previous build was checked out by Bamboo).
   - ‘Watchers’ — A plan’s watchers are the Bamboo users who have marked this plan as one of their favourites.
   - ‘Group’ — Type the name of the appropriate Bamboo group(s).
   - ‘User’ — Type the username of the appropriate Bamboo user; or click the following icon to select from a list of users: 🧑‍💻.
8. ‘Email Address’ — This is useful if you need to send email notifications to a person who is not a Bamboo user. Type the appropriate email address. Please note, that if you specify the email address of an existing Bamboo user, the user will receive notifications, even if they have elected not to receive notifications in their user preferences.
9. ‘Instant Messaging Address’ — This is useful if you need to send IM notifications to a person who is not a Bamboo user. Type the appropriate IM address. Note that if you specify a broadcast address (e.g. ‘project-x@broadcast.chat.mycompany.com’), Bamboo will not know the context of related IM responses. Please note, that if you specify the IM address of an existing Bamboo user, the user will receive notifications, even if they have elected not to receive notifications in their user preferences.

6. Click the ‘Add’ button.
7. Repeat steps 5 and 6 until you have added all the build notifications that you wish to enable for this plan.
8. Click the ‘Done’ button if you are editing an existing plan; or if you are creating a new plan, click the ‘Next’ button and go to Specifying a Plan’s Post Actions.

Screenshot: ‘Build Notifications’
Specifying a Plan's Post Actions

For each plan, you can optionally specify actions that will occur after ('post') each build:

- what labels (if any) will be automatically applied to the plan's build results. For details please see Specifying Labels for a Plan's Build Results.
- this plan's specific schedule for deletion of build results (if different to the default). For details please see Specifying Expiry for a Plan's Build Results.

The actions listed above are supported out-of-the-box. There are a number of third-party plugin modules available that can provide additional actions at the completion of builds (e.g. Pre-Post Build Command plugin). You can also write your own plugins to provide additional post actions for a plan. See the Bamboo Plugin Guide for further details.

Specifying Expiry for a Plan's Build Results

By enabling build expiry, you can choose how much build results data will be kept in your Bamboo system, and for how long it will be kept (e.g. for reporting purposes), before being automatically deleted.

If you disable build expiry, your build results will never be automatically deleted from Bamboo.

You can enable/disable build expiry for:

- all plans (see Enabling Expiry of Build Results). This is generally the easiest way to manage your build expiry. Your settings will apply to all plans that do not have individual expiry settings.
- individual plans (as described below). You would generally only do this if there is a specific reason to keep/delete a particular plan's build results.

Note that you can also delete build results manually — see Deleting a Build Result.

If you enable build expiry, ensure that you back up your build results data before its expiry date is reached.

To enable expiry for a plan's build results,
If you are creating a new plan, start at step 5.

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 🌐
4. The 'Configuration' tab will be displayed. Click the 'Post Actions' sub-tab.
5. The 'Post Actions' tab will be displayed. Under 'Build Expiry', check the 'Override global build expiry configuration' checkbox. The following fields will display:
   - 'What should be expired?': select 'Artifacts' if you want to delete all user-defined artifacts but keep all other build results data. Select 'Build result' if you want to delete all build results data (including artifacts).
   - Choose one of these three methods for specifying how much data to keep, using the 'Expiry period' and 'Minimum builds to keep' fields:
     - To keep all build results up to a certain age,
       - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your build results, e.g. specify '24 months' to keep all build results for the last two years.
       - In the 'Minimum builds to keep' field, specify '0'.
     - To keep a certain number of build results per plan,
       - In the 'Expiry period' field, specify '0'.
       - In the 'Minimum builds to keep' field, specify the number of build results you want to keep, e.g. specify '50' to keep the latest 50 build results for each plan.
     - To keep all build results up to a certain age, and a certain number of build results per plan,
       - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your build results data, e.g. specify '24 months' to keep all build results for the last two years.
       - In the 'Minimum builds to keep' field, specify the number of build results you want to keep, e.g. specify '50' to keep the latest 50 build results for each plan. (This means that, even if all of a plan's builds are over two years old, the last 50 build results will not be deleted.)
   - If you wish to keep builds with particular labels, check the 'Don't expire builds with certain labels' checkbox. The 'Labels to keep' field will display:
     - In the 'Labels to keep' field, specify any labels for which you always want to keep labelled builds. (If you want to specify more than one label, use spaces to separate them.) For any label(s) that you specify, all builds that have a matching label will never be deleted, regardless of which of the three methods you followed for keeping and expiring data above.

   Please note, builds can either be labelled:
   - manually, as described in Labelling a Build Result in the Bamboo User's Guide; or
   - automatically, as described in Specifying Labels for a Plan's Build Results in the Bamboo Administrator's Guide.

6. Click the 'Save' button if you are editing an existing plan; or if you are creating a new plan, either click the 'Next' button and go to Specifying a Plan's Permissions.

When will the plan's build results be expired?

The build expiry event is a global event that runs periodically, regardless of whether you disable or enable build expiry in your plans. When this event occurs, the build results for your plan will be expired according to the criteria specified in the settings above or globally. To configure the global event and global build expiry settings, please see Enabling Expiry of Build Results.

Screenshot: 'Post Actions - Enabling Build Expiry'
To disable expiry for a plan’s build results data,

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 📊
4. The ‘Configuration’ tab will be displayed. Click the ‘Post Actions’ sub-tab.
5. The ‘Post Actions’ tab will be displayed. Under ‘Build Expiry’, check the ‘Override global build expiry configuration’ checkbox.
6. Enter ‘999999999 months’ in the ‘Expiry Period’ field.
   - This is a workaround for this existing Bamboo issue (BAM-4270). Please vote for this issue, if you would like to see it implemented in Bamboo.
7. Click the ‘Save’ button.
Specifying Labels for a Plan's Build Results

A label is a convenient way to tag and group build results that are logically related to each other. Labels can also be used to define RSS feeds and to control build expiry.

Labels can be applied to build results automatically, by specifying the label(s) in a build plan (note that only Bamboo administrators can do this). Labels can also be applied ad hoc to build results by Bamboo users.

To specify labels for a plan’s build results,

If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Post Actions’ sub-tab.
5. The ‘Post Actions’ tab will be displayed (see screenshot below).
6. In the ‘Regex Pattern’ field, you can either:
   • Specify a regular expression on which to match the log files. Labels will be applied to the build if the regular expression finds a match (see the examples below).
   ● For more information about regular expressions, please visit http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html#sum.
   • Leave this field blank to label every build.
7. In the ‘Labels’ field, type the word (or multiple words, separated by commas and/or spaces) with which the plan’s build results are to be labelled.
8. Click the ‘Save’ button if you are editing an existing plan; or if you are creating a new plan, either click the ‘Next’ button and go to Specifying a Plan’s Permissions.

Screenshot: ‘Post Actions’
Regex example:

```
There are \d+ results
```

In the above regex, "\d+" represents *any number with one or more digits.* ("\d" means ‘any digit’, and ‘+’ means ‘one or more times’. When combined, they mean ‘any sequence of one or more digits’.)

Therefore, positive matches would include:

- ‘There are 0 results’
- ‘There are 123 results’

**Regex example with multiple labels:**

You can use “capturing groups” with Bamboo 1.2.1 or later to create different labels for different purposes.

For example,

- Enter the following into the ‘**Regex Pattern**’ field:
  
  ```
  (PERFORMANCE_IMPROVED|PERFORMANCE_DETERIORATED)
  ```

- Enter the following into the ‘**Labels**’ field:
  
  ```
  \1
  ```

These settings will label your builds with PERFORMANCE_IMPROVED if “PERFORMANCE_IMPROVED” appears in the build log, and PERFORMANCE_DETERIORATED if “PERFORMANCE_DETERIORATED” appears in the build log. If both strings appear in a log, then both labels are applied to the build.

**Specifying a Plan’s Permissions**

A plan permission is the ability to perform a particular operation in relation to a build plan. For each plan, different permissions can be granted to particular groups and/or users.

The following plan permissions are available:

<table>
<thead>
<tr>
<th>Plan permission</th>
<th>Description</th>
<th>Can be granted to</th>
</tr>
</thead>
<tbody>
<tr>
<td>'View'</td>
<td>Permission to:</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>view this plan's build results</td>
<td></td>
</tr>
<tr>
<td></td>
<td>add comments or labels to this plan's build results¹</td>
<td></td>
</tr>
</tbody>
</table>

People who don't have the 'View' permission will not know that the plan exists.

<table>
<thead>
<tr>
<th>'Edit'</th>
<th>Permission to view and edit this plan's configuration, except for the plan's permissions.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>'Build'</th>
<th>Permission to:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>manually start/stop a build for this plan.</td>
</tr>
<tr>
<td></td>
<td>enable/disable this plan from submitting builds to the queue.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'Clone'</th>
<th>Permission to copy this plan when creating a new plan.</th>
</tr>
</thead>
</table>

(Note: only users with the 'Create Plan' global permission can create new plans.)

<table>
<thead>
<tr>
<th>'Admin'</th>
<th>Permission to:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>edit this plan's permissions.</td>
</tr>
<tr>
<td></td>
<td>delete this plan's build results and working files.</td>
</tr>
</tbody>
</table>

| - a particular user |
| - a particular group |
| - all logged-in users |
| - anonymous users ² |

¹ Only logged-in users (not anonymous users) can label or comment on a build result.

² Anonymous users cannot access Bamboo at all unless they have been granted the 'Access' global permission. See Allowing Anonymous Users to access Bamboo.

Anyone with the 'Admin' global permission automatically has all plan permissions for every plan.

The processes for granting and revoking plan permissions are described below. Note that, for ongoing ease of management, it is recommended that you grant permissions to groups rather than to individual users.

To grant plan permissions to a user,

**If you are creating a new plan, start at step 5.**

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Permissions' tab (see screenshot below).
5. In the 'Grant permission to' list at the bottom of the screen, select 'User'.
6. Type the username into the box, or click the following icon to select from a list of users: 

   ✅ Note that the assignment of permissions to LDAP users and groups in Bamboo is case sensitive. For instance, if the username of the LDAP user is 'Bob', you will need to type in 'Bob' (not 'bob' or 'BOB').
7. Click the 'Add' button.
8. The user will be added to the list of users on the 'Permissions' tab. Select the check-box for each permission that you wish to grant to this user.
9. Click the 'Save' button.

To grant plan permissions to a group,
If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Permissions’ tab (see screenshot below).
5. In the ‘Grant permission to’ list at the bottom of the screen, select ‘Group’.
6. Type the group name into the box.
  
  Note that the assignment of permissions to LDAP users and groups in Bamboo is case sensitive. For instance, if the name of the LDAP group is ‘Dev’, you will need to type in ‘Dev’ (not ‘dev’ or ‘DEV’).
7. Click the ‘Add’ button.
8. The group will be added to the list of groups on the ‘Permissions’ tab. Select the check-box for each permission that you wish to grant to this group.
9. Click the ‘Save’ button.

To grant plan permissions to all Bamboo users,

If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Permissions’ tab (see screenshot below).
5. Locate ‘Logged in users’ (under ‘Other’).
6. Select the check-box for each permission that you wish to grant to all Bamboo users.
7. Click the ‘Save’ button.

To grant plan permissions to anonymous users,

If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Permissions’ tab (see screenshot below).
5. Locate ‘Anonymous users’ (under ‘Other’).
6. Select the check-box for each permission that you wish to grant to all anonymous users.
7. Click the ‘Save’ button.

To revoke plan permissions,

If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 
4. The ‘Configuration’ tab will be displayed. Click the ‘Permissions’ tab (see screenshot below).
5. Locate the relevant user/group/all logged-in users/anonymous users.
6. Deselect the check-box for each permission that you wish to revoke from the user/group/all users/anonymous users.
  
  If you deselect all permissions for a user or group, that user or group will disappear from the ‘Permissions’ tab for this plan.
7. Click the ‘Save’ button.
Editing a Plan

To edit an existing plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click the 'Edit' icon.
4. The 'Configuration' tab will be displayed (see below). Click the nine sub-tabs to edit the following:
   - 'Plan Details' — see Renaming a Plan or Project.
   - 'Source Repository' — see Specifying a Plan's Source Repository.
   - 'Builder Configuration' — see Specifying a Plan's Builder.
   - 'Capability Requirements' — see Specifying a Plan's Capability Requirements.
   - 'Build Artifacts' — see Specifying a Plan's Build Artifacts.
   - 'Build Notifications' — see Specifying a Plan's Build Artifacts.
   - 'Post Actions' — see Specifying Labels for a Plan's Build Results and Specifying Expiry for a Plan's Build Results.
   - 'Dependencies' — see Setting up Build Dependencies.
   - 'Permissions' — see Specifying a Plan's Permissions.

Screenshot: 'Configuration'

Disabling or deleting a Plan

Sometimes, for example if a plan's latest build is broken and cannot be fixed quickly, you might need to temporarily stop the plan from being built. You can achieve this by disabling the plan, which will prevent it from submitting builds to the queue.

If a plan is no longer relevant, you have the option to completely delete it from your Bamboo system. To do this you will require the 'Admin' global permission. Note that deleting a plan will also delete all of the plan's build results, artifacts, labels and comments.

To disable a plan,

To delete a build that is currently in progress, see Stopping an Active Build.
1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click the plan name.
4. This will display the ‘Plan Summary’. Click the ‘Build Actions’ link (at the right of the page) and select ‘Disable Plan’.

To delete a plan,

**Before you begin**
If you need to keep a permanent record of the plan's build results, see Exporting Data for Backup.

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Remove Plans’ link in the left navigation column.
3. This will display a list of all plans in your Bamboo system. Select the check-box for the plan you wish to delete.
4. Click the ‘Delete’ button at the bottom of the list.
5. You will be prompted to confirm your deletion.

**Screenshot: 'Remove Plans’**

### Remove Plans

**Check the plans to be removed**

- When deleting plans from Bamboo, build history is unrecoverable for the deleted plans.

Check one or more of the following plans to be deleted:

- [ ] Select all plans and projects
- [ ] Bamboo (BAM)
- [ ] HEAD (MAIN)
- [ ] Stable Branch (STAB)
- [ ] LDAP Integration Tests (LDAP)
- [ ] Acceptance tests (BAMFUNC)

### Deleting a Build Result

If a build result is no longer relevant, you have the option to completely remove it from your Bamboo system. Note that you can also automatically remove build results that reach a particular age — see Enabling Expiry of Build Results.

To remove a build that is currently in progress, first see Stopping an Active Build.

### To delete a build result,

1. Go to the build result's plan. There are two ways to do this:
   a. Click ‘Home’ to go to the Dashboard, then click the ‘All Plans’ tab. Locate the plan in the list and click the plan name.
   OR:
   b. From the build result, click the plan name.
2. This will display the ‘Plan Summary’. Click the ‘Completed Builds’ tab.
3. This will display a list of all completed build results for this plan (see screenshot below). Locate the relevant build result and click the ‘Delete’ icon.
4. The build result, and any artifacts created by the build, will be deleted.
Only people with the 'Admin' global permission or the 'Admin' plan permission can delete build results.

Deleting a Plan's Working Files

If you want to ensure a clean check-out of your source code when Bamboo runs the next build for a particular plan, you will need to delete the plan's current Working Files.

To delete a plan's working files,

1. Click 'Home' to go to the Dashboard, then click the 'All Plans' tab.
2. Locate the plan in the list and click the plan name.
3. Click the 'Files' tab.
4. A list of the plan's working files will be displayed. Scroll down to the bottom of the screen and click the 'Delete all build files' icon.

Only people with the 'Admin' global permission or the 'Admin' plan permission can delete Working Files.

Stopping an Active Build

If you are running Bamboo on Windows, it may only be possible to stop an active build by going to Windows Task Manager and ending the relevant processes.

To stop an active build,
1. Click ‘Home’ to go to the Dashboard.
2. Locate the relevant plan on the ‘All Plans’ tab
   OR:
   Locate the relevant plan on the ‘Current Activity’ tab.
3. Click the ‘Stop Build’ icon: ⏳

To start a build on demand, see Triggering a Build Manually.

To stop a plan from submitting builds to the queue, see Disabling or deleting a Plan.

Moving a Plan to a different Project

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans.

Each project has a Name (e.g. “CRM System”) and a Key (e.g. “CRM”). The Project Key is prefixed to the relevant Plan Keys, e.g. the “CRM” project could have plans “CRM-TRUNK” and “CRM-BRANCH”.

Moving a plan to a different project will therefore involve changing the plan's Project Key (as well as possibly the Plan Name and/or Plan Key), which will also change the build key for all of the plan's build results.

Moving a plan does not affect the plan's configuration, nor any comments or labels that have been applied to the plan's build results.

Before you begin

- Note that moving a plan will require Bamboo to re-index all its data, so your Bamboo system may run slowly for a few minutes.
- It is recommended that you backup your Bamboo build results before you move a plan — see Exporting Data for Backup.

To move a plan to a different project,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Move Plans’ link in the left navigation column.
3. This will display the ‘Move Build Plan Wizard’. Plans are listed under their project name, e.g. in Screenshot 1 below, the ‘Clover Build’ plan is listed under the ‘Atlassian Config’ project. Select the check-box for the plan you wish to move.
4. Select the ‘Destination Project’ from the drop-down box at the bottom of the list.
5. Click the ‘Move’ button.
6. This will display the ‘Configure New Plan Details’ screen (see Screenshot 2 below).
7. If the destination project already includes a plan with the same Plan Name, or if you want to change the Plan Name for some other reason, overtype the ‘New Plan Name’ field.
8. If the destination project already includes a plan with the same Plan Key, or if you want to change the Plan Key for some other reason, overtype the ‘New Plan Key’ field.
9. Click the ‘Move’ button.
**Move Build Plan Wizard**

You can move a plan to another project with this wizard. Simply select the plans you want to move and the destination project. As names and keys may conflict, you’ll then be asked to enter new names and keys for the plans. Note that because we are changing plan keys, this operation requires some slow operations (e.g. indexing of all builds) and may take a few minutes.

**Plans:**

- AllAssan Bucket (BUCKET)
  - Main Build (MAIN)

- AllAssan Config (CONFIG)
  - Main Build (MAIN)

- Clover Build (CLOVER)

- AllAssan Core (CORE)
  - Main Build (MAIN)

- AllAssan Event (EVENT)
  - Main Build (MAIN)

- AllAssan Extras (EXTRA)
  - Main Build (MAIN)

**Destination Project:**

- AllAssan Bucket

The project you want to move your plans to.

**Move**
Configure New Plan Details

Choose new build keys and build names

The following plans will be moved to project Atlassian Bucket. You can update the plan names and keys below.

The existing plans for the destination project include:
- Main Build (MAIN)

<table>
<thead>
<tr>
<th>Original Project</th>
<th>Original Name</th>
<th>New Name</th>
<th>Original Key</th>
<th>New Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlassian Config (CONFIG)</td>
<td>Closer Build</td>
<td>Closer Build</td>
<td>CLOVER</td>
<td>CLOVER</td>
</tr>
</tbody>
</table>

Move Cancel

Renaming a Plan or Project

To rename a plan or a project,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click the 'Edit' icon.
4. The 'Configuration' tab will displayed. Click the 'Plan Details' sub-tab.
5. The 'Plan Details' sub-tab will be displayed (see below).
6. In the 'Project Name' field, type the project's new name.
7. In the 'Build Plan Name' field, type the plan's new name.
8. Click the 'Save' button.

Note that a plan's Project Key and Build Plan Key are not editable, but can be changed as described in Moving a Plan to a different Project.

Screenshot: 'Edit Plan Details'

Modifying Multiple (Bulk) Plans

Bulk actions enable changes to be made to multiple plans at once. The following list details the available bulk actions:

- **Add new notification**
  This action allows you to add a new notification to multiple plans at once.
- **Update SVN repository URL**
  This action allows you to update the Subversion repository URL of multiple plans (with Subversion repositories) at once.
- **Update SVN credentials**
This action allows you to update the Subversion username, authentication type and details of multiple plans (with Subversion repositories) at once.

- **Update CVS root and credentials**
  This action allows you to update the CVS root, authentication type and details of multiple plans (with CVS repositories) at once.

- **Update CVS module**
  This action allows you to update the CVS module (containing the source code) for multiple plans (with CVS repositories) at once.

- **Update web repository URL**
  This action allows you to update the web repository URL of multiple plans at once.

Screenshots: Adding a notification to multiple plans example (click to view full-sized images)

To add a new notification to multiple plans,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Bulk Action’ link in the left navigation column under the ‘Plans’ subheading. The ‘Select Bulk Action’ screen will display.
3. Select ‘Add new notification’ from the ‘Select bulk action’ dropdown and click ‘Next’. The ‘Choose Plans’ screen will display.
4. Select the plans that you wish to add a new notification to and click ‘Next’. The ‘Perform Bulk Action’ screen will display.
5. Configure the new notification that you want to add to the previously selected plans and click ‘Next’. Please see Adding or Removing Notifications for a Plan if you need detailed information on adding new notifications.
6. A confirmation window will display. Click ‘Confirm’ to add your new notification to the previously selected plans.
7. The ‘Results’ page will display. Click ‘Done’ to return to the ‘Select Bulk Action’ screen.

To update the SVN repository URL for multiple plans,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Action' link in the left navigation column under the 'Plans' subheading. The 'Select Bulk Action' screen will display.
3. Select 'Update SVN repository URL' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
4. Select the plans that you wish to update the SVN repository URL for and click 'Next'. The 'Perform Bulk Action' screen will display.
5. Update the 'Repository URL' to the desired URL and click 'Next'. Please see the 'Subversion Configuration' section of the Subversion documentation for further information on the SVN repository URL.
6. A confirmation window will display. Click 'Confirm' to update the SVN repository URL for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

To update the SVN repository credentials for multiple plans,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Action' link in the left navigation column under the 'Plans' subheading. The 'Select Bulk Action' screen will display.
3. Select 'Update SVN credentials' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
4. Select the plans that you wish to update the SVN repository credentials for and click 'Next'. The 'Perform Bulk Action' screen will display.
5. Update the SVN credentials as desired and click 'Next'. Please see the 'Subversion Configuration' section of the Subversion documentation for further information on SVN credentials.
6. A confirmation window will display. Click 'Confirm' to update the SVN credentials for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

To update the CVS root and credentials for multiple plans,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Action' link in the left navigation column under the 'Plans' subheading. The 'Select Bulk Action' screen will display.
3. Select 'Update CVS root and credentials' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
4. Select the plans that you wish to update the CVS root and credentials for and click 'Next'. The 'Perform Bulk Action' screen will display.
5. Update the CVS root and credentials as desired and click 'Next'. Please see the 'CVS Configuration' section of the CVS documentation for further information on the CVS root and credentials.
6. A confirmation window will display. Click 'Confirm' to update the CVS root and credentials for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

To update the CVS module for multiple plans,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Action' link in the left navigation column under the 'Plans' subheading. The 'Select Bulk Action' screen will display.
3. Select 'Update CVS module' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
4. Select the plans that you wish to update the CVS module for and click 'Next'. The 'Perform Bulk Action' screen will display.
5. Update the details of the CVS module as desired and click 'Next'. Please see the 'CVS Configuration' section of the CVS documentation for further information on the CVS module.
6. A confirmation window will display. Click 'Confirm' to update the CVS module for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

To update the web repository URL for multiple plans,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Action' link in the left navigation column under the 'Plans' subheading. The 'Select Bulk Action' screen will display.
3. Select 'Update web repository URL' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
4. Select the plans that you wish to update the web repository URL for and click 'Next'. The 'Perform Bulk Action' screen will display.
5. Update the 'Web Repository URL' and 'Web Repository Module' as desired and click 'Next'. Please see the 'Common Repository Configuration' section of the Subversion, CVS or Perforce documentation for further information on the web repository URL.
6. A confirmation window will display. Click 'Confirm' to update the web repository URL for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

Configuring Agents and Capabilities

About Agents and Capabilities

An agent is a service that runs Bamboo builds. There are two types of agents:

- **local agents** run on the Bamboo server.
- **remote agents** run on computers other than the Bamboo server. An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2)

(Note: Local agents run in the server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.)
What is a capability?

A **capability** is a feature of an **agent**. A capability can be a:

- builder (e.g. Maven)
- JDK
- custom capability (a key-value property which defines a particular characteristic of an agent, e.g. `'operating.system=WindowsXP'` or `'fast.builds=true'`)
- Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists). See [Configuring Capabilities](#).

How are capabilities used?

A **requirement** is an **agent capability** required by a **build plan**.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See [Specifying a Plan's Capability Requirements](#).

How are builds distributed to agents?

An agent will consume a single plan at a time and block until that build is complete. If you'd like to build multiple plans concurrently on the Bamboo server then simply [setup multiple local agents](#). If the agents are remote, then you'll need to [install that number of agent instances](#) on the machine. Separate installations are required because each remote agent will need its own home and log directories.
How do capabilities affect the distribution of builds to agents?

Plan ACME submits build number 123 to the Build Queue.

The Build Queue forwards build no. ACME-123 to the next available agent whose capabilities meet Plan ACME's requirements.
Configuring Agents

An agent is a service that runs Bamboo builds. There are two types of agents:

- **local agents** run on the Bamboo server.
- **remote agents** run on computers other than the Bamboo server. An **elastic agent** is a remote agent that runs in the Amazon Elastic Cloud. 

---

Plan ACME

**Capability requirements**

- Builder = Maven
- JDK = 1.5
- Custom: fast.build = true

Build no. ACME-123

Plan ACME submits build number 123 to the Build Queue

**Build Queue**

The Build Queue forwards build no. ACME-123 to the next available agent whose capabilities meet Plan ACME's requirements.

**Note:** agent-specific capabilities override shared capabilities.

**BAMBOO SERVER**

- Local Agent 1
- Local Agent 2

**Network**

**Remote Agent 1**

**Remote Agent 2**

**Remote Agent 3**

**Agent-specific capabilities**

- JDK = 1.5
- Custom: fast.build=true

**Shared local capabilities**

- Builder = Ant
- JDK = 1.5

**Shared remote capabilities**

- Builder = Maven
- JDK = 1.4
Compute Cloud (EC2)

(Note: Local agents run in the server’s process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.)

- To view an agent, including the agent properties, capabilities and the plans that an agent can build, please see Viewing an Agent.
- To view the status of all of your agents, please see Monitoring Agent Status.
- To create a new agent, see:
  - Creating a Local Agent
  - Creating a Remote Agent
- To edit an existing agent’s details, see:
  - Editing an Agent’s Details
- To configure an existing agent’s capabilities, see:
  - Configuring Capabilities
- To disable or delete an agent, see:
  - Disabling or Deleting an Agent

Elastic Agents
If you are looking for information on elastic agents, please refer to the documentation on Working with Elastic Bamboo.

Note:

A capability is a feature of an agent. A capability can be a:

- builder (e.g. Maven)
- JDK
- custom capability (a key-value property which defines a particular characteristic of an agent, e.g. ‘operating.system=WindowsXP’ or ‘fast.builds=true’)
- Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists). See Configuring Capabilities.

Creating a Local Agent

An agent is a service that runs Bamboo builds. There are two types of agents:

- local agents run on the Bamboo server.
- remote agents run on computers other than the Bamboo server. An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2)

(Note: Local agents run in the server’s process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.)

One local agent was automatically created when you installed Bamboo.

To create a new local agent,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. This will display the ‘Agents’ screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system. Click the ‘Add Local Agent’ button.
4. The ‘Add Local Agent’ screen will be displayed (see screenshot below).
5. In the ‘Name’ field, type the name of the new agent. (Note that this name will be displayed on the Dashboard.)
6. In the ‘Description’ field, type a description of the new agent. (This description will only be displayed to administrators.)
7. Click the ‘Add’ button.
8. This will return you to the ‘Agents’ screen. Your new agent will appear in the ‘Local Agents’ list.
9. Your new local agent will inherit all local server capabilities that are defined in your Bamboo system. You can also define agent-specific capabilities (note that these will override shared capabilities) --- see:
   - Configuring an Agent-specific Builder Capability
   - Configuring an Agent-specific JDK Capability
   - Configuring an Agent-specific Custom Capability

Your new agent will be able to run builds for all plans whose requirements are met by the agent’s capabilities (see Specifying a Plan’s Capability Requirements).
Creating a Remote Agent

Before you begin:

- Not sure whether to install a remote agent? See About Agents to understand how remote agents interact with your Bamboo server.
- Ensure that you have specified the Broker URL, as described in the Bamboo Setup Wizard and the Bamboo 2.0 Upgrade Guide.
- Do you have sufficient agent licenses? See Bamboo licensing for details.
- Have you enabled the creation of remote agents, as described in Disabling Remote Agents Support.
- Ensure that you have Java Runtime Environment 5.0 or later installed on the agent machine.
- Have you implemented your own remote agent service wrapper? You may not want to use the remote agent supervisor that is bundled with the remote agent. You can choose to install the legacy remote agent (pre-Bamboo 2.2) instead, which does not have a service wrapper.

On this page:

- Step 1. Download and install the Remote Agent
- Step 2. Launch the Remote Agent
  - Changing where the remote agent stores its data
  - Changing the logging on the remote agent
  - Suppressing the self-signed certificate of the server
  - Running Bamboo without the Remote Agent Supervisor
  - Running the remote agent with different start-up commands
- (Windows only) Installing the remote agent as a Windows service
- Step 3. Configure the Remote Agent's Capabilities
- Step 4. (Optional) Rename the Remote Agent

To install the Bamboo Remote Agent manually,

Step 1. Download and install the Remote Agent
1. Create a directory on the agent machine (e.g. `bamboo-agent-home`) to serve as the Bamboo agent home for the remote agent.

2. On your Bamboo server, click the ‘Administration’ link in the top navigation bar.

3. Click the ‘Agents’ link in the left navigation column.

4. This will display the ‘Agents’ screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system.

5. Click the ‘Install Remote Agent’ link. The following screen will be displayed:

   ![Image showing the Install Remote Agent screen]

   Click the ‘DOWNLOAD Remote Agent JAR’ button and save the JAR file to the directory you created in step 1.1.

6. Note the command under the heading ‘Running a Remote Agent’ for use in step 2 below.

   ![Image showing the Running a Remote Agent section]

   Step 2. Launch the Remote Agent

Once installed, you can run the remote agent by executing the command line obtained in the previous step. This command will look something like the following:

```java
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
http://bamboo-host-server:8085/agentServer/
```

The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.

You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or suppress the self-signed certificate of the server.

### Changing where the remote agent stores its data

By default, the remote agent will store its data in a directory called `bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter before the JAR file name:

```java
-Dbamboo.home=RemoteAgentHome
```

where `RemoteAgentHome` is the path to the Bamboo agent home directory you created in step 1.1.

Your command line will look something like this:

```java
java -Dbamboo.home=RemoteAgentHome -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
http://bamboo-host-server:8085/agentServer/
```

The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.

### Changing the logging on the remote agent

You can also specify different log files, file location, logs level and more to change the logging on the remote agent.
By default, the remote agent will use the same logging level as the Bamboo server. However, you can control the level of logging of your remote agent independently of your Bamboo server by setting up a separate logging configuration file.

Please see Configuring Logging in Bamboo for further details.

**Suppressing the self-signed certificate of the server**

If your Bamboo server uses SSL (https) with a self-signed certificate, you will need to carry out one of the following two options:

- **Add the following parameter** `-Dbamboo.agent.ignoreServerCertName=true` **to the remote agent's command line**, for example:
  ```
  java -Dbamboo.agent.ignoreServerCertName=true -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
  http://bamboo-host-server:8085/agentServer/
  ```
  Please be aware that this **reduces the security of your configuration**, as the identity of your Bamboo server will not be authenticated by the remote agent.

- **Use the keytool utility to add the self-signed certificate to the trusted certificates in your keystore**. This is a more secure option, but is complex to set up. For detailed instructions of how to do this, please refer to the relevant Sun documentation.

**Running Bamboo without the Remote Agent Supervisor**

The **remote agent supervisor** is included in the remote agent JAR bundled with Bamboo. The appropriate remote agent supervisor for the operating system of your remote machine, will be automatically installed when you run the default remote agent start-up command line.

>!* The remote agent supervisor cannot be installed on a small number of operating systems (i.e. the remote agent will start without the remote agent supervisor). If the remote agent supervisor fails to install, please check the operated systems list on the remote agent supervisor page. If your operating system is on the list and the remote agent supervisor still fails to install, please raise a support request in the Bamboo project.*!

If you need to run the remote agent without running the remote agent supervisor, you can execute the 'classic' version of the remote agent JAR.

The 'classic' agent jar is available from bamboo's agent installation page for download. Follow the steps below to run the 'classic' version of the remote agent:

1. Browse to
   ```http://<host>:8085/admin/agent/addRemoteAgent.action```

2. Click the 'the direct agent JAR is available at bamboo-agent-2.2.2.jar.' link and save 'classic' agent jar.

3. Start the agent with
   ```
   java -jar bamboo-agent-2.2.2.jar http://<host>:8085/agentServer/
   ```

>!* The name of the jar file (e.g. bamboo-agent-2.2.2.jar) will vary depending on the version of Bamboo you are running.*!

**Running the remote agent with different start-up commands**
The **remote agent supervisor** is executed by default when you run the default remote agent start-up command line. The **remote agent supervisor** is implemented via a Java Service wrapper. The wrapper allows you to execute a number of general start-up commands when the remote agent is run. These commands are appended to the end of the default remote agent start-up command line, i.e.

```java
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where `<wrapper_command>` is one of the keywords described below:

- **console** — runs the remote agent in the foreground, i.e. display all of the commands on the screen. The **agent home directory** will be populated **only** if it is empty. This parameter is used by default.
- **start** — runs the remote agent in the background, i.e. no commands are displayed on screen. If you have installed the remote agent as a **Windows service**, this command will work with the service.
- **stop** — stops a remote agent that is running. If you have installed the remote agent as a **Windows service**, this command will work with the service.
- **status** — (non-**Windows OS only**) returns the status of the remote agent, e.g. “Remote agent is not running.”
- **install** — installs the files for the remote agent, but does not start it. This will overwrite any changes that have been made to the **wrapper.conf** file. The **agent home directory** will be populated, regardless of whether it is empty or not, i.e. existing files will be overwritten. You may wish to use this option, if you want to customise the remote agent files before starting it.

> The name of the jar file (e.g. atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar) will vary depending on the version of Bamboo you are running.

**(Windows only) Installing the remote agent as a Windows service**

The **remote agent supervisor** is executed by default when you run the default remote agent start-up command line. The **remote agent supervisor** is implemented via a Java Service wrapper. The wrapper allows you to install or uninstall the remote agent as a service in Windows (i.e. start the Bamboo remote agent automatically when the machine boots). This is done by appending the appropriate wrapper commands to the end of the default remote agent start-up command line, i.e.

```java
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where `<wrapper_command>` is one of the keywords described below:

- **installntservice** — (**Windows only**) installs the remote agent as a Windows service.
- **uninstallntservice** — (**Windows only**) uninstalls the remote agent as a Windows service.

> The name of the jar file (e.g. atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar) will vary depending on the version of Bamboo you are running.

If you have installed the NT service, you will be able to use the **start** and **stop** start-up console commands with the service.

**Step 3. Configure the Remote Agent’s Capabilities**

Please see Configuring Capabilities.

**Step 4. (Optional) Rename the Remote Agent**

Your new remote agent has been automatically given a default name (e.g. 'Remote Agent on mymachine'). If you wish to rename your new remote agent, please see Editing an Agent’s Details.

**Disabling Remote Agents Support**
Disabling remote agent support in Bamboo will disable all remote agents and prevent any users from creating new remote agents. This function will not delete any remote agents that you have already created. To delete a remote agent, see Disabling or Deleting an Agent.

**Please note, remote agent support must be enabled to use Elastic Bamboo. Disabling remote agent support will disable Elastic Bamboo.**

### To disable remote agent support,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. This will display the 'Agents' screen. Click the 'Disable Remote Agent Support' link (see screenshot below).

**Screenshot: 'Disabling Remote Agent Support'**

<table>
<thead>
<tr>
<th>Remote Agents</th>
<th>Shared Remote Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote agents run on computers other than the Bamboo server.</td>
<td>Install Remote Agent Disable Remote Agent Support</td>
</tr>
<tr>
<td>There are currently no remote agents configured on this Bamboo instance.</td>
<td></td>
</tr>
</tbody>
</table>

If you wish to re-enable remote agent support, please follow the steps below:

### To re-enable remote agent support,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. This will display the 'Agents' screen with a message indicating that remote agent support is disabled. Click the 'Enable Remote Agent Support' link (see screenshot below).

**Screenshot: 'Re-enabling Remote Agent Support'**

**Remote Agents**

Remote agents run on computers other than the Bamboo server.

![Enable Remote Agent Support]

**Support for remote agents is disabled.**

Before enabling the remote agent support, please carefully consider the [security implications](#).

**Editing an Agent’s Details**

### To edit an agent’s name or description,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all local agents and remote agents in your Bamboo system. Click the 'Edit' link in the 'Operations' column of the agent you wish to edit.
4. The 'Edit Details' page for the agent will be displayed.
5. In the 'Name' field, type the agent's new name.
6. In the 'Description' field, type the new description of the agent.
7. Click the 'Save' button.

To edit an agent’s capabilities, see:

- Configuring an Agent-specific Builder Capability
- Configuring an Agent-specific JDK Capability
- Configuring an Agent-specific Custom Capability
Disabling or Deleting an Agent

Bamboo allows you to disable or delete an agent, when you do not want an agent to run any further builds.

- **Disabling an agent** lets you keep the agent in Bamboo, but stops it from running builds.
  - Sometimes you might need to prevent Bamboo from building any plans at all (e.g. while you re-index Bamboo). You could achieve this by disabling all agents. If you do this, all builds will wait in the queue until you re-enable the agents.
- **Deleting an agent** removes it from Bamboo altogether. If you need to use the agent again in future, you will need to recreate it (see Creating a Local Agent and Creating a Remote Agent).

Note that you can also delete/disable individual plans — this prevents the plan(s) from submitting builds to the queue. See Disabling or deleting a Plan.

**To disable an agent,**

1. Click the **Administration** link in the top navigation bar.
2. Click the **Agents** link in the left navigation column.
3. This will display a list of all agents in your Bamboo system. The **Status** column indicates which agents are currently enabled/disabled.
4. Locate the relevant agent and click the corresponding **Disable** link in the **Operations** column.

**To delete an agent,**

1. Click the **Administration** link in the top navigation bar.
2. Click the **Agents** link in the left navigation column.
3. This will display a list of all agents in your Bamboo system.
4. Locate the relevant agent and click the corresponding **Delete** link in the **Operations** column.

Viewing an Agent

An **agent** is a service that runs Bamboo builds. There are two types of agents:

- **local agents** run on the Bamboo server.
- **remote agents** run on computers other than the Bamboo server. An **elastic agent** is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2)

(Note: Local agents run in the server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own
Viewing an Agent's Capabilities

A capability is a feature of an agent. A capability can be a:

- builder (e.g. Maven)
- JDK
- custom capability (a key-value property which defines a particular characteristic of an agent, e.g. 'operating.system=WindowsXP' or 'fast.builds=true')
- Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

How capabilities are used:
A requirement is an agent capability required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See Specifying a Plan's Capability Requirements.

To view an agent's capabilities,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system. Click the name of the agent in which you are interested.
4. The 'Agent' page will be displayed (see screenshot below), showing a list of all shared capabilities and agent-specific capabilities (if any exist) that apply to this agent in the 'Capabilities' tab. The capabilities will be grouped under the 'Agent-Specific Capabilities' and 'Shared Capabilities' sections, as follows:

   - 'Builder' — builder capabilities. This sub-section will only display, if you have builder capabilities defined.
   - 'JDK' — JDK capabilities. This sub-section will only display, if you have JDK capabilities defined.
   - 'Custom' — custom capabilities. This sub-section will only display, if you have custom capabilities defined.
   - 'Perforce' — perforce capability. This sub-section will only display, if you have a Perforce capability defined.

To define a new capability, see Configuring Capabilities.

Screenshot: 'Agent Capabilities'
Viewing the Plans that an Agent can build

A Bamboo plan (or build plan) is the "recipe" for a build.

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project.

Determining which plans an Agent can build:
A requirement is an agent capability required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See Specifying a Plan's Capability Requirements.
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system. Click the name of the agent in which you are interested.
4. The 'Agent' page will be displayed. Click the 'Executable Plans' tab.
5. The plans that the agent is capable of building will be listed (see screenshot below).

To determine which agents are capable of building which plans, see Determining which Agents can build which Plans.

**Screenshot: 'Agent — View Executable Plans'**

**Agents > Local Agent 1 (local)**

**Description of Local Agent 1**

**Information**

Current status: Idle

View Recent Activity | Edit Details | Disable | Delete

**Capabilities**

**Executable Plans**

This agent meets the requirements to execute builds for the following plans:

1. Bamboo - Acceptance tests
2. Bamboo - Distribution
3. Bamboo - HEAD
4. Bamboo - HEAD Coverage
5. Bamboo - HEAD Nightly
6. Bamboo - LDAP Integration Tests
7. Bamboo - Remote Agent Functional Tests
8. Bamboo - Stable Acceptance Tests
9. Bamboo - Stable Branch
10. Bamboo 2.0 on Win32 - Branch

**Determining which Agents can build which Plans**

Unable to render [include] Couldn't find a page to include called: ALLDOC:/Draft warning

Determining whether an agent can execute builds for a particular plan depends on whether the capabilities of an agent match the requirements specified for a plan. Read more on Specifying a Plan's Capability Requirements and Configuring Capabilities.

The 'Agents and Plans Matrix' page displays a matrix of which agents are capable of building which plans. This matrix will also display elastic images, as elastic agents inherit their capabilities from the images that they are created from.

To view which agents can build which plans,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agent Matrix' link in the left navigation column.
3. The 'Agents and Plans Matrix' page will be displayed (see screenshot below) showing a matrix of all the plans currently set up, including disabled plans. Every shared agent, remote agent and elastic image will be listed against each plan with either a tick (agent is capable of building this plan) or a cross (agent is not capable of building this plan).

To view the plans that a specific agent can build, please also see Viewing the Plans that an Agent can build.

4. If an agent is not capable of building a particular plan, hover your mouse over the cross to see which plan requirements are not being met.

**Screenshot: 'Agents and Plans Matrix'**
To view the system properties for an agent,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system. Click the name of the agent in which you are interested.
4. The 'Agent' page will be displayed. Click the 'System Properties' tab.
5. The plans that the agent is capable of building will be listed (see screenshot below).

**Agents and Plans Matrix**

The matrix below shows which Bamboo agents can execute which plans. Each column represents an agent while each row represent a build plan. If you have enabled Elastic Bamboo, your elastic image will also be displayed in a column below. Elastic agents will derive their capabilities from the elastic image. Hover your mouse over any ✗ icons to see which plan requirements are not being met by the relevant agent.

<table>
<thead>
<tr>
<th></th>
<th>Default Agent</th>
<th>Remote Agent on sydney.atlassian.com(3)</th>
<th>Elastic Image: ami-55997e3c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo - Functional Tests</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Bamboo - HEAD</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bamboo - Install</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bamboo - Tests</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1. bamboo.functionalTest equal true

*Screenshot: 'Agent — View System Properties'*
Monitoring Agent Status

You can monitor your agents’ status to check that all agents are functioning as expected.

To see which plans are currently being built, look at the ‘Current Activity’ tab on the Dashboard.

To monitor the status of your agents,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. This will display the ‘Agents’ screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system (see screenshot below). Each agent has one of the following statuses:
   - ‘Offline’ — not currently running (Note: only applicable to Remote Agents)
   - ‘Idle’ — available to execute builds
   - ‘Building’ — currently executing a build
   - ‘Cancelling’ — currently cancelling a build
   - ‘Disabled’ — not available to execute builds (see Disabling or Deleting an Agent)
   - ‘Disabled - Building’ — currently executing a build, but disabled so cannot execute further builds
   - ‘Disabled - Cancelling’ — currently cancelling a build, and disabled so cannot execute further builds

Screenshot: ‘Agents’
Configuring Capabilities

A **capability** is a feature of an **agent**. A capability can be a:

- builder (e.g. Maven)
- JDK
- custom capability (a key-value property which defines a particular characteristic of an agent, e.g. `operating.system=WindowsXP` or `fast.builds=true`)
- Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

- **To find out what capabilities an agent already has**, please see Viewing an Agent's Capabilities.
- **To define a new capability**, see:
  - Configuring a new Builder
  - Configuring a new JDK
  - Configuring a new Custom Capability
  - Configuring a new Perforce Capability
- **To edit an existing capability**, see:
  - Editing a Capability
- **To rename an existing capability**, see:
  - Renaming a Capability
- **To delete a capability**, see:
  - Deleting a Capability
- **To view the agents and plans related to a capability**, see:
  - Viewing the Agents & Plans Related to a Capability

Note:

A **requirement** is an **agent capability** required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See Specifying a Plan's Capability Requirements.

Configuring a new Builder

At least one builder was automatically configured when you installed Bamboo¹. You can configure more by defining **builder capabilities**. Bamboo supports the following types of builders:

- Ant
- Maven
- Maven 2
- NAnt
- devenv.com
- msbuild.exe²
- PHPUnit³
You can define a new builder capability for:

- **a specific local agent** --- see Configuring an Agent-specific Builder Capability
- **all local agents** --- see Configuring a Shared Builder Capability
- **a specific remote agent** --- see Configuring an Agent-specific Builder Capability
- **all remote agents** --- see Configuring a Shared Builder Capability

Note that if an agent has its own specific builder capability, the value will override the value of a shared builder capability of the same name (if one exists).

Once you have configured a new builder capability in your Bamboo system, its label (e.g. 'Ant') will appear in the 'Build Resources' tab when you configure a build plan (see Specifying a Plan's Builder). The builder that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a builder capability whose label is specified in the plan's 'Builder' field.

¹ This depends on the system environment variables (e.g. 'ANT_HOME=/opt/java/ant') that were present on the machine on which Bamboo was installed:

- On the Bamboo server, environment variables that were present during installation were saved as **local server capabilities** in Bamboo.
- On remote agents, environment variables that were present during installation were saved as **agent-specific capabilities** in Bamboo.

² You will need to install the .NET framework SDK and reference the default path for msbuild.exe, (e.g. C:\Windows\Microsoft.NET\Framework*64*\v2.0.50727), to use this builder.

³ You will need to install PHPUnit and reference the path to your PHP command-line interpreter, (e.g. /usr/bin/phpunit on Ubuntu), to use this builder.

### Configuring an Agent-specific Builder Capability

Once you have configured a new builder capability in your Bamboo system, its label (e.g. 'Ant') will appear in the 'Build Resources' tab when you configure a build plan (see Specifying a Plan's Builder). The builder that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a builder capability whose label is specified in the plan's 'Builder' field.

An **agent-specific capability** applies to one agent only. Note that the value of an agent-specific capability will override the value of a **shared capability** of the same name (if one exists).

To configure a new agent-specific Builder capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to configure a new Builder capability.
5. The 'Agent Capabilities' screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent. Click the 'Add Capability' link in the top-right of the 'Agent-Specific Capabilities' section.
6. The 'Add Capability' page will be displayed (see screenshot below).
7. Set the 'Capability Type' field to 'Builder'.
8. Select the appropriate 'Type' from the drop-down list.
9. In the 'Label' field, type the name that you want to be displayed in the 'Builders' drop-down list when a plan is configured.
10. In the 'Path' field, type the appropriate path. This will vary depending on the 'Type' you selected in the previous step; relevant instructions will be shown below the 'Type'.
11. Click the 'Add' button. This will verify whether the Builder and Path you have specified are valid. If they are not, re-enter the values and then click the 'Add' button again.

*Screenshot: 'Add Capability — Builder'*
Configuring a Shared Builder Capability

Once you have configured a new builder capability in your Bamboo system, its label (e.g. ‘Ant’) will appear in the ‘Builder’ drop-down list on the ‘Build Resources’ tab when you configure a build plan (see Specifying a Plan’s Builder). The builder that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a builder capability whose label is specified in the plan’s ‘Builder’ field.

Shared capabilities are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an agent-specific capability of the same name (if one exists).

Please note, shared remote builder capabilities are not shared with elastic agents.

To configure a new local server Builder capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Server Capabilities’ link in the left navigation column.
3. The ‘Server Capabilities’ screen will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Locate the ‘Add Capability’ section below the list (see screenshot below).
5. Select ‘Builder’ from the ‘Capability Type’ dropdown.
6. Select the appropriate type of Builder from the ‘Type’ drop-down list.
7. In the ‘Label’ field, type the name that wish to appear in the ‘Builder’ drop-down list when a plan is configured.
8. In the ‘Path’ field, type the appropriate path. This will vary depending on the ‘Type’ you selected in the previous step; relevant instructions will be shown below the ‘Type’.
9. Click the ‘Add’ button. This will verify whether the JDK and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.

To configure a new shared remote Builder capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. The ‘Agents’ page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the ‘Shared Remote Capabilities’ link in the top right of the ‘Remote Agents’ section.
5. The ‘Shared Remote Capabilities’ screen will be displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Locate the ‘Add Capability’ section below the list (see screenshot below).
7. Select ‘Builder’ from the ‘Capability Type’ dropdown.
8. Select the appropriate type of Builder from the ‘Type’ drop-down list.
9. In the ‘Label’ field, type the name that wish to appear in the ‘Builder’ drop-down list when a plan is configured.
10. In the ‘Path’ field, type the appropriate path. This will vary depending on the ‘Type’ you selected in the previous step; relevant instructions will be shown below the ‘Type’.
11. Click the ‘Add’ button. This will verify whether the Builder and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.

Screenshot: ‘Add Capability — Builder’
Viewing your Builder Capabilities

You can view all the builder capabilities that have been defined in Bamboo on the 'Builders' page. These include local server capabilities, local agent-specific capabilities and remote agent-specific capabilities.

To view the Builder capabilities defined in Bamboo

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Builders' link in the left navigation column.
3. The 'Builders' page will be displayed, showing the list of all the Builder capabilities defined in your Bamboo system.
4. If you want to delete a Builder capability, click the 'Delete' link in the 'Operations' column for the Builder capability you wish to delete.
5. If you want to add the Builder as a local server capability, click the 'Add builder to server capabilities' at the bottom of the page to navigate to the 'Server Capabilities' page.

Configuring a new JDK

At least one JDK was automatically configured when you installed Bamboo¹. You can configure more by defining JDK capabilities.

You can define a new JDK capability for:
• a specific local agent --- see Configuring an Agent-specific JDK Capability
• all local agents --- see Configuring a Shared JDK Capability
• a specific remote agent --- see Configuring an Agent-specific JDK Capability
• all remote agents --- see Configuring a Shared JDK Capability

Note that if an agent has its own specific JDK capability, the value will override the value of a shared JDK capability of the same name (if one exists).

Once you have configured a new JDK capability in your Bamboo system, its label (e.g. ‘1.5’) will appear in the ‘Build JDK’ drop-down list on the ‘Build Resources’ tab when you configure a build plan (see Specifying a Plan's Builder). The JDK that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a JDK capability whose label is specified in the plan’s ‘Build JDK’ field.

¹This depends on the system environment variables (e.g. ‘JAVA_HOME=/opt/java/java_sdk1.5’) that were present on the machine on which Bamboo was installed:

• On the Bamboo server, environment variables that were present during installation were saved as shared local capabilities in Bamboo.
• On remote agents, environment variables that were present during installation were saved as agent-specific capabilities in Bamboo.

Configuring an Agent-specific JDK Capability

Once you have configured a new JDK capability in your Bamboo system, its label (e.g. ‘1.5’) will appear in the ‘Build JDK’ drop-down list on the ‘Build Resources’ tab when you configure a build plan (see Specifying a Plan's Builder). The JDK that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a JDK capability whose label is specified in the plan’s ‘Build JDK’ field.

An agent-specific capability applies to one agent only. Note that the value of an agent-specific capability will override the value of a shared capability of the same name (if one exists).

To configure a new agent-specific JDK capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. The ‘Agents’ page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to configure a new JDK capability.
5. The ‘Agent Capabilities’ screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent. Click the ‘Add Capability’ link in the top-right of the ‘Agent-Specific Capabilities’ section.
6. The ‘Add Capability’ page will be displayed (see screenshot below).
7. Set the ‘Capability Type’ field to ‘JDK’.
8. In the ‘Label’ field, type the name that you wish to appear in the ‘Build JDK’ drop-down list when a plan is configured.
9. In the ‘Java Home’ field, type the location of the JDK Home Directory.
10. Click the ‘Add’ button. This will verify whether the JDK and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.

Configuring Generic JDK Capabilities

If you want to indicate that an agent is capable of running builds for a set of related JDKs (e.g. all point versions of JDK 1.5), you set up generic JDK capabilities to encompass these JDKs.

For example, you can set up the following JDK capabilities for your Bamboo agent(s):

• JDK (where Label='JDK', Java Home='usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with any JDK requirement.
• JDK 1.5 (where Label='JDK 1.5', Java Home='usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is capable of running builds with a JDK 1.5 requirement or any point version of JDK 1.5, e.g. 1.5_0_07, 1.5_0_08, etc.
• JDK 1.5_0_07 (where Label='JDK 1.5_0_07', Java Home='usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is only capable of running builds with a JDK 1.5_0_07 requirement.

(If you have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.)
Configuring a Shared JDK Capability

Once you have configured a new JDK capability in your Bamboo system, its label (e.g. ‘1.5’) will appear in the ‘Build JDK’ drop-down list on the ‘Build Resources’ tab when you configure a build plan (see Specifying a Plan’s Builder). The JDK that you select when you configure a plan will be used for every build that is executed for that plan. That is, the plan can only be built by agents which have a JDK capability whose label is specified in the plan’s ‘Build JDK’ field.

Shared capabilities are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an agent-specific capability of the same name (if one exists).

Please note, shared remote JDK capabilities are not shared with elastic agents.

To configure a new local server JDK capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Server Capabilities’ link in the left navigation column.
3. The ‘Server Capabilities’ screen will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Locate the ‘Add Capability’ section below the list (see screenshot below).
5. Select ‘JDK’ from the ‘Capability Type’ dropdown.
6. In the ‘Label’ field, type the name that you wish to appear in the ‘Builder JDK’ drop-down list when a plan is configured.
7. In the ‘Java Home’ field, type the location of the JDK Home Directory.
8. Click the ‘Add’ button. This will verify whether the JDK and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.

To configure a new shared remote JDK capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. The ‘Agents’ page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the ‘Shared Remote Capabilities’ link in the top right of the ‘Remote Agents’ section.
5. The ‘Shared Remote Capabilities’ screen will be displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Locate the ‘Add Capability’ section below the list (see screenshot below).
7. Select ‘JDK’ from the ‘Capability Type’ dropdown.
8. In the ‘Label’ field, type the name that you wish to appear in the ‘Builder JDK’ drop-down list when a plan is configured.
9. In the ‘Java Home’ field, type the location of the JDK Home Directory.
10. Click the ‘Add’ button. This will verify whether the JDK and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.
Configuring Generic JDK Capabilities

If you want to indicate that an agent is capable of running builds for a set of related JDKs (e.g. all point versions of JDK 1.5), you can set up generic JDK capabilities to encompass these JDKs.

For example, you can set up the following JDK capabilities for your Bamboo agent(s):

- **JDK** (where Label='JDK', Java Home='/usr/java/jdk1.5.0_07')) — this JDK capability indicates that an agent(s) is capable of running builds with any JDK requirement.
- **JDK 1.5** (where Label='JDK 1.5', Java Home='/usr/java/jdk1.5.0_07')) — this JDK capability indicates that an agent(s) is capable of running builds with a JDK 1.5 requirement or any point version of JDK 1.5, e.g. 1.5.0_07, 1.5.0_08, etc.
- **JDK 1.5.0_07** (where Label='JDK 1.5.0_07', Java Home='/usr/java/jdk1.5.0_07') — this JDK capability indicates that an agent(s) is only capable of running builds with a JDK 1.5.0_07 requirement.

(If you have set up redundant JDK capabilities, you can view the list of JDK capabilities set up in Bamboo and delete any unwanted JDK capabilities.)

Viewing your JDK Capabilities

You can view all the JDK capabilities that have been defined in your Bamboo system on the 'JDKs' page. These include local server capabilities, local agent-specific capabilities and remote agent-specific capabilities.

To view the JDK capabilities defined in Bamboo

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'JDKs' link in the left navigation column.
3. The 'JDKs' page will be displayed, showing the list of all the JDK capabilities defined in your Bamboo system.
   - To view a JDK capability, click the 'View' link in the 'Operations' column for the JDK capability you wish to view. This will show you the agents and plans related to the JDK capability.
   - To edit a JDK capability, click the 'Edit' link in the 'Operations' column for the JDK capability you wish to edit.
   - To delete a JDK capability, click the 'Delete' link in the 'Operations' column for the JDK capability you wish to delete.
   - To add the JDK as a local server capability, click the 'Add JDK to server capabilities' at the bottom of the page to navigate to the 'Server Capabilities' page.
Configuring a new Custom Capability

Custom capabilities can be used to control which build plans will be built by a particular agent. For example, if the builds for a particular plan should only run in a Windows environment, you could create a custom capability 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this plan. (See Specifying a Plan's Capability Requirements.)

You can configure a new custom capability for:

- a specific local agent --- see Configuring an Agent-specific Custom Capability
- all local agents --- see Configuring a Shared Custom Capability
- a specific remote agent --- see Configuring an Agent-specific Custom Capability
- all remote agents --- see Configuring a Shared Custom Capability

Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

Configuring an Agent-specific Custom Capability

Custom capabilities can be used to control which build plans will be built by a particular agent. For example, if the builds for a particular plan should only run in a Windows environment, you could create a custom capability 'operating.system=WindowsXP' for the appropriate agent(s), and specify it as a requirement for this plan. (See Specifying a Plan's Capability Requirements.)

An agent-specific capability applies to one agent only. Note that the value of an agent-specific capability will override the value of a shared capability of the same name (if one exists).

To configure a new agent-specific custom capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to configure a new Custom capability.
5. The 'Agent Capabilities' screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent. Click the 'Add Capability' link in the top-right of the 'Agent-Specific Capabilities' section.
6. The 'Add Capability' page will be displayed (see screenshot below).
7. Set the 'Capability Type' field to 'Custom'.
8. In the 'Key' field, type the key for your new custom capability.
9. In the 'Value' field, type the value of your new custom capability.
10. Click the 'Add' button.
Configuring a Shared Custom Capability

Custom capabilities can be used to control which build plans will be built by a particular agent. For example, if the builds for a particular plan should only run in a Windows environment, you could create a custom capability `operating.system=WindowsXP` for the appropriate agent(s), and specify it as a requirement for this plan. (See Specifying a Plan’s Capability Requirements.)

*Shared capabilities* are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an agent-specific capability of the same name (if one exists).

Please note, *shared remote custom capabilities are not shared* with elastic agents.

To configure a new local server custom capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Server Capabilities’ link in the left navigation column.
3. The ‘Server Capabilities’ screen will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Locate the ‘Add Capability’ section at the bottom of the screen (see screenshot below).
5. In the ‘Capability Type’ field, select ‘Custom’.
6. In the ‘Key’ field, type the key for your new custom capability.
7. In the ‘Value’ field, type the value of your new custom capability.
8. Click the ‘Add’ button.

To configure a new shared remote custom capability,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column.
3. The ‘Agents’ page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the ‘Shared Remote Capabilities’ link (above the ‘Remote Agents’ list).
5. The ‘Shared Remote Capabilities’ screen will be displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Locate the ‘Add Capability’ section at the bottom of the screen (see screenshot below).
7. In the ‘Capability Type’ field, select ‘Custom’.
8. In the ‘Key’ field, type the key for your new custom capability.
9. In the ‘Value’ field, type the value of your new custom capability.
10. Click the ‘Add’ button.
Configuring a new Perforce Capability

If you wish to build plans on your server and remote agents using a Perforce repository, you need to specify the location of the Perforce P4 client application for your server and each remote agent using Perforce. These locations are set by specifying a mandatory local server Perforce capability for your server and agent-specific remote Perforce capabilities for each of your remote agents using Perforce.

**Shared capabilities** are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an agent-specific capability of the same name (if one exists).

An agent-specific capability applies to one agent only. Note that the value of an agent-specific capability will override the value of a shared capability of the same name (if one exists).

To configure a new local server Perforce capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Server Capabilities' link in the left navigation column.
3. The 'Server Capabilities' screen will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Locate the 'Add Capability' section at the bottom of the screen (see screenshot below).
5. In the 'Capability Type' field, select 'Perforce'.
6. In the 'Perforce Executable' field, type the location of the P4 client application for your server.
7. Click the 'Add' button.

To configure a new agent-specific remote Perforce capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to configure a new Custom capability.
5. The 'Agent Capabilities' screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent. Click the 'Add Capability' link in the top-right of the 'Agent-Specific Capabilities' section.
6. The 'Add Capability' page will be displayed (see screenshot below).
7. Set the 'Capability Type' field to 'Perforce'.
8. In the 'Perforce Executable' field, type the location of the P4 client application for your remote agents.
9. Click the 'Add' button.

**Screenshot: 'Add Capability — Perforce'**

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Perforce Executable</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perforce</td>
<td>/usr/local/bin/p4</td>
<td>Delete</td>
</tr>
</tbody>
</table>

**Editing a Capability**

**Before you begin:**
Because each agent can only run builds for plans whose requirements are met by the agent's capabilities (see Specifying a Plan's Capability Requirements), modifying a capability may mean that some plans can no longer be built.
To edit an agent-specific capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to edit an agent-specific capability.
5. The 'Agent Capabilities' screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
6. Click the 'Edit' corresponding to the agent-specific capability you wish to edit.
7. The 'Edit Capability' page will display. Depending on what type of capability you are editing, you will be able to modify the following values:
   - If you are editing a Builder capability, you can modify the 'Path' of the builder (see screenshot 'Edit Builder Capability' below).
   - If you are editing a JDK capability, you can modify the 'Java Home' of the JDK (see screenshot 'Edit JDK Capability' below).
   - If you are editing a Custom capability, you can modify the 'Value' of the capability (see screenshot 'Edit Custom Capability' below).
   - If you are editing a Perforce capability, you can modify the 'Perforce Executable' path (see screenshot 'Edit Perforce Capability' below).

To edit a local server capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Server Capabilities' link in the left navigation column.
3. The 'Server Capabilities' page will display, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Click the 'Edit' link corresponding to the capability you wish to edit.
5. The 'Edit Capability' page will display. Depending on what type of capability you are editing, you will be able to modify the following values:
   - If you are editing a Builder capability, you can modify the 'Path' of the builder (see screenshot 'Edit Builder Capability' below).
   - If you are editing a JDK capability, you can modify the 'Java Home' of the JDK (see screenshot 'Edit JDK Capability' below).
   - If you are editing a Custom capability, you can modify the 'Value' of the capability (see screenshot 'Edit Custom Capability' below).
   - If you are editing a Perforce capability, you can modify the 'Perforce Executable' path (see screenshot 'Edit Perforce Capability' below).

To edit a shared remote capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section.
5. The 'Shared Remote Capabilities' screen will display, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Click the 'Edit' link corresponding to the capability you wish to edit.
7. The 'Edit Capability' page will display. Depending on what type of capability you are editing, you will be able to modify the following values:
   - If you are editing a Builder capability, you can modify the 'Path' of the builder (see screenshot 'Edit Builder Capability' below).
   - If you are editing a JDK capability, you can modify the 'Java Home' of the JDK (see screenshot 'Edit JDK Capability' below).
   - If you are editing a Custom capability, you can modify the 'Value' of the capability (see screenshot 'Edit Custom Capability' below).
   - If you are editing a Perforce capability, you can modify the 'Perforce Executable' path (see screenshot 'Edit Perforce Capability' below).
Edit Capability - Maven 2
You can update the value for the capability on this page

**Capability Details**
- **Shared Capabilities:** Server Capabilities
- **Capability Type:** Builder
- **Label:** Maven 2
- **Path:** /opt/devtools/maven-2.0.5

[Update] [Cancel]

Edit Capability - JDK 1.5.0_13
You can update the value for the capability on this page

**Capability Details**
- **Shared Capabilities:** Server Capabilities
- **Capability Type:** JDK
- **Label:** JDK 1.5.0_13
- **Java Home:** /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home

[Update] [Cancel]

Edit Capability - OS
You can update the value for the capability on this page

**Capability Details**
- **Shared Capabilities:** Server Capabilities
- **Capability Type:** Custom
- **Key:** OS
- **Value:** Linux

[Update] [Cancel]
Renaming a Capability

To rename an agent-specific capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to edit an agent-specific capability.
5. The 'Agent Capabilities' screen will be display, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
6. Click the 'View' corresponding to the agent-specific capability you wish to rename.
7. The 'View Capability' page will display. Click the 'Rename Capability' link.
8. The 'Rename Capability' page will display. Update the 'Key' and click 'Rename Capability'.

To rename a local server capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to edit an agent-specific capability.
5. The 'Agent Capabilities' screen will be display, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
6. Click the 'View' corresponding to the agent-specific capability you wish to rename.
7. The 'View Capability' page will display. Click the 'Rename Capability' link.
8. The 'Rename Capability' page will display. Update the 'Key' and click 'Rename Capability'.

To rename a shared remote capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to edit an agent-specific capability.
5. Click the 'Remote Agents' section.
6. Click the 'View' corresponding to the agent-specific capability you wish to rename.
7. The 'View Capability' page will display. Click the 'Rename Capability' link.
8. The 'Rename Capability' page will display. Update the 'Key' and click 'Rename Capability'.

Screenshot: 'Rename Capability'
Deleting a Capability

Before you begin:
Because each agent can only run builds for plans whose requirements are met by the agent's capabilities (see Specifying a Plan’s Capability Requirements), deleting a capability may mean that some plans can no longer be built.

To delete an agent-specific capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent that you wish to delete an agent-specific capability from.
5. The 'Agent Capabilities' screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
6. Click the 'Delete' corresponding to the agent-specific capability you wish to delete.

To delete a local server capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Server Capabilities' link in the left navigation column.
3. The 'Server Capabilities' page will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Click the 'Delete' link corresponding to the capability you wish to delete.

To delete a shared remote capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section.
5. The 'Shared Remote Capabilities' screen will be displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Click the 'Delete' link corresponding to the capability you wish to delete.

Viewing the Agents & Plans Related to a Capability

You can view a capability to determine the following information about it:

- which agents have/inherit the capability.
- which build plans have the capability specified as a requirement.

To view an agent-specific capability,
To view a local server capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Server Capabilities' link in the left navigation column.
3. The 'Server Capabilities' page will display, showing a list of all local server capabilities currently defined in your Bamboo system.
4. Click the 'View' link corresponding to the capability you wish to view.
5. The 'View Capability' summary page will display (see example screenshot 'View Capability' below).

To view a shared remote capability,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click on the name of the agent for which you wish to view an agent-specific capability.
5. The 'Agent Capabilities' screen will display, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
6. Click the 'View' link corresponding to the capability you wish to view.
7. The 'View Capability' summary page will display (see example screenshot 'View Capability' below).

Please note, shared remote capabilities are not shared with elastic agents.

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The 'Agents' page will display, showing lists of all Local Agents and Remote Agents in your Bamboo system.
4. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section.
5. The 'Shared Remote Capabilities' screen will display, showing a list of all shared remote capabilities currently defined in your Bamboo system.
6. Click the 'View' link corresponding to the capability you wish to view.
7. The 'View Capability' summary page will display (see example screenshot 'View Capability' below).

**Screenshot: 'View Capability'**

**Builders > Maven 2**

The screen shows the summary of a capability. You can see which plans have a requirement on this capability and which agents have the capability.

<table>
<thead>
<tr>
<th>Agents with capability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agent</strong></td>
</tr>
<tr>
<td>All local agents</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plans with requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plan</strong></td>
</tr>
<tr>
<td>SVN Test - Default</td>
</tr>
</tbody>
</table>

**Working with Builds**

The following pages contain information on working with your builds. If you are looking for information on configuring a build plan, please also see Configuring Projects and Plans.

- Triggering a Build
Triggering a Build

About Build Triggering

There are a number of ways in which a build can be triggered for a plan,

- **Build Strategy:**
  - Code updated — a build can be triggered whenever one or more authors checks-in code.
  - Scheduled build — a build can be scheduled to occur at regular intervals.
  - Manual build — a build can be triggered manually.
  - Initial clean build — a build will be triggered when a new plan is created.

- **Build Dependency:**
  - Dependency — a build can be triggered whenever a successful build occurs for another plan.

The way in which each build was triggered is listed in the ‘Reason’ column on the Dashboard.

Build Dependencies and Build Strategies

Build dependencies are "parent-child" relationships between builds that can you can configure to trigger builds. See Setting up Build Dependencies.

Build dependencies work together with the build strategy to trigger builds. For example, you can set up your build plan to poll your repository for changes as well as configure build dependencies between builds. In this case, your builds will be triggered when code changes are detected as well as when parent builds complete successfully.

If you want your builds to only be triggered by successful parent builds from your build dependencies, you can do this by specifying "manual & dependent builds only" as the build strategy for your plan. See Triggering a Build Manually.

Considerations for choosing a Build Strategy

Code Updated

Triggering a build when code is updated ensures that a build only occurs when necessary. There are two ways to trigger a build when code is updated:

- "Pull strategy" — Polling the repository for code changes means that Bamboo will check-out the source-code on a regular basis, and examine it for changes. If Bamboo detects a change, it will trigger a build.
  See Polling the Repository for Code Changes.

- "Push strategy" — Triggering a build on code check-in has the advantage of placing minimal load on your Bamboo server, but requires that your source-code repository is configured to fire an event to the Bamboo server.
  See Triggering a Build on Code Check-in.

Scheduled Build
Triggering a build on schedule can allow a team to structure the day according to a predictable schedule. Note that scheduled builds are run regardless of whether or not any code changes have occurred. There are two ways to schedule a build:

- **Single Daily Build** — A *single daily build* runs at a time of your choice. This is particularly suitable for builds that take a long time to complete. See [Scheduling a Single Daily Build](#).
- **Cron-Based Scheduling** — A *cron-based schedule* allows you to schedule builds according to a flexible cron expression. For example, "0 0/30 9-19 ? * MON-FRI" would trigger a build every half-an-hour from 9am to 7pm, Monday to Friday. See [Specifying a Cron-based Schedule](#).

### Manual and Dependent Builds

Triggering a build manually or by build dependencies allows you to ensure that builds are only triggered manually or by parent builds of dependent builds (as described in the 'Build Dependencies and Build Strategies' section above). See [Triggering a Build Manually](#).

**Related Topics**

- Stopping an Active Build

### Triggering a Build when Code is Updated

Triggering a build when code is updated ensures that a build only occurs when necessary. There are two ways to trigger a build when code is updated:

- **"Pull strategy"** — Polling the repository for code changes means that Bamboo will check-out the source-code on a regular basis, and examine it for changes. If Bamboo detects a change, it will trigger a build. See [Polling the Repository for Code Changes](#).
- **"Push strategy"** — Triggering a build on code check-in has the advantage of placing minimal load on your Bamboo server, but requires that your source-code repository is configured to fire an event to the Bamboo server. See [Triggering a Build on Code Check-in](#).

### Polling the Repository for Code Changes

Polling the repository for code changes means that Bamboo will check-out the source-code on a regular basis, and examine it for changes. If Bamboo detects a change, it will trigger a build.

To poll the repository for changes,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. In the 'Build Strategy' field, select 'Poll the repository for changes' (see screenshot below).
6. In the 'Polling Frequency' field, specify how often (in seconds) Bamboo should check-out the source-code and examine it for changes.
7. Click the 'Save' button.

**Screenshot: 'Plan Configuration--Source Repository-Build Strategy: Poll the repository for changes'**
Triggering a Build on Code Check-in

Triggering a build on code check-in has the advantage of placing minimal load on your Bamboo server, but requires that your source-code repository is configured to fire an event to the Bamboo server.

Step 1. Configure your source repository
1. Configure your source-code repository to run post-commit scripts to tell Bamboo whenever a code commit has occurred.
   - For **CVS**, edit two files in the CVSROOT module: commitinfo and logininfo.
   - For **commitinfo** add a line like this:
     ```shell
     ^jira(/|$) /path/to/preCommit.sh
     ```
     where "jira" is your module.
   - For **logininfo**, add a line like this:
     ```shell
     ^jira(/|$) /path/to/postCommitBuildTrigger.sh %{} http://bambooserver
     JIRA-MAIN JIRA-BRANCH
     ```
     where JIRA-MAIN and JIRA-BRANCH are the Bamboo plans that you would like to trigger.
   - For **Subversion**, edit the Subversion repository's hooks/post-commit trigger file with something like:
     ```shell
     /path/to/postCommitBuildTrigger.sh http://bambooserver JIRA-MAIN JIRA-BRANCH
     ```
   - For **Perforce**, add the script as a change-commit trigger.
     ```shell
     triggerName change-commit //myDepot/... "/usr/local/bin/postCommitBuildTrigger.sh http://bambooserver/ MYPLAN-DEFAULT"
     ```

2. Copy the scripts to your repository. If you are using Bamboo Standalone, the scripts are located in the /scripts folder of your Bamboo Installation Directory. If you are using Bamboo EAR-WAR you can find them in the /repositoryScripts folder. You can also download the scripts by following this link.

3. Depending on which operating system your repository is running on, you may need to edit the scripts. The scripts assume that 'wget' is in '/usr/bin/'; if this isn't the case for your repository (e.g. Solaris 10 has it in /usr/sfw/bin/), edit the scripts and change '/usr/bin/' to the appropriate location.

4. Ensure that the user which Bamboo is running as has appropriate file permissions to execute the scripts, i.e. the scripts should be executable by non-root user(s).

5. Enable Bamboo's remote API so that the scripts can use Bamboo's REST-style remote API to access Bamboo's data.

---

### Step 2. Configure Bamboo to trigger a build on code check-in

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. In the 'Build Strategy' field, select 'Repository triggers the build when changes are committed' (see screenshot below).
6. This will display the 'Trigger IP Address' field. If you want Bamboo to receive post-commit notifications from the repository's primary IP address, leave the 'Trigger IP Address' field blank. If you want Bamboo to receive post-commit notifications from a different IP address, type the IP address in the 'Trigger IP Address' field.
7. Click the 'Save' button.

*Screenshot: Plan Configuration--Source Repository--Build Strategy: Repository triggers the build when changes are committed*
Triggering a Build on Schedule

Triggering a build on schedule can allow a team to structure the day according to a predictable schedule. Note that scheduled builds are run regardless of whether or not any code changes have occurred. There are two ways to schedule a build:

- **Single Daily Build** — A single daily build runs at a time of your choice. This is particularly suitable for builds that take a long time to complete. See Scheduling a Single Daily Build.
- **Cron-Based Scheduling** — A cron-based schedule allows you to schedule builds according to a flexible cron expression. For example, "0 0/30 9-19 ? * MON-FRI" would trigger a build every half-an-hour from 9am to 7pm, Monday to Friday. See Specifying a Cron-based Schedule.

Scheduling a Single Daily Build

A single daily build runs at a time of your choice. This is particularly suitable for builds that take a long time to complete. To schedule a single daily build,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. In the 'Build Strategy' field, select 'Single Daily Build' (see screenshot below).
6. In the 'Build Time' field, specify the time of day at which the build should run. Specify the time in "hh:mm" format, using a 24-hour clock.
7. Click the 'Save' button.

Screenshot: 'Plan Configuration--Source Repository--Build Strategy: Single daily build'
Specifying a Cron-based Schedule

A **cron-based schedule** allows you to schedule builds according to a flexible cron expression. For example, "0 0/30 9-19 ? * MON-FRI" would trigger a build every half-an-hour from 9am to 7pm, Monday to Friday.

To specify a cron-based schedule,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon: 
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. In the 'Build Strategy' field, select 'Cron Based Scheduling' (see screenshot below).
6. In the 'Cron Expression' field, type your cron expression. A cron expression consists of 6 mandatory and one optional field. The fields in sequential order are: seconds, minutes, hours, day-of-month, month, day-of-week and (optional) year.
7. Click the 'Save' button.

**Screenshot:** 'Plan Configuration--Source Repository--Build Strategy: Cron-based schedule'
To start a manual build,

1. Click 'Home' to go to the Dashboard.
2. Locate the relevant plan and click the 'Check Out and Build' icon:

   See also Stopping an Active Build.

You can specify that a plan should only ever be built manually or triggered by other builds. This is useful for:

- **Broken builds:**
  If a build is broken, you may want to temporarily specify 'manual & dependent builds only'. This means that a failing build will not be triggered frequently and hence will not take up time and processing power when other builds could be running.

- **Dependent builds:**
  If you specify that a build should run when another build successfully finishes, you may want to prevent it from running at other times. You can achieve this by specifying 'manual & dependent builds only'.

To specify 'manual & dependent builds only',

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:
4. The 'Configuration' tab will be displayed. Click the 'Source Repository' sub-tab.
5. In the 'Build Strategy' field, select 'Manual & dependent builds only' (see screenshot below).
6. Click the 'Save' button.
Setting up Build Dependencies

Sometimes you may want to trigger a build when another plan’s build has successfully completed. This ensures that changes to one plan’s code do not break a dependent build (known as a ‘child’ build).

For example, there could be two plans in Bamboo:

1. ‘ACME-CORE’ — which contains the core code for an application.
2. ‘ACME-PLUGIN’ — which contains code for a plugin to the application.

In this scenario, ACME-PLUGIN is a child of ACME-CORE. Any changes to the ACME-CORE code should trigger a build of Acme-PLUGIN. In turn, every time a build completes successfully for ACME-PLUGIN, you might want to then run builds for some additional plans (e.g. ‘ACME-FUNCTIONAL-TESTS’ and ‘ACME-UNIT-TESTS’). In this case, ACME-PLUGIN is a parent of ACME-FUNCTIONAL-TESTS and ACME-UNIT-TESTS, as well as being a child of ACME-CORE.

Build Dependencies and Build Strategies

Build dependencies work together with the build strategy to trigger builds. For example, you can set up your build plan to poll your repository for changes as well as configure build dependencies between builds. In this case, your builds will be triggered when code changes are detected as well as when parent builds complete successfully.

If you want your builds to only be triggered by successful parent builds from your build dependencies, you can do this by specifying “manual & dependent builds only” as the build strategy for your plan. See Triggering a Build Manually.

Tips for setting up dependent builds

Please take note of the following information when setting up dependent builds:

- Please take care not to create circular dependencies, where your child build triggers one of its parent builds. Otherwise your plans may build continuously.
- If you specify that a build should run when another build successfully finishes, you may want to prevent it from running at other times. You can achieve this by specifying ‘manual & dependent builds only’. See Triggering a Build Manually.
- If the child build uses the same source as the parent build (e.g. the Subversion URL is the same), the child build will be forced to check out the same revision of source code as the parent build. This ensures that builds are consistent when triggering one build from another.

To trigger a build when another build finishes,
1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in which you are interested.
4. Click this icon: 
5. The plan's 'Configuration' tab will be displayed. Click the 'Dependencies' sub-tab (see screenshot below).
6. This will display lists of all plans in your Bamboo system (see screenshot below). All of these are possible 'Parent plans' and 'Child plans' for the current plan.
   - In the 'Dependency Blocking Strategy', select the dependency blocking strategy that you want to apply to this build. Read more about Dependency Blocking Strategies.
   - Please note, dependency blocking only works when triggered by code updates.
   - In the 'Parent plans' list, select any plans for which a successful build should trigger a build for the current plan.
   - In the 'Child plans' list, select any plans for which a build should be triggered when the current plan's build finishes successfully.
7. Click the 'Save' button.

Screenshot: 'Plan Configuration — Dependencies'

Dependency Blocking Strategies

Dependency blocking is a advanced feature of dependent build triggering that can be used to manage builds with parents. This is particularly useful for ensuring that a "tree" of dependent builds always runs in order of the tree hierarchy, even if child builds are triggered independently of their parents.

⚠️ Please note, dependency blocking only works when triggered by code updates (i.e. polling or check-in). It will not work when a build is triggered on a schedule or triggered via a parent build (when there are multiple parent builds in progress).

The three dependency blocking strategies are explained below:

- 'Do not block' — If a build with this dependency blocking strategy is triggered by a code update, the build will run whenever it is triggered by a code update regardless of whether it has parent builds or not.
- 'Block build if parent builds are queued or in progress' — If a build with this dependency blocking strategy is triggered by a code update, Bamboo will check whether its parents are building or queued to build. If so, the build will be blocked.
- 'Block build if parent plans have unbuilt changes' — If a build with this dependency blocking strategy is triggered by a code update, Bamboo will check whether its parents are building, queued to build or have changes. If so, the build will be blocked. Additionally, if not,
Bamboo will check if any of the parent builds have changes. If there are parent builds with changes, they will be triggered and the build blocked. If there are no parent builds with changes, the build will run.

These dependence blocking strategies are illustrated in the flowchart below:

---

**Reordering your Build Queue**

Bamboo automatically assign builds to the build queue when they are triggered and no agents are available to run them.

If you want to prioritise one build over another in the build queue, you can manually reorder the build queue. Please note, this will not force your build to run immediately, it will just promote it in the build queue. Your build will still need an agent to become available that has the capabilities to meet its requirements. Similarly, you can demote a build in the build queue if you do not need it to run urgently.

---

**Build Queues and Build Dependencies**

Reordering the build queue only affects the priority of the build within the build queue. It does not change any build triggering dependencies for the build.

---

To reorder the build queue,
1. Log into Bamboo as an administrator.
2. Click the 'Current Activity' tab. The 'Build Queue' panel will display in the 'Agents' column on the left (see screenshot).
3. Click the 'Re-order' link in the 'Build Queue' panel. The panel will refresh displaying arrow icons against each build.
4. Reorder the build queue as follows:
   - Click the up arrow (↑) to move the build up one place in the build queue.
   - Click the down arrow (↓) to move the build down one place in the build queue.
   - Click the curly up arrow ( CURLY ↑ ) to move the build to the bottom of the build queue.
   - Click the curly down arrow ( CURLY ↓ ) to move the build to the bottom of the build queue.
5. Click the 'Operations' link to return your build queue to its original view.

**Monitoring Builds**

The following Bamboo features can help you monitor your running builds. Please see each page for details on configuring and using each feature.

- Configuring the Hanging Build Event
- Configuring the Build Queue Timeout Event
- Disabling Build Monitoring

**Configuring the Hanging Build Event**

The Hanging Build event is thrown when a Bamboo determines that a build has become unresponsive according to two criteria:

- 'Expected Build Time' — the 'Expected Build Time' = 'Build Time Multiplier' times the 'Average Build Time'
  - the 'Build Time Multiplier' is a user-defined setting.
• the 'Average Build Time' is calculated by Bamboo using an average of previous build times (in minutes).
• 'Log Quiet Time' — the length of time (in minutes) between log entries for a build.

The 'Expected Build Time' and 'Log Quiet Time' must both be exceeded for Bamboo to throw the build hung event.

On this page:

• Where is this event used?
• How do I configure the Hanging Build event?
• How do I disable the Hanging Build event?
• How do I change how often Bamboo checks for hung builds?

Where is this event used?

This event can currently be used to trigger notifications. Read more about Adding or Removing Notifications for a Plan.

How do I configure the Hanging Build event?

You can change the criteria governing when the Hanging Build event is thrown. You can also disable build monitoring altogether so that the Hanging Build event never occurs.

1 Please note that the Hanging Build criteria can be also be set for a specific plan, when specifying a plan's builder. Plan-level criteria will override the global criteria described on this page (including disabling the event).

To edit the Hanging Build event settings,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Build Monitoring' link in the left navigation column.
3. The 'Build Monitoring' screen will be displayed. Click the 'Edit' button.
4. The Default Build Monitoring Criteria will now be editable (see screenshot below). Update the values for 'Build Time Multiplier' and 'Log Quiet Time' as desired.
5. Click the 'Save' button to update your settings.

Screenshot: 'Editing Hanging Build event settings'

Build Monitoring

Default Build Monitoring Criteria

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans in the builder configuration of each plan. Bamboo determines that a build has hung, if both of the criteria below are exceeded.

• Expected Build Time - calculated as the average build time * build time multiplier
• Log Quiet Time - the length of time Bamboo goes without receiving any log messages for that build

Build Time Multiplier: 2.5
Log Quiet Time: 30

You can configure also default settings for detecting build waiting in build queue longer than specified time.

Build Queue Timeout - the length of time build waits in the queue before notification

Build Queue Timeout: 60

How do I disable the Hanging Build event?

You can disable the Hanging Build event by disabling build monitoring for your Bamboo installation. See Disabling Build Monitoring.

1 Please note, you cannot disable the Hanging Build event without disabling all build monitoring features for your Bamboo installation.

How do I change how often Bamboo checks for hung builds?
By default, Bamboo will check whether a Hanging Build event has been thrown every 60 seconds. You can change this by configuring the system property, `bamboo.buildHangingMonitor.checkInterval`. This property is specified in seconds.

Please read the [Configuring System Properties](#) page for instructions on how to configure the `bamboo.buildHangingMonitor.checkInterval` system property.

### Configuring the Build Queue Timeout Event

The Build Queue Timeout event is thrown when a build has been waiting in the build queue for longer than a specified period of time.

**On this page:**

- Where is this event used?
- How do I configure the Build Queue Timeout event?
- How do I disable the Build Queue Timeout event?
- How do I change how often Bamboo checks for build queue timeouts?

**Where is this event used?**

This event can currently be used to trigger notifications. Read more about [Adding or Removing Notifications for a Plan](#).

**How do I configure the Build Queue Timeout event?**

You can change the criteria governing when the Build Queue Timeout event is thrown. You can also [disable build monitoring](#) altogether so that the Build Queue Timeout event never occurs.

To edit the Build Queue Timeout event settings,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Build Monitoring' link in the left navigation column.
3. The 'Build Monitoring' screen will be displayed. Click the 'Edit' button.
4. The Default Build Monitoring Criteria will now be editable (see screenshot below). Update the value for 'Build Queue Timeout' (in minutes) as desired.
5. Click the 'Save' button to update your settings.

**Screenshot: 'Editing Build Queue Timeout event settings'**

### Build Monitoring

**Default Build Monitoring Criteria**

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans in the build configuration of each plan.

- Expected Build Time - calculated as the average build time * build time multiplier
- Log Quiet Time - the length of time Bamboo goes without receiving any log messages for that build

#### Build Time Multiplier:

2.5

Multiplier to be applied to the average build time (e.g. 2.5)

#### Log Quiet Time:

30

The length of time (in whole minutes) without logs being received (e.g. 10 minutes)

You can configure also default settings for detecting build waiting in build queue longer than specified time. Bamboo determines that build time has been exceeded if build waits in the queue longer specified timeout.

- Build Queue Timeout - the length of time build waits in the queue before notification

#### Build Queue Timeout:

60

The length of time (in whole minutes) before queue timeout would be detected (e.g. 60 minutes)

**How do I disable the Build Queue Timeout event?**

You can disable the Build Queue Timeout event by disabling build monitoring for your Bamboo installation. See [Disabling Build Monitoring](#).

Please note, you cannot disable the Build Queue Timeout event without disabling all build monitoring features for your Bamboo installation.
How do I change how often Bamboo checks for build queue timeouts?

By default, Bamboo will check whether a Build Queue Timeout event has been thrown every 60 seconds. You can change this by configuring the system property, `bamboo.buildQueueMonitor.checkInterval`. This property is specified in seconds.

Please read the Configuring System Properties page for instructions on how to configure the `bamboo.buildQueueMonitor.checkInterval` system property.

**Disabling Build Monitoring**

To disable the Build Monitoring,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Build Monitoring' link in the left navigation column.
3. The 'Build Monitoring' screen will be displayed (see screenshot below). Click the 'Disable' button to disable all build monitoring for your Bamboo installation.

⚠️ All build monitoring features will be disabled, including the build hanging event and build queue timeout notifications. It is not possible to disable separate build monitoring features separately.

**Screenshot: 'Disabling Build Monitoring'**

**Building Monitoring**

**Hanging Build Detection**

You can configure the default settings for detecting hanging builds on this page. Hanging build detection can also be disabled altogether on this page. You can override these settings for individual plans through the build configuration of each plan. Bamboo determines if a build has hung if both of the criteria below are exceeded.

- **Expected Build Time** - calculated as the average build time * build time multiplier
- **Log Quiet Time** - the length of time Bamboo goes without receiving any log messages for that build

**Build Queue Timeout Detection**

You can configure default settings for detecting build waiting in build queue longer than specified time. Bamboo determines if build time has been exceeded if build waits in the queue longer specified timeout.

- **Build Queue Timeout** - the length of time builds waits in the queue before notification

### Default Build Monitoring Criteria

<table>
<thead>
<tr>
<th>Build Time Multiplier</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Quiet Time</td>
<td>30</td>
</tr>
<tr>
<td>Build Queue Timeout</td>
<td>60</td>
</tr>
</tbody>
</table>

**Working with Elastic Bamboo**

*Elastic Bamboo* is a feature in Bamboo that allows you to utilise computing resources from the Amazon Elastic Compute Cloud (EC2) to run builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Builds can be run on these 'elastic agents' in the same way that builds are run on non-elastic agents.

The following pages and sub-pages contain important information about working with Elastic Bamboo:

- **About Elastic Bamboo** — Elastic Bamboo concepts.
- **Getting Started with Elastic Bamboo** — please see this page if you are setting up Elastic Bamboo for the first time. It contains instructions on enabling Elastic Bamboo for your Bamboo installation and running your first build.
- **Configuring Elastic Bamboo** — please see this page and the related sub-pages if you wish to change any of the initial settings after you have set up Elastic Bamboo. This includes instructions on how to use Amazon's Elastic Block Storage to persist build information for your builds on Elastic Bamboo.
- **Managing Elastic Bamboo** — please see this page and the related sub-pages for information on managing your elastic image, instances and agents.
- **Running Builds using Elastic Bamboo** — please see this page for helpful information on running your builds using Elastic Bamboo.
Elastic Bamboo is a feature in Bamboo that allows you to utilise computing resources from the Amazon Elastic Compute Cloud (EC2) to run builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Builds can be run on these 'elastic agents' in the same way that builds are run non-elastic agents.

A build can be run on an elastic agent, provided that the capabilities of the elastic agent meet the requirements of the build. Bamboo will assign the relevant build to an available elastic agent from the build queue automatically, in the same way that builds are assigned to non-elastic agents. The elastic agent must already be running for a build to be assigned to it.

An elastic agent is started by creating a new instance of an elastic image. Creating this new elastic instance automatically runs an elastic agent process in the instance. The agent inherits the capabilities of the image it was created from. Only one agent process can be run in an instance, although multiple instances can be created from the same image.

Once a build has completed the build results are made available, as per any other build. The elastic agent and instance will continue to run until they are manually shut down. Shutting down an elastic instance will terminate the agent, not take it offline. However, Bamboo will store historical information about the terminated elastic agent, such as the builds it has run.

Please note, Bamboo does not automatically start elastic agents to meet build demand.
An Amazon Web Services (AWS) account is required to use Elastic Bamboo. Elastic Bamboo Costs are charged by Amazon, separate to Bamboo licence costs, as Elastic Bamboo is powered by Amazon resources.

**Key Terms**

<table>
<thead>
<tr>
<th>Elastic Image</th>
<th>An elastic image is an Amazon Machine Image (AMI) that is stored in the Amazon Simple Storage Service (S3) for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances. You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Instance</td>
<td>An elastic instance is an instance of an elastic image. An elastic instance is created whenever an image is started. An image can be started multiple times, creating multiple instances. Each time an elastic instance is started, one elastic agent is created. Conceptually, an elastic instance can be thought of as a computer. Elastic agent processes are run on this computer and the elastic image is the boot hard disk. Please note however, that elastic instances are temporary and transient. Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) are not persisted when the instance is shut down. Any customisations to the instance itself will also be lost when the it is shut down.</td>
</tr>
<tr>
<td>Elastic Agent</td>
<td>An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an instance of an Elastic Bamboo elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.</td>
</tr>
</tbody>
</table>

**Setting Up Elastic Bamboo**

If you would like to set up Elastic Bamboo for your Bamboo installation, please read Getting Started with Elastic Bamboo. This document guides you through the initial configuration of Elastic Bamboo and running your first build.

**Elastic Bamboo Costs**

This page provides high level guidelines to Elastic Bamboo costs. As usage patterns vary from user to user, these guidelines are only intended to provide a picture of how Elastic Bamboo operates, not to make definitive pricing statements.

> The Bamboo pricing page on the Atlassian website details the costs for Elastic Bamboo. This page is intended to complement that information.

**Amazon EC2 Pricing Information**

You can use Elastic Bamboo to run remote agents on elastic instances in the Amazon Elastic Compute Cloud (EC2). If you choose to do this, you will be charged by Amazon for your EC2 compute usage. These charges will be billed to the AWS account that you provide.

> Please note, if you do not have an AWS account, you must register for one on the AWS registration page before you can enable Elastic Bamboo.

Full details on Amazon EC2 pricing is available on the Amazon EC2 pricing page. Please also note the following important information, which is relevant to EC2 usage by Elastic Bamboo:

- **You are responsible for all EC2 compute usage costs incurred on your AWS account.**
- Elastic Bamboo creates "High-CPU Medium" instances by default, however you can configure the EC2 instance type. Read the 'Elastic Instance Settings' section of Configuring Elastic Bamboo for instructions on how to change your default instance type. Please note the different costs for different instance types.
- You are responsible for creating and shutting down elastic instances to run agents in EC2.
- You can track your EC2 usage in near real-time on the AWS Account page.
- Your Elastic Bamboo compute usage will not be distinguishable from your non-Bamboo EC2 compute usage in your AWS billing.

**General Notes about EC2 Usage and Costs**

The following information is based on our usage of Elastic Bamboo at Atlassian. These points are intended to be guidelines to EC2 usage and
The bulk of EC2 costs from using Elastic Bamboo is for the uptime of EC2 instances. We strongly recommend that you shut down your instances when not in use.

- The costs for storing and moving data in and out of the EC2 will vary. However these costs are minimal (e.g. storing image) compared to instance uptime costs. Using the Amazon Elastic Block Store (EBS) with Elastic Bamboo can significantly reduce the data transfer (and associated costs) in and out of the EC2. Read more about configuring elastic instances to use EBS.
- The costs for using the Amazon Elastic Block Store (EBS) is minimal, relative to instance uptime costs.

Getting Started with Elastic Bamboo

Elastic Bamboo is a feature in Bamboo that allows you to utilise computing resources from the Amazon Elastic Compute Cloud (EC2) to run builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Builds can be run on these ‘elastic agents’ in the same way that builds are run non-elastic agents.

On this page:

- 1. Read Important Documents
- 2. Enable and Configure Elastic Bamboo
- 3. Start an Elastic Instance
- 4. Run a Build
- 5. Shut Down Elastic Instance
- Further Information

1. Read Important Documents

If you are using Elastic Bamboo for the first time, we highly recommend that you start by reading the following important documents:

- **About Elastic Bamboo** — This high-level overview explains the key concepts behind the Elastic Bamboo feature.
- **Elastic Bamboo Security** — We **strongly recommend** that you read this document to understand the security implications of enabling Elastic Bamboo. This includes important information on securing your version control system (VCS) for use with Elastic Bamboo.
- **Elastic Bamboo Costs** — Elastic Bamboo sources resources from the Amazon Elastic Compute Cloud (EC2) which are charged separately to your Bamboo license fee. We recommend that you read this document to understand how you will be charged for using Elastic Bamboo.

2. Enable and Configure Elastic Bamboo

Once you have understood the concepts, security implications and costs of Elastic Bamboo, you can enable and configure Elastic Bamboo for your Bamboo installation. You will also need to make your version control system (VCS) available to Amazon for Elastic Bamboo to work correctly.

2.1. Enabling Elastic Bamboo

To enable Elastic Bamboo,

1. Enable remote agent support in Bamboo — if you have disabled remote agent support, you must enable it before you can enable Elastic Bamboo. The **Disabling Remote Agents Support** documentation also contains instructions on how to enable remote agent support.
2. In Bamboo, click the 'Administration' link in the top navigation bar.
3. Click the 'Configuration' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Elastic Bamboo Configuration' screen will display.
4. Click the 'Enable' button to enable Elastic Bamboo for your Bamboo installation.

2.2. Configuring Elastic Bamboo

Before you can use Elastic Bamboo, you must configure it as detailed in the 'Configuring Elastic Bamboo' document. This is a simple three-step process:

1. Provide your Amazon Web Services account details
2. Configure your Elastic Bamboo global settings
3. Configure your elastic instance settings

- **Read the Configuring Elastic Bamboo document.**

2.3. Providing Access to your VCS
You need to make your version control system available to Amazon to run builds using Elastic Bamboo. This has security implications, particularly if your VCS is behind a firewall.

- Read the [Elastic Bamboo Security document for further instructions](#), if you have not read it already.

### 3. Start an Elastic Instance

Now that you have enabled and configured Elastic Bamboo for your Bamboo installation, you can try running a build with Elastic Bamboo. You can manually start an elastic instance via the Bamboo administration console. Starting an elastic instance will automatically start an elastic agent process on it.

- Read about [starting an elastic instance](#)

### 4. Run a Build

To run a build on your elastic agent, you must set up a build with requirements that can be met by your elastic agent's capabilities. Elastic agents inherit the capabilities of the image that they are started from. We recommend that you use the Bamboo default image to start with.

- Read about the [capabilities of the default image](#).

For the purposes of this guide, you should set up your build so that its requirements can only be met by the elastic agent's capabilities. This will ensure that the build runs on your elastic agent. If you cannot set up any build requirements that are uniquely met by the elastic agent's capabilities in your Bamboo installation, you can customise your elastic agent's capabilities to add a unique custom capability, e.g. `elastic=true`.

- Read about [configuring the capabilities of elastic agents](#).

Builds on elastic agents are run just like builds on any other agent. You will see the progress of your build on your dashboard and can view the build result when it has completed.

Tip: You can significantly reduce the costs and time taken to run a build by configuring Elastic Bamboo to use Amazon's Elastic Block Store (EBS).

### 5. Shut Down Elastic Instance

When your build has completed successfully, shut down your elastic instance. As described in [Elastic Bamboo Costs](#), the bulk of your Elastic Bamboo costs are from instance uptime. We strongly recommend that you shut down your elastic instances when not in use.

- Read about [shutting down an elastic instance](#).

Please note, that when you shut down an elastic instance, the agent process it is running is terminated. This means that elastic agents are not present on the ‘Agents’ page in Bamboo unless they are online. If you wish to view information about a terminated elastic agent, you can find the agent in the elastic agent usage history.

- Read about [viewing your elastic agent usage history](#).

Congratulations! You have successfully set up and run a build with Elastic Bamboo.

### Further Information

You may be interested in reading the following related topics below to help you manage and improve your Elastic Bamboo builds:

- [Configuring Elastic Instances to use EBS](#) — information on configuring Elastic Bamboo to use the Amazon Elastic Block Store (EBS) to improve build times.

### Configuring Elastic Bamboo

Elastic Bamboo is a feature in Bamboo that allows you to utilise computing resources from the Amazon Elastic Compute Cloud (EC2) to run builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Builds can be run on these ‘elastic agents’ in the same way that builds are run non-elastic agents.
Elastic Bamboo configuration settings include information that is required for Elastic Bamboo to work and parameters for optional features. These settings are divided into three logical groups:

On this page:

- **Amazon Web Services (AWS) Settings** — AWS account details for Elastic Bamboo.
- **Elastic Bamboo Global Settings** — global Elastic Bamboo settings for the Bamboo installation.
- **Elastic Instance Settings** — elastic instance settings for the Bamboo installation, including Elastic Block Store (EBS) settings.

All three groups of settings are located on the Elastic Bamboo Configuration screen (see screenshot). The instructions below describe how to update each of the groups of settings separately, but you can update all of the settings at once, if you wish.

Remote Agent Support

If you have disabled remote agent support, you must enable it before you can enable Elastic Bamboo. The Disabling Remote Agents Support documentation also contains instructions on how to enable remote agent support.

**Screenshot: ‘Elastic Bamboo Configuration’ (click to view full-size image)**

Amazon Web Services (AWS) Settings

Before you can use Elastic Bamboo in your Bamboo instance, you must enter your Amazon Web Services (AWS) account details into the Bamboo application. If you do not have an AWS account, you must register for one on the AWS registration page before you can enable Elastic Bamboo.

**Elastic Bamboo Costs**

Elastic Bamboo dynamically creates and runs remote agents in the Amazon Elastic Compute Cloud (EC2). Hence, if you choose to use Elastic Bamboo, you will be charged by Amazon for your EC2 compute usage (separate to your Bamboo license fee). These charges will be billed to the AWS account that you provide. Please read Elastic Bamboo Costs for more details.

To configure your Amazon Web Services (AWS) account details,

**Please note, if you change your AWS account details, Bamboo will stop all elastic agents that are currently running.**
Elastic Bamboo Global Settings

Elastic Bamboo provides you with a number of global configuration options to help you optimise EC2 usage for your Bamboo builds. These settings control how the Bamboo server operates and how it manages its elastic instances and agents.

To configure your Elastic Bamboo global settings,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Configuration' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Elastic Bamboo Configuration' screen will display.
3. Click the 'Edit' button to edit your Elastic Bamboo configuration.
4. Your elastic instance settings will be displayed under the 'Elastic Instance Settings' header. Update these settings as desired:
   - **Maximum Number of Elastic Instances** — this setting controls how many elastic agents can be running at one time. For example, you may wish to change this setting to a small value if you are concerned about EC2 compute costs, and you have a large number of concurrent builds that cannot be supported by your non-elastic agents.
   - **Automatically shut down elastic instance when elastic agent process ends** — this setting controls whether your elastic instances will automatically shut down after the elastic agents running on them end. If you have automated the starting and stopping of elastic agents (i.e. via the Bamboo Remote API), you can use this option to automatically shut down their instances as well.
   - **Shutdown Delay** — this setting will only display if you have checked the 'Automatically shut down elastic instance when elastic agent process ends' option. It controls how long an elastic instance will wait before shutting down, after its elastic agent process ends.
5. Click the 'Save' button to update your elastic instance settings.

Elastic Instance Settings

These settings control how elastic instances operate. This section includes settings that are used to configure elastic instances to work with the Amazon Elastic Block Store (EBS).

**Improving your build times with Amazon EBS**

Using EBS with your elastic instances can significantly reduce the amount of data transfer required to run a build, compared with starting a clean elastic instance. To find out more about this feature and how to set it up in Elastic Bamboo, read Configuring Elastic Instances to use EBS.

To configure your elastic instance settings,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Configuration' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Configure Elastic Bamboo' screen will display (see screenshot below).
3. Click the 'Edit' button to edit your account details.
4. The 'Elastic Bamboo Configuration' screen will display. Update your elastic instance settings as desired:
   - **Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified)** — Check this option to upload the AWS Account Private Key File and Account Certificate File to all new elastic instances started. This is mandatory if you wish to use EBS to store snapshot build information. However, you can also check this option if you are not using EBS (e.g. if you wish upload the AWS account identifiers in order to use Amazon's AWS command line tools).
   - **Account Private Key File** — this setting will only display if you have checked the 'Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified)' option. You must specify the location of this file (on your Bamboo server) to use the Amazon EBS with Elastic Bamboo.
   - **Account Certificate File** — this setting will only display if you have checked the 'Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified)' option. You must specify the location of this file (on your Bamboo server) to use the Amazon EBS with Elastic Bamboo.

   **How do I generate my AWS private key file and certificate file?**
   If you haven't downloaded an AWS private key file or certificate file to your Bamboo server yet, please see **Generating your AWS Private Key File and Certificate File** for instructions.

5. Click the 'Save' button to update your elastic instance settings.

### Generating your AWS Private Key File and Certificate File

The Amazon Web Services (AWS) private key file and certificate file work together to allow Elastic Bamboo to securely access AWS. It is different to the security mechanism provided by the AWS Secret Access Key and is required to enable certain features, such as EBS for elastic instances and the Amazon command line tools.

- **The certificate file** contains the public key associated with your AWS account. This file is kept by Amazon, (not on your Bamboo server).
- **The private key file** contains the private key that is used to authenticate requests to AWS. This file must be stored on your Bamboo server, if you are using EBS for elastic instances or the Amazon command line tools.
- The public key and private key from these files together form an **X.509 certificate**.

**On this page:**
- Generating the Files
- Downloading the Files
- Important Information

#### Generating the Files

The first time you use Elastic Bamboo, Bamboo will automatically generate the private key file and certificate file for you. The certificate file will be kept by Amazon (to inject into your elastic instances) and the private key file will be downloaded to your Bamboo server in your Bamboo Home directory. If you are setting up Elastic Bamboo on multiple Bamboo servers using the same AWS account, you can simply copy the private key file across from the original Bamboo server. You should not need to regenerate the private key file and certificate file unless your private key file is lost or corrupted.

If you do need to regenerate the private key file and certificate file, please follow the instructions in the Amazon X.509 Certificates documentation. The Amazon documentation also contains instructions on using your own certificate, if you wish.

#### Downloading the Files

Once the files are generated, you will be able to download them (see screenshot below). We recommend that you store the files in the Home directory of your Bamboo server.

**Screenshot: Downloading the generated AWS private key file and certificate file**
Important Information

Please take note of the following important information regarding your AWS private key file and certificate file:

- If you wish to use this security mechanism with multiple Bamboo installations using the same AWS account (e.g., you have configured your elastic instances on each installation to use EBS), you will need to copy the AWS private key file and certificate file to each Bamboo server.
- You can only download the AWS private key file at the time it is generated. If the private key file has already been generated for your AWS account, you will not be able to download it from AWS again (for security purposes). You will have to copy it from wherever it was previously downloaded to. Otherwise, you will have to generate a new private key file and certificate file to go with it.
- If you regenerate a new private key file and certificate file, any Bamboo servers using the old private key file and certificate file will no longer be able to access the Amazon EC2, as only one X.509 certificate can be associated with your AWS account.
- You can download the AWS certificate file as many times as you want. This file does not need to be regenerated.

Configuring Elastic Instances to use EBS

Amazon Elastic Block Store (EBS) provides persistent storage volumes that can be attached to EC2 instances. Elastic Bamboo can use the EBS to store snapshots of relatively static build information, such as checkouts of source code and Maven repository data. You can choose a snapshot to create EBS volumes from and associate it with an elastic image configuration. These volumes can then be attached to your elastic instances when they are started from their elastic image.
Why should I use Amazon EBS with Elastic Bamboo?
Elastic agents are effectively stateless. This means that any externally-retrieved resources such as Maven artifacts, cached application servers, etc, will be downloaded again each time an elastic agent is started. The elastic agent will also perform a full checkout the first time a plan is built on the agent. If your builds have external dependencies and/or you are not performing clean builds each time, you may find that Amazon EBS will significantly improve your build times.

Additionally, EBS can be used as an easy mechanism for customising elastic agents, rather than changing the configuration of the elastic images (see Creating a Custom Elastic Image) that the agents are started from. For example, you could upload files and scripts to your EBS volume that would install PostgreSQL databases on elastic agents when are started from the related elastic instance.

On this page:
- Creating your first EBS snapshot
- Configuring an Elastic Image to use an EBS snapshot
- Important EBS Directories and Files

Creating your first EBS snapshot

To create your first EBS snapshot,

1. Download Amazon Web Services (AWS) account identifiers to your Bamboo server — You will need to store the AWS private key file and certificate file on your Bamboo server to use Elastic Bamboo with EBS. If you haven't downloaded an AWS private key file or certificate file to your Bamboo server yet, please see Generating your AWS Private Key File and Certificate File for instructions.
2. Update your Bamboo configuration settings with the location of the AWS account identifier files you have downloaded. This will ensure that these files are uploaded to any new elastic instances started. See the Elastic Instance Settings section on the Configuring Elastic Bamboo for instructions (you will need to update the 'Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified)' checkbox and 'Account Private Key File' and 'Account Certificate File' fields described in this document).
4. Access your elastic instance via SSH (see Accessing an Elastic Instance for instructions).
5. Follow the steps below to create an EBS volume and attach it to the elastic instance (step a & b), upload content to the EBS volume (step c & d), and generate the snapshot (step e & f):
   - **All the scripts described below are bundled with Bamboo.**
     a. Run `createInitialVolume.sh <volume size>` — This script creates a EBS volume (where <volume size> is the size of the volume), attaches the volume and mounts it on the elastic instance. For example, `createInitialVolume.sh 100` will create a 100GB EBS volume and attach and mount it on the elastic instance.
     b. Run `rewarmEbsSnapshot.sh` — This script sets up the standard structure for Elastic Bamboo on the EBS volume. The directories and files for this standard volume structure are detailed in the Important EBS Directories and Files section below.
     c. **(optional)** Populate your EBS volume — Your EBS volume can now be populated with any files and scripts that you wish to make available to the elastic instances that use the EBS volume. For example, you may want to upload maven repository data, source code, scripts and files to install databases on your elastic agents, etc. You must upload your files to the `/mnt/bamboo-ebs` folder or its subfolders, if you want them to be included in the snapshot. We recommend that you read Populating your EBS volume for guidelines on how to populate your EBS volume effectively.
     d. Ensure all uploaded content has the owner `bamboo:bamboo` — You can set the owner of a file by executing the following command: `chown -R bamboo:bamboo <filename>`
     e. Execute the `killall java` command — This command kills all processes on the instance, such as agent processes, so that the volume can be unmounted to be snapshotted.
     f. Run `generateSnapshot.sh` — This script unmounts and detaches the volume, before creating a snapshot based on the volume. The time taken to create the snapshot will vary depending on the amount of content that you have uploaded to the EBS volume. The Snapshot ID for the snapshot will be available in the logs for the elastic instance. See Accessing an Elastic Instance for instructions on how to access the logs for your elastic instance.

Congratulations! You have created your first EBS snapshot.

Configuring an Elastic Image to use an EBS snapshot

Once you have set up an EBS snapshot, the final step is to add the snapshot details to an elastic image configuration, so that any instances
To configure Elastic Bamboo to use an EBS snapshot,

1. Determine the Snapshot ID of the EBS snapshot you have just created. The Snapshot ID should be recorded in the logs of the elastic instance you created it on. You can also view your EBS snapshots in the AWS Console by clicking the 'Snapshots' menu item.

2. Navigate to the Bamboo administration menu, i.e. click the 'Administration' link in the top navigation bar of the Bamboo application.

3. Click the 'Image Configurations' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Manage Elastic Image Configurations' page will display.

4. Click the 'Edit' link in the 'Operations' column of the elastic image configuration that you would like to add your EBS snapshot to. The 'Edit Elastic Image Configuration - <imagename>' screen will display (see screenshot below).

5. Check the 'Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances' checkbox. The 'EBS Snapshot ID' field will display.

6. Enter the Snapshot ID of your EBS snapshot in the 'EBS Snapshot ID' field.

7. Click the 'Save' button. A new EBS volume will be created from the specified snapshot and attached to any new elastic instances started from that image.

**Updating your EBS snapshot**

If you are currently using EBS with Elastic Bamboo and want to update your snapshot, follow the instructions below. These are similar to the instructions for creating a new EBS snapshot.

To update your EBS snapshot,

2. (optional) Run a build on the elastic agent of the instance to populate the attached EBS volume. We recommend that you read Populating your EBS volume for guidelines on how to populate your EBS volume effectively.
3. Access your elastic instance via SSH (see Accessing an Elastic Instance for instructions) and do the following:
   a. (optional) Upload any additional content to the attached EBS volume via Secure Copy (SCP). You must upload your files to the `/mnt/bamboo-ebs` folder or its subfolders, if you want them to be included in the snapshot.
   b. Execute `killall java` — This command kills all agent processes, so that nothing is using the mounted volume.
   c. Execute `jps -vl` — This command displays a list of all java processes running on your instance. There should be no java processes running.
   d. Run `generateSnapshot.sh` — This script unmounts and detaches the volume, before creating a snapshot based on the volume.
   e. Check the elastic instance logs for the Snapshot ID of the snapshot you just created. See Accessing an Elastic Instance for instructions on how to access the logs for your elastic instance.
   f. Update the new Snapshot ID in your Elastic Bamboo configuration, as described in Configuring an Elastic Image to use an EBS snapshot above.
Important EBS Directories and Files

By convention, Bamboo will attach an EBS device at /dev/sdh. This will be mounted at /mnt/bamboo-ebs. The contents of the standard structure are:

- **bin/customiseInstance.sh** - This script is run on startup of an elastic instance. We recommend that you do not customise this script, as it is overwritten when `rewarmEbsSnapshot.sh` is run.
- **bin/customise-extras.sh** - This script is run on startup of an elastic instance as the root (as opposed to being run as the Bamboo user). This script is safe to customise, as it will never be overwritten. You can customise this script to automate processes such as setting up your database, move files to custom locations on the instance, etc.
- **profile-extras.sh** - This script gets appended to the profile that is run under the Bamboo user (as opposed to being run as the root). It is useful for setting up environment variables.
- **bamboo-agent/bamboo-agent.cfg.xml** - This configuration file modifies the build working directory to point to build working directory on the EBS volume.
- **bamboo-agent/build-dir** - This is the build working directory.
- **maven/build.properties** - This properties file is copied to /home/bamboo on startup of an elastic instance. It points the Maven 1 default repository to /mnt/bamboo-ebs/maven/.maven
- **maven/settings.xml** - This configuration file is copied to /home/bamboo/.m2 on startup of an elastic instance. It points the Maven 2 default repository to /mnt/bamboo-ebs/maven/.m2/repository.
- **tmp-extras** - The contents of this directory is copied to /tmp on startup of an elastic instance.

Populating your EBS volume

This page is intended to complement the instructions for Configuring Elastic Instances to use EBS. It lists different methods of for populating your EBS volume, depending on the data you wish to have available in your snapshot.

On this page:

- Uploading Maven 2 repository data
- Uploading Ant repository data
- Setting up PostgreSQL on elastic agents
- Setting up Selenium on elastic agents

Uploading Maven 2 repository data

You can upload **Maven 2 repository data** to your EBS volume, so that it doesn't have to be downloaded every time an elastic agent (running on an instance that uses the EBS volume) is started.

To populate your EBS snapshot with your Maven repository data, we recommend that you upload it via SCP (see step 5c of the 'Creating your first EBS snapshot' section in Configuring Elastic Instances to use EBS). In most cases, you will have a modified settings.xml file if you are using Maven 2. This means that you will need to upload this file and Maven repository data to your EBS volume, rather than populating your volume by running a build.

Uploading Ant repository data

You can upload **Ant repository data** to your EBS volume, so that it doesn't have to be downloaded every time an elastic agent (running on an instance that uses the EBS volume) is started.

To populate your EBS snapshot with your Ant repository data, we recommend that you run a build on an elastic agent with a blank EBS volume attached to the elastic instance (see step 2 of the 'Updating your EBS snapshot' section in Configuring Elastic Instances to use EBS). This is a faster and more reliable method of populating your volume, if you are using Ant.

Setting up PostgreSQL on elastic agents

You can upload scripts to your EBS volume so that any agents started on elastic instances that use the EBS volume will have **PostgreSQL** automatically installed.

To set up the automatic installation of PostgreSQL on elastic agents, you will need to create the following script:

**setupPostgreSQL.sh**
What does this script do?
This script uses the package management tools provided by Fedora to install and configure PostgreSQL on the agent when it is started.

1. Uses yum to install the PostgreSQL server packages. Details on the yum tool can be found in the Fedora Software Management Guide.
2. Initialises the PostgreSQL server environment by creating the database directories and default config files.
3. Creates a new pg_hba.conf file which trusts all local connections and all connections coming from localhost.
4. Starts PostgreSQL.

You then need to update the `customise-extras.sh` file on your EBS volume (see Important EBS Directories and Files) to invoke this script.

Finally, you need to add a custom capability (e.g. postgres-true) to the elastic agents with PostgreSQL installed. You can do this by updating the elastic image configuration that the agents inherit their capabilities from. Read Configuring Elastic Agent Capabilities for detailed instructions.

Setting up Selenium on elastic agents

You can upload scripts to your EBS volume so that any agents started on elastic instances that use the EBS volume will be able to run Selenium tests.

To set up elastic agents to support Selenium test, you will need to create the following script:

setupSelenium.sh

```bash
#!/bin/sh
yum install -y vnc-server xorg-x11-server-Xvfb xterm xorg-x11-server-utils twm xorg-x11-fonts-* yum install compat-libstdc++-33
if [ ! -d /home/bamboo/.vnc ]; then
    mkdir /home/bamboo/.vnc
fi
cp /mnt/bamboo-ebs/bin/vncpasswd /home/bamboo/.vnc/passwd
cp /mnt/bamboo-ebs/bin/vncxstartup /home/bamboo/.vnc/xstartup
chown -R bamboo.bamboo /home/bamboo/.vnc
chmod 600 /home/bamboo/.vnc/passwd
chmod u+x /home/bamboo/.vnc/xstartup
rm -rf /opt/firefox
zcat /mnt/bamboo-ebs/bin/firefox-2.0.0.20.tar.gz | tar -xvf -C /opt
grep -q LD_LIBRARY_PATH /home/bamboo/.bashrc
if [ "$?" == "1" ]; then
echo "export LD_LIBRARY_PATH=/opt/firefox" >> /home/bamboo/.bashrc
fi
```
What does this script do?
This script uses the package management tools provided by Fedora to install Mozilla's Firefox and enough of X to get a VNC (Virtual Network Computing) server running.

1. Uses yum to install the following packages. Details on the yum tool can be found in the Fedora Software Management Guide:
   - vnc-server — the vnc server used by the selenium test server.
   - xorg-x11-server-Xvfb xterm xorg-x11-server-utils twm xorg-x11-fonts— these packages cover the xorg dependencies to get Firefox to run.
2. The script then copies some prepared VNC authentication files into the bamboo home directory and sets their permissions accordingly. These files are:
   - vncpasswd — this is the password file used by the VNC server (copied to /home/bamboo/.vnc/passwd)
   - vncxstartup — this is the script executed by the VNC server when a connection is made (copied to /home/bamboo/.vnc/xstartup)
3. The last step of this script is to manually install Firefox into /opt/firefox (we manually install Firefox because the package that would be installed by the Fedora 8 package management appears to be outdated).
   - The tar is extracted to the appropriate directory.
   - The .bashrc file is customised to include the Firefox directory when searching for libraries. This is so Firefox will be able to find its libraries.

You then need to update the customise-extras.sh file on your EBS volume (see Important EBS Directories and Files) to invoke this script.

Finally, you need to add a custom capability (e.g. selenium=true) to the elastic agents with PostgreSQL installed. You can do this by updating the elastic image configuration that the agents inherit their capabilities from. Read Configuring Elastic Agent Capabilities for detailed instructions.

Managing Elastic Bamboo

The following pages and the related sub-pages contain information on managing your elastic image, instances and agents.

- Managing your Elastic Images — please see this page and the related sub-pages for detailed information about Elastic Bamboo images in Bamboo. This includes instructions on how to view and customise the capabilities of your Elastic Bamboo images.
- Managing your Elastic Instances — please see this page and the related sub-pages for detailed information about Elastic Bamboo instances in Bamboo. This includes instructions on how to view, start, stop and access an elastic instance.
- Managing your Elastic Agents — please see this page and the related sub-pages for detailed information about Elastic Bamboo remote agent instances in Bamboo. This includes instructions on how to view and disable an elastic instance.

Managing your Elastic Images

An elastic image is an Amazon Machine Image (AMI) that is stored in the Amazon Simple Storage Service (S3) for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances.

You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.

- To view an elastic image, including the image properties, capabilities and the plans that an image can build, see Viewing an Elastic Image.
- To customise the capabilities of an elastic image, see Configuring Elastic Agent Capabilities.
- To create your own custom elastic image, see Creating a Custom Elastic Image.
- To associate an elastic image with your Bamboo installation, see Managing your Elastic Image Configurations.

Viewing an Elastic Image

An elastic image is an Amazon Machine Image (AMI) that is stored in the Amazon Simple Storage Service (S3) for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances.

You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.
An image is similar to an agent, hence the 'Image' page closely resembles the 'Agent' page. A number of functions available for agents are also available for images.

- **Viewing an elastic image's capabilities** — your image has capabilities, similar to how agents have capabilities. Read more about viewing an agent's capabilities.
- **Viewing the plans that an image can build** — you can also view the plans that an image is capable of building (via the elastic agent created from the image), similar to how you view the plans that an agent is capable of viewing. Read more about viewing the plans that an agent can build and determining which agents can build which plans.

To view an image,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Image Configurations' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Image Configurations' page will display.
3. Click the name of the image that you want to view, e.g. “Default”. The elastic image configuration will display (see screenshot below).

   - **Name** — this is the name of the image.
   - **AMI ID** — this is the Amazon Machine Image identifier that uniquely identifies the image.
   - **EBS Snapshot ID** — this is the ID of the EBS Snapshot that you have associated with this image. See Configuring Elastic Instances to use EBS and Managing your Elastic Image Configurations for more information on how to use EBS with Elastic Bamboo.
   - **'Instance Type'** — the instance type of new instances started from this image. Read more about Amazon instance types.
   - **'Availability Zone Preference'** — new instances started from this image will be run in the Amazon availability zone nominated in this field.
   - **'Active Instances'** — number of currently active instances that were started from this image.

*Screenshot: Elastic Bamboo Image Configuration*
## Elastic Image Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Elastic Image Configuration shipped with Bamboo</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMI ID</td>
<td>ami-31f9e56</td>
</tr>
<tr>
<td>EBS Snapshot ID</td>
<td>snap-s3e86c5</td>
</tr>
<tr>
<td>Instance Type</td>
<td>High-CPU Medium</td>
</tr>
<tr>
<td>Availability Zone Preference</td>
<td>Default (chosen by EC2)</td>
</tr>
<tr>
<td>Active Instances</td>
<td>0</td>
</tr>
</tbody>
</table>

## Elastic Image Capabilities

### Custom

Custom capabilities are key-value pairs that define specific characteristics of an agent (e.g., `operating-system=WindowsXP`, `test.build=true`). For an agent to be able to build a plan, both the `Key` and `Value` must match the plan’s requirements.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo.functionalTest</td>
<td>true</td>
<td>View</td>
</tr>
</tbody>
</table>

### Builder

Builder capabilities define the builders which are available to your build plans.

<table>
<thead>
<tr>
<th>Label</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (Ant)</td>
<td>/opt/apache-ant-1.7.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 1 (Maven)</td>
<td>/opt/maven-1.0.2</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.0)</td>
<td>/opt/maven-2.0</td>
<td>View</td>
</tr>
</tbody>
</table>

### JDK

JDK capabilities define the JVMs which are available to your build plans.

<table>
<thead>
<tr>
<th>Label</th>
<th>Java Home</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.5</td>
<td>/opt/jdk-5</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6</td>
<td>/opt/jdk-6</td>
<td>View</td>
</tr>
</tbody>
</table>

## Add Capability

- **Capability Type:** Custom
- **Key:**
- **Value:**

[Add | Cancel]
Creating a Custom Elastic Image

An elastic image is an Amazon Machine Image (AMI) that is stored in the Amazon Simple Storage Service (S3) for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances.

You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.

At a high level, the process for creating a custom elastic image consists of taking one of the existing AMIs (Amazon Machine Images) available on Amazon EC2, starting an instance of the AMI, customising the instance and then creating an image from the customised instance. This image can then be used as an elastic image in your Bamboo installation.

Please note, this is not a trivial process. You may wish to consider customising the elastic agents started from your existing image instead, by using Amazon's Elastic Block Store (EBS), as described in Configuring Elastic Instances to use EBS. This is a much simpler option. If you are having problems, please don't hesitate to contact us for further help.

Important Information

Please note the following important information:

- These instructions are written for Unix/Linux operating systems.
- A number of the EC2 commands in the steps below can be completed using the AWS console rather than command line tools (e.g. registering an image). However, we recommend that you follow the steps below unless you are an experienced AWS user.

On this page:

- 1. Requirements
- 2. Selecting an Existing AMI
- 3. Starting an Instance
- 4. Accessing your Instance
- 5. Customising your Instance
  - 5.1 Installing Bamboo Prerequisite Packages
  - 5.2 Adding User Customisations to your Instance
  - 5.3 Deploying Bamboo onto your Instance
  - 5.4 Instance Configuration
- 6. Creating an Image of your Customised Instance
- 7. Next Steps
- 8. Need More Help?

1. Requirements

Before you begin, you need to ensure that you have set up the following:

- Amazon Web Services (AWS) account with EC2 — if you are already using Elastic Bamboo, you should already have an AWS account with EC2 set up. If not, please read Getting Started with Elastic Bamboo.
- Amazon EC2 API Tools — you must install the EC2 API tools on your local machine, otherwise you will not be able to start and access your AMI instance. You can install the EC2 API tools by executing the following commands:

```
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools-1.3-30349.zip
unzip ec2-api-tools-1.3-30349.zip
```

- Environment Variables — you must set up the following environment variables on your local machine before creating a custom elastic image:
  - EC2_HOME — set this to the path to the installed EC2 API Tools
  - EC2_CERT — set this to the path to the certificate assigned to EC2 account
  - EC2_PRIVATE_KEY — set this to the path to the private key assigned to your AWS account
- Registered Key Pair — you need a registered EC2 key pair, which consists of a private key file and certificate file, to use the EC2 API tools with your AMI instance. If you have previously generated and registered an EC2 key pair (e.g. to use the EC2 API tools), you can re-use it. If you need to generate a new key pair, you can use the following command to do so:

```
$EC2_HOME/bin/ec2_add_keypair <key_pair_name>
```

The content of the private key will display in the command-line output on your console. Save this content in a file, starting with the line

```
```
"-----BEGIN RSA PRIVATE KEY-----" and ending with the line "-----END RSA PRIVATE KEY-----". This private key file will be used to access your AMI instance. Set up the appropriate permissions on the private key file by executing the following command.

```
chmod 600 <private_key_file>
```

2. Selecting an Existing AMI

We strongly recommend that you select an existing Linux/UNIX AMI to customise, rather than starting with a blank AMI. You can get the list of available AMIs by executing the following command:

```
$EC2_HOME/bin/ec2-describe-images -a
```

⚠️ Please check whether you want to launch 32-bit or 64-bit instances from your custom elastic image before selecting an existing AMI. Elastic Bamboo creates "High-CPU Medium" instances (32-bit) by default.

We recommend the following existing Linux/UNIX AMIs for customisation:

- **ami-e55bbd8c** — this AMI is the default image provided by Atlassian. It is a Fedora 8 image updated and prepared for Bamboo, i.e. you will not have to install any Bamboo prerequisites.
- **ami-2b5fba42** — this is the base Fedora 8 image provided by Amazon. It does not have any Bamboo prerequisites installed.
- **ami-0d729464** — this is an Ubuntu 9.04 image provided by Alestic. It does not have any Bamboo prerequisites installed. You can find more Ubuntu images on the Alestic site. Please note, these images are not maintained by Atlassian.

3. Starting an Instance

After you have selected an existing AMI to customise, the next step is to start an instance of the AMI. Use the `ec2-run-instances` command to start your instance, as follows:

```
$EC2_HOME/bin/ec2-run-instances <image_name> -k <key_pair_name>
```

where `<image_name>` is the name of the AMI selected in the previous step and `<key_pair_name>` is the name of the registered key pair generated in ‘1. Requirements’ (the public certificate of this key will be injected into your instance).

For example, if you wanted to start an instance of image `ami-e55bbd8c` using key pair `my-keypair`, you would run the following command:

```
$EC2_HOME/bin/ec2-run-instances ami-e55bbd8c -k my-keypair
```

This command would produce the following command-line output:

```
INSTANCE i-25b86743 ami-e55bbd8c running my-keypair
```

*i-25b86743* is the name of your new instance, in the above example. Note this down, as you will need the instance name to access your instance in the next step.

⚠️ Don't forget to shut down unused instances

Please note, once you start an instance, you will be billed by Amazon for instance uptime. If you decide to abandon the setup of a custom elastic image after this step, please ensure that you shut down your instance via the AWS console.

4. Accessing your Instance

Once your instance is running, you will need to obtain the address of the instance so you can access it. To do this, use the following command:
For example, if you wanted to find the address of instance `i-25b86743`, you would enter:

```
ec2-describe-instances i-25b86743
```

This command would produce the following command-line output similar to this:

```
RESERVATION r-790f7210 121852097033 default
INSTANCES i-25b86743 ami-e55bbd8c ec2-174-129-94-241.compute-1.amazonaws.com
domU-12-31-29-04-38-87 compute-1.internal running elasticbamboo 0 ml.small
2009-06-24T12:36:20+0000 us-east-1c aki-a71cf9ce ari-a51cf9cc
```

The address of your instance in the above example is `ec2-174-129-94-241.compute-1.amazonaws.com`

You can then use this address to access the instance via SSH. See Accessing an Elastic Instance for instructions. If you are using the example command text from that document, you will need to adjust it as follows:

- replace `/opt/bamboo/home/xml-data/configuration/elasticbamboo.pk` in the example command text with the private key file you generated in '1. Requirements'.
- replace `ec2-68-111-185-197.compute-1.amazonaws.com` in the example command text with the address of your instance.

5. Customising your Instance

Customising your instance is the most complicated part of creating a custom elastic image. You need to install the packages that are prerequisites for Bamboo onto your instance (if you didn’t choose the Elastic Bamboo default image as your base AMI), add your customisations, deploy Bamboo onto your instance and set up an EC2 environment on your instance.

5.1 Installing Bamboo Prerequisite Packages

If you selected `ami-e55bbd8c` as your base AMI in '2. Selecting an Existing AMI', you can skip this step and go to '5.2 Adding Customisations' as image `ami-e55bbd8c` has been pre-configured for Bamboo. If you have selected a different AMI, you will need to install the following packages onto your instance using the commands shown below:

1. Amazon EC2 API tools

```
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools-1.3-30349.zip
unzip ec2-api-tools-1.3-30349.zip
mv ec2-api-tools-1.3-30349 /usr/local/ec2/ec2-api-tools
```

2. Java 6 and Java 5

```
# get Java 6
wget "http://download.java.net/dlj/binaries/jdk-6u11-dlj-linux-i586.bin"
mkdir unbundle-jdk-6
cd unbundle-jdk-6
sh ../jdk-6u11-dlj-linux-i586.bin --accept-license
cd ..

# get Java 5
wget "http://download.java.net/dlj/binaries/jdk-5.0u17-dlj-linux-i586.bin"
mkdir unbundle-jdk-5
cd unbundle-jdk-5
sh ../jdk-5.0u17-dlj-linux-i586.bin --accept-license
cd ..

# Build the JDKs
wget -O construct.sh "https://jdk-distros.dev.java.net/source/browse/*checkout*
jdk-distros/trunk/utils/construct.sh?content-type=text%2Fplain&rev=148" --no-check-certificate
sh construct.sh unbundle-jdk-6 /opt/jdk-6 /opt/jre-6
sh construct.sh unbundle-jdk-5 /opt/jdk-5 /opt/jre-5
```

3. Ant
5.3 Deploying Bamboo onto your Instance

Once you have installed the Bamboo pre-requisites on your instance and added your customisations, you can deploy Bamboo onto your instance. Before you deploy anything however, you need to create a 'bamboo' user on your instance by running the following command:

```
useradd bamboo
```

After you have created the 'bamboo' user, you need to copy the Bamboo artifacts onto your instance by using the following commands:

```bash
cd /mnt
scp -i <private_key_file> * root@<instance_address>:/mnt
```

where `<private_key_file>` is the private key file generated in '4. Accessing your Instance', `<instance_address>` is the address of your instance from '4. Accessing your Instance' and `<BAMBOO_STANDALONE_DIRECTORY>` is your Bamboo Installation directory.

Navigate to the /mnt directory on your instance and install the artifacts, as described below:

1. **Maven 1.0.2** (patched version)

   ```
tar xzC /opt -f maven-1.0.2.tar.gz
   ``

2. **Bamboo Agent binaries**

   ```
tar xzC /opt -f atlassian-bamboo-agent-elastic-assembly-2.3.tar.gz
   ln -s bamboo-elastic-agent-2.3 /opt/bamboo-elastic-agent
   ```

Please note however, you cannot customise the operating system of a running instance. If you want to create an instance with a customised operating system (e.g. Ubuntu), you will need to select an AMI with that operating system installed (as previously described in '2. Selecting an Existing AMI').

5.2 Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,

1. Log into your elastic instance (as previously described in '4. Accessing your Instance').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run '```
sudo apt-get install tomcat6
```
configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.

3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '6. Creating an Image of your Customised Instance'). Any instances started from this image will have all of your user customisations automatically installed.

5. Maven 1.1

A patched version of Maven 1.0 is also required. This will be installed as part of step '5.3 Deploying Bamboo onto your Instance'.

I. Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,

1. Log into your elastic instance (as previously described in '4. Accessing your Instance').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run '```
sudo apt-get install tomcat6
```
configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.

3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '6. Creating an Image of your Customised Instance'). Any instances started from this image will have all of your user customisations automatically installed.

4. Maven 2

A patched version of Maven 2 is also required. This will be installed as part of step '5.3 Deploying Bamboo onto your Instance'.

I. Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,

1. Log into your elastic instance (as previously described in '4. Accessing your Instance').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run '```
sudo apt-get install tomcat6
```
configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.

3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '6. Creating an Image of your Customised Instance'). Any instances started from this image will have all of your user customisations automatically installed.

5. Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,

1. Log into your elastic instance (as previously described in '4. Accessing your Instance').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run '```
sudo apt-get install tomcat6
```
configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.

3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '6. Creating an Image of your Customised Instance'). Any instances started from this image will have all of your user customisations automatically installed.

4. Maven 1.0

A patched version of Maven 1.0 is also required. This will be installed as part of step '5.3 Deploying Bamboo onto your Instance'.

I. Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,

1. Log into your elastic instance (as previously described in '4. Accessing your Instance').
2. Once you have logged into your elastic instance, you can treat it as a standalone machine and install anything you want. For example, if you want to install Tomcat on an Ubuntu instance you would run '```
sudo apt-get install tomcat6
```
configure it, ensure that your startup scripts are in place, etc, just as you would when installing Tomcat on a standalone machine.

3. Everything that you install will be saved in snapshot image created at the end of these instructions (see '6. Creating an Image of your Customised Instance'). Any instances started from this image will have all of your user customisations automatically installed.

4. Maven 1.0

A patched version of Maven 1.0 is also required. This will be installed as part of step '5.3 Deploying Bamboo onto your Instance'.

I. Adding User Customisations to your Instance

Adding your own customisations is quite a simple process, once you have made it this far.

To add user customisations to your instance,
atlassian-bamboo-agent-elastic-assembly-2.3.tar.gz is available in webapp sub-directory of the Bamboo installation directory.

3. Bamboo Agent default capabilities definition file

mv bamboo-capabilities.properties /home/bamboo/

5.4 Instance Configuration

At this stage, you should have a customised instance with Bamboo deployed onto it. The last step in creating a customised instance is to set up an EC2 environment on your instance. Carry out the following steps to set this up:

1. Transfer Amazon private key file and certificate to your instance
   Transfer the key files to your instance by running these commands on your local machine:

   ```bash
   scp -i <private_key_file> $EC2_PRIVATE_KEY root@<instance_address>:/mnt
   scp -i <private_key_file> $EC2_CERT root@<instance_address>:/mnt
   ```

   where `<private_key_file>` is the private key file from your local machine created in step 'Registered Key Pair' of 1. Requirements and the `<instance_address>` is the address of your instance from '4. Accessing your Instance'

2. Set up EC2_HOME and JAVA_HOME environment variables
   Set up these environment variables by running the following commands on your instance:

   ```bash
   export EC2_HOME=/usr/local/ec2/ec2-api-tools
   export EC2_PRIVATE_KEY=/mnt/<ec2_private_key_file>
   export EC2_CERT=/mnt/<ec2_certificate_file>
   export JAVA_HOME=/opt/jdk-5
   ```

3. Set appropriate permissions for the bamboo user directory
   Run the following command on your instance to set permissions on the bamboo user directory:

   ```bash
   chown -R bamboo:bamboo /home/bamboo/
   ```

4. Configure path variables
   The `profile.sh` file should have been transferred to your instance in previous steps. This file contains Bamboo path configuration settings, as seen below:

   ```bash
   export JAVA_HOME=/opt/jdk-5
   export M2_HOME=/opt/maven-2.0
   export MAVEN_HOME=/opt/maven-1.0.2
   export ANT_HOME=/opt/apache-ant-1.7.1
   export EC2_HOME=/usr/local/ec2/ec2-api-tools
   export EC2_PRIVATE_KEY=/root/pk.pem
   export EC2_CERT=/root/cert.pem
   export PATH=/opt/bamboo-elastic-agent/bin:$EC2_HOME/bin:$JAVA_HOME/bin:$M2_HOME/bin:$MAVEN_HOME/bin:$ANT_
   ```

   If all of the tools on this page were installed in recommended locations, no changes are required. Otherwise, you can update the file as required.

   Once `profile.sh` has been updated, you need to copy it to the `/etc/profile.d` directory by running the following command on your instance in the `mnt` directory:

   ```bash
   mv profile.sh /etc/profile.d/bamboo.sh
   ```

5. Configure automatic startup of the Bamboo agent
   You will need to configure your instance to start up the Bamboo agent automatically when the instance is started. You can do this by appending the `rc.local` file to the one that already exists on your instance, by running the following command on your instance in the `mnt` directory:

   ```bash
   cat rc.local >> /etc/rc.d/rc.local
   ```

6. Final settings and cleanup
   Finally, create a Bamboo welcome screen and clean up keys on your instance by running the following command:
6. Creating an Image of your Customised Instance

The final step is to create an image from your customised instance. To do this, you will require the following information:

- Amazon Account Number
- Access Key ID
- Secret Access Key
- Amazon S3 bucket name that will be used to store image (if you don't have access to Amazon S3, you can sign up on this page.)

You can create an image of your customised instance by using the `ec2-bundle-vol` command, as follows:

```
/usr/local/bin/ec2-bundle-vol -c /mnt/$EC2_CERT -k /mnt/$EC2_PRIVATE_KEY -u <amazon_account_number>
-p <elastic_image_name> --batch --debug
```

where `<ec2_private_key_file>` is the private key file on your instance, `<amazon_account_number>` is your Amazon Account Number and the `<elastic_image_name>` is the name that you want to assign to your custom image (e.g. 'CustomImage1')

Once the image is created, you need to upload it to Amazon S3 by running the command below:

```
/usr/local/bin/ec2-upload-bundle -b <s3_bucket_name> -m /tmp/<elastic_image_name>.manifest.xml -a <access_key_id> -s <secret_access_key> --retry --debug
```

where `<s3_bucket_name>`, `<access_key_id>` and `<secret_access_key>` are the Amazon S3 bucket name, Access Key ID and Secret Access Key described previously, and `<elastic_image_name>` is the name that you want to assign to your custom image (e.g. 'CustomImage1')

You will then need to register your image with Amazon EC2 by using the `ec2-register` command:

```
$EC2_HOME/bin/ec2-register <s3_bucket_name>/<elastic_image_name>.manifest.xml
```

where `<s3_bucket_name>` is the Amazon S3 bucket name described previously and `<elastic_image_name>` is the name that you want to assign to your custom image (e.g. 'CustomImage1')

The output of this command will show the AMI ID of your custom image.

Congratulations, you have successfully set up a custom elastic image!

7. Next Steps

Now that you have created a custom elastic image, there are two more steps that you will need to complete before you can use it.

First, you will need to associate your custom elastic image with your Bamboo installation by creating an Elastic Image Configuration. Please note the AMI ID of your new custom image and read Managing your Elastic Image Configurations for further instructions.

Secondly, you will need to configure the capabilities of the elastic agents that will run on instances started from your image. This is done by adding the appropriate builder, JDK, Perforce and custom capabilities to your elastic image configuration, so that it reflects what your custom elastic image actually can do. For example, if you have created a custom elastic image with JDK 1.6 and Maven 2 installed, you will need to add capabilities for JDK 1.6 and Maven 2 to the elastic image configuration. Read Configuring Elastic Agent Capabilities for further instructions.

8. Need More Help?
If you need more help, there are a number of resources that you can take advantage of:

- **AWS Support Center** — if you are having problems with any of your Amazon services, not specifically related to Bamboo, you can obtain basic support from the AWS Support Center. Note, you will need to sign up for Premium Support to get access to web/phone support.
- **AWS Resource Center** — the AWS Resource Center has links to online documentation, code samples and tools for AWS services.
- **Atlassian Support** — if you are having problems with any of the Bamboo-related steps of this process, please raise a ticket in the Bamboo project on our support system and our support team will respond as soon as possible.
- **Bamboo Developer Forums** — please feel free to discuss any useful tips or issues regarding this process in the Bamboo Developer Forums.

### Managing your Elastic Image Configurations

An **elastic image** is an Amazon Machine Image (AMI) that is stored in the **Amazon Simple Storage Service (S3)** for use with the **Elastic Bamboo feature**. An elastic image is used to create **elastic instances**, which in turn create **elastic agents**. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances.

You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.

**Associating a custom elastic image** with your Bamboo installation allows you to start elastic instances with a different set of inherited capabilities than those inherited from the default image. For example, you may wish to associate an elastic image with Ubuntu installed with your Bamboo installation, so that you can run Ubuntu-related tests on the instances started from that image.

Once you have associated a custom elastic image with Bamboo, the settings for your elastic image are stored as an elastic image configuration.

**To associate a custom image with Bamboo,**

1. Click the ‘Administration’ link in the top navigation bar in Bamboo.
2. Click the ‘Image Configurations’ link in the left navigation column under the ‘Elastic Bamboo’ sub-header. The ‘Manage Elastic Image Configurations’ screen will display.
3. Enter the details of your custom elastic image in the panel under the ‘Create Elastic Image Configuration’ header:
   - **Name** — Enter the name of your custom elastic image. If you created your own custom image, you should have named it in step 6 of the 'Creating a Custom Elastic Image' instructions. You can also view the image name via the AWS console.
   - **Description** *(optional)* — Enter an optional description of your image for display in Bamboo only.
   - **AMI ID** — If you created your own custom image, you should obtained the AMI ID as an output of step 6 of the 'Creating a Custom Elastic Image' instructions. You can also view the AMI IDs for images via the AWS console.
   - **Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances** *(optional)* — Check this option if you want the elastic instances started from this image to use EBS. Read more about Configuring Elastic Instances to use EBS.
   - **EBS Snapshot ID** *(displays if ‘Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances’ is checked)* — If you have checked the ‘Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances’ option, you must specify the EBS Snapshot ID of the EBS volume that you wish to attach to new instances.
   - **Instance Type** *(default is “High-CPU Medium”)* — Choose the instance type for new instances started from this image. Amazon offers a number of instance types that provide different computing capacity. Read more about Amazon EC2 instance types.
   - **Availability Zone** *(default is “Default (chosen by EC2)”)* — Choose the availability zone to start your new instances from this image in (e.g. if you wish to use Elastic Bamboo with reserved instances). We recommend that you select “Default (chosen by EC2)” to allow Amazon to select the best zone for your instance. Read more about Amazon EC2 availability zones.
Managing your Elastic Instances

An elastic instance is an instance of an elastic image. An elastic instance is created whenever an image is started. An image can be started multiple times, creating multiple instances. Each time an elastic instance is started, one elastic agent is created.

Conceptually, an elastic instance can be thought of as a computer. Elastic agent processes are run on this computer and the elastic image is the boot hard disk. Please note however, that elastic instances are temporary and transient. Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) are not persisted when the instance is shut down. Any customisations to the instance itself will also be lost when the it is shut down.

- To view a running elastic instance, see Viewing an Elastic Instance.
- To access your elastic instance via a client, see Accessing an Elastic Instance.
- To start one or more elastic instances, see Starting an Elastic Instance.
- To shut down one or more elastic instances, see Shutting Down an Elastic Instance.
- To configure your Elastic Bamboo settings for elastic instances, see the Elastic Instance Settings section in the Configuring Elastic Bamboo document.

Viewing an Elastic Instance

An elastic instance is an instance of an elastic image. An elastic instance is created whenever an image is started. An image can be started multiple times, creating multiple instances. Each time an elastic instance is started, one elastic agent is created.

Conceptually, an elastic instance can be thought of as a computer. Elastic agent processes are run on this computer and the elastic image is the boot hard disk. Please note however, that elastic instances are temporary and transient. Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) are not persisted when the instance is shut down. Any customisations to the instance itself will also be lost when the it is shut down.

To view an elastic instance,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the name of the instance that you want to view, e.g. 'i-05ff716c'. The information for the instance will display (see screenshot below).

**Information**

- **Current status** — this is the status of the elastic instance. Valid values include, 'Pending' (instance starting up), 'Running' and 'Shutting down'.
- **Public DNS** — this is the public DNS address of the elastic instance. The IP address of the elastic instance is displayed under this field.
- **Start Time** — this is the start time of the instance, based on the Amazon EC2 timezone (US Eastern Time for Elastic Bamboo). Please note, start time is the time when you send the request to start an instance, not the time when the instance progresses to 'Running' status. Up time of the instance (including the time taken for the instance to start up) is shown in brackets after the start time.
- **Elastic Agent** — this is the elastic agent process currently running on your elastic instance. Currently Elastic Bamboo only supports one elastic agent per elastic image. Click the 'Elastic Agent on <instance_name>' link to view the elastic agent. If the agent is running a plan, the plan name will be shown in brackets after the elastic agent name.
- **Current Availability Zone** — the availability zone that your elastic instance is running in. Read more about Amazon EC2 availability zones. This field will also show you what your availability zone preference is, in brackets after the current availability zone. For instructions on how to selecting the availability zone for your instances, please see Managing your Elastic Image Configurations.
- **Attached Volumes** — if you have configured your elastic instances to use EBS, the IDs of the attached EBS volumes will be displayed in this table.

**Configuration**

- **Configuration** — this is the name of the elastic image configuration that was used to create this elastic instance. Click the name to configure the elastic image.
- **AMI ID** — this is the ID of the elastic image (i.e. Amazon Machine Image) that the elastic instance was created from (as part of the elastic image configuration).
- **EBS Snapshot ID** — if you have configured your elastic instances to use EBS, the ID of the EBS snapshot that was used to create the EBS volumes attached to your instance will be displayed in this field.
- **Instance Type** — the instance type of your instance is displayed in this field.

**SSH Access**

- Please see Accessing an Elastic Instance for information on using this function.

**Accessing Logs**

- Please see Accessing an Elastic Instance for information on using this function.

---

**Amazon Web Services Management Console**

You can also view information about your elastic instances on the AWS Management Console. Please note, we strongly recommend that you use the console for viewing instance information only. You may experience errors if you attempt to manage your instances outside of Bamboo.

_Screenshot: View Instance_
Accessing an Elastic Instance

It is possible to connect directly to a running elastic instance to access logs or upload files. Access is available via SSH (secure shell) and file transfer is enabled via SCP (secure copy).

⚠️ Please note, you can only access elastic instances that are running. You may need to configure the automatic termination of elastic instances.

To access your elastic instance via SSH,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the name of the instance that you want to access, e.g. 'i-05ff716c'. The information for the instance will display.
4. Copy the command text listed under the 'SSH Access' section. It will be similar to the following example command text:
   ```
   ssh -i /opt/bamboo/home/xml-data/configuration/elasticbamboo.pk
   root@ec2-68-111-185-197.compute-1.amazonaws.com
   ```
5. Execute the text in your terminal and you will have full SSH access to the Elastic Instance.

   You can also download the private key via the link in the 'SSH Access' section to access your elastic instance via SSH. Click the 'here' link in the following text on screen to download the key: "You can also download the SSH private key file from here and use the private key to access the EC2 instance."

   **Permission issues for SSH access**
   If you are experiencing permission issues when attempting to access your elastic instance via SSH, you may need to modify permissions on your Elastic Bamboo private key file. See this FAQ for further details.

To access your logs via SCP,

   You can also use SCP to upload files to your elastic instance.

```
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the name of the instance that you want to access, e.g. 'i-05ff716c'. The information for the instance will display.
4. Copy the command text listed under the 'Accessing Logs' section. It will be similar to the following example command text:
   ```
   scp -i /opt/bamboo/home/xml-data/configuration/elasticbamboo.pk
   root@ec2-68-111-185-197.compute-1.amazonaws.com:/home/bamboo/bamboo-elastic-agent.out ./
   ```
5. Execute the text in your terminal to download the logs from your elastic instance.
```

**Starting an Elastic Instance**

An elastic agent process runs in an elastic instance and will automatically start when an instance is started. If you want to run a build on an elastic agent, you can start an elastic instance for the agent to run in. The elastic agent will inherit the capabilities of the image that the instance is started from.

**Limitations on the number of elastic instances**

An elastic agent is counted as a remote agent for licensing purposes. Hence, if starting an elastic instance (and hence an elastic agent) causes you to exceed the total number of remote agents allowed under your license, you will not be able to start the instance.

To start an elastic instance(s),
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the 'Start New Elastic Instances' link. The 'Start New Elastic Instances' screen will display (see screenshot below).
   - Enter the number of new elastic instances you would like to start in the 'Number of instances' field.
   - Select the elastic image configuration that you would like to start your instance from in the 'Elastic Image Configuration Name' dropdown.

   ![Screenshot: Starting elastic instances (click to view full-size image)](image)

4. Click the 'Submit' button. The 'Manage Elastic Instances' page will display, showing the startup of your new instances, as follows:
   a. A note will display stating that the elastic instances (and corresponding agents) are starting.

      ```
      1 new elastic instance is pending. New instances and corresponding elastic agents may take a few minutes to start up.
      ```

   b. Your elastic instances will then display with a status of 'Pending' while they start up. This generally takes a few minutes.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2204914b</td>
<td>Pending</td>
<td>8 seconds</td>
<td>View</td>
</tr>
</tbody>
</table>

   c. Once your elastic instances have started up, they will progress to 'Running' status. An elastic agent process will then start up for each instance. They will display a status of 'Pending' while they start.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2204914b</td>
<td>Running</td>
<td>1 minute</td>
<td>View</td>
</tr>
<tr>
<td>Elastic on i-2204914b</td>
<td>Pending</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   d. Once the elastic agents have started, they will display a status of 'Online'.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2204914b</td>
<td>Running</td>
<td>10 minutes</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-2204914b</td>
<td>Online</td>
<td></td>
<td>Disable</td>
</tr>
</tbody>
</table>

Scheduling your Elastic Instances

You can schedule the startup and shutdown of elastic instances in Bamboo. For example, you may wish to shut down all elastic instances on weekends or start up additional instances to help cope with builds during regular busy periods.

On this page:
- Managing your Elastic Instance Schedules
- Adding a New Elastic Instance Schedule

Managing your Elastic Instance Schedules

To manage your elastic instance schedules,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instance Schedule' link in the left navigation column under the 'Elastic Bamboo' subheading. The 'View Elastic Instance Schedules' screen will display (see screenshot below).
   - Adding a new elastic instance schedule — click the 'Add Elastic Instance Schedule' link to create a new schedule from scratch or click the 'Copy' link next to an existing schedule to create a new schedule using the existing schedule as a template. See the Adding a New Elastic Instance Schedule section below for further instructions.
   - Editing an elastic instance schedule — click the 'Edit' link next to an existing schedule to edit it.
   - Enabling an elastic instance schedule — click the 'Enable All' link to enable all schedules or click the 'Enable' link next to any disabled schedules to enable them individually.
   - Disabling an elastic instance schedule — click the 'Disable All' link to disable all schedules or click the 'Disable' link next to any enabled schedules to disable them individually.
   - Deleting an elastic instance schedule — click the 'Delete' link next to an existing schedule to delete the schedule.

You can also view the configuration for the elastic image that the instances will be created from, by clicking the image configuration name (e.g. 'Default') in the table of schedules.

**Screenshot: Viewing Elastic Instance Schedules**

### View Elastic Instance Schedules

Configure when to start up or shut down elastic instances of a particular elastic image.

<table>
<thead>
<tr>
<th>Next Scheduled Run</th>
<th>Cron Expression</th>
<th>Image Config</th>
<th>Active Instances</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 Jul 2009, 11:00 PM (in 3 days)</td>
<td>Each FRI at 11:00 pm</td>
<td>Default</td>
<td>0</td>
<td>Edit</td>
</tr>
<tr>
<td>Trigger disabled</td>
<td>Daily at 8:00 pm</td>
<td>Default</td>
<td>0</td>
<td>Edit</td>
</tr>
</tbody>
</table>

**Adding a New Elastic Instance Schedule**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instance Schedule' link in the left navigation column under the 'Elastic Bamboo' subheading. The 'View Elastic Instance Schedules' screen will display (see screenshot below).
3. Click the 'Add Elastic Instance Schedule' link to create a new schedule from scratch or click the 'Copy' link next to an existing schedule to create a new schedule using the existing schedule as a template. The 'Add Elastic Instance Schedule' screen will display.
   - 'Enabled' (checked by default) — uncheck this if you do not want this schedule to be enabled when you create it.
   - 'Trigger on' — Select whether you want this schedule to trigger on Bamboo startup or when a set of cron conditions are met. If you select 'A cron schedule' the following field will display:
     - 'Cron Expression' (mandatory if 'A cron schedule' selected) — specify the cron conditions that must be met for this schedule to trigger. cron is a time-based scheduler with a unique syntax. For information on constructing cron expressions, please see this FAQ.
     - 'On trigger Bamboo should' — select whether Bamboo should adjust the number of active elastic instances or shut down all instances when it is triggered. If you select 'Adjust number of active instances', the following fields will display (all of the fields below are mandatory if 'Adjust number of active instances' is selected):
       - 'Image Config' — specify which image the elastic instances should be started from. The elastic agents running on the instances will inherit the capabilities from the image.
       - 'Logical operator' — specify the constraint on the number active instances which is specified in the 'Active Instances' field.
       - 'Active Instances' — number of active instances. See the logical operator field above for details on how this value is used.

**Screenshot: Adding an Elastic Instance Schedule**
**Shutting Down an Elastic Instance**

We recommend that you shut down any elastic instances that are not being used. Amazon EC2 charge for the period of time that you have an instance running, so you can minimise your costs simply by shutting down instances with inactive agents. You should also shut down your elastic instances if you are going to restart your Bamboo server, otherwise you will orphan them from your Bamboo server.

If you have set up automated procedures via the Bamboo Remote API to terminate agents (e.g. cron jobs), you can also configure Elastic Bamboo to automatically shut down instances after the agent processes terminate.

Please ensure that the agent on an elastic instance is not running a build, before shutting down the instance. Any builds running on the agent **will be abandoned** when you shut down the elastic instance.

On this page:

- Shutting down an elastic instance
- Shutting down all elastic instances
- Configuring automatic shutdown of instances after agent termination
- Shutting down elastic instances via the Amazon Web Services (AWS) Console

**Shutting down an elastic instance**

*To shut down an elastic instance,*
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the 'Shut Down' link in the 'Operations' column, next to the instance that you wish to shut down. The 'Shut Down Instance' screen will display.

**Shutdown Instance**

⚠️ You are attempting to shutdown elastic instance i-fcf97795. All builds running on this instance will be abandoned.

[Confirm] [Cancel]

4. Click the 'Confirm' button to shut down the elastic instance. The 'Manage Elastic Instances' screen will display again. The elastic instance that you have shut down will display a 'Shutting down' status for a few minutes, before it shuts down and disappears from this screen.

Shutting down all elastic instances

**To shut down all elastic instances,**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Instances' link in the left navigation column. The 'Manage Elastic Instances' screen will display.
3. Click the 'Shut Down All Instances' link. The 'Shut Down All Instances' screen will display.

**Shutdown All Instances**

⚠️ You are attempting to shutdown ALL elastic instances. All builds running on these instances will be abandoned.

[Confirm] [Cancel]

4. Click the 'Confirm' button to shut down all elastic instances. The 'Manage Elastic Instances' screen will display again. The elastic instances will display 'Shutting down' status' for a few minutes, before they shut down and disappear from this screen.

Configuring automatic shutdown of instances after agent termination

**To configure Elastic Bamboo to automatically shut down instances when agents are terminated,**

Please refer to Configuring Elastic Bamboo and follow the instructions for setting the 'Automatically shut down elastic instance when elastic agent process ends' option in the 'Elastic Bamboo Global Settings' section.

Shutting down elastic instances via the Amazon Web Services (AWS) Console

We strongly recommend that you manage your instances via the Elastic Bamboo user interface. However, if you have orphaned your elastic instances from your Bamboo server (e.g. restarted your Bamboo server without shutting down your elastic instances), you may need to shut your elastic instances down directly in AWS.

Please refer to How do I shut down my elastic instances if I have restarted my Bamboo server? for further details.

Managing your Elastic Agents

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an instance of an Elastic Bamboo elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

- To view your elastic agents, see Viewing your Elastic Agents.
To view elastic agents that have terminated, see Viewing your Elastic Agent Usage History.
To disable an elastic agent, see Disabling an Elastic Agent.

Viewing your Elastic Agents

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an instance of an Elastic Bamboo elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

To view your elastic agents,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The agents for your Bamboo instance will be displayed (see screenshot below). Any elastic agents that are running will be listed in the 'Remote Agents' section. The elastic agent name will be prefixed with 'Elastic Agent', e.g. 'Elastic Agent on i-2204914b'.

A note about elastic agent status
An elastic agent will always have an 'Online' status, (i.e. 'Online' or 'Online (Disabled)'). If you disable an elastic agent, the elastic instance will remain online. However, if you shut down the elastic instance, then the elastic agents process is killed and will not appear in the remote agents list. Hence, an elastic agent will never have an 'Offline' status.

Agents

An agent is a service that executes Bamboo builds. You can use this page to view, add and delete agents. You can also use this matrix to determine which agents can execute which build plans.

**Local Agents**

Local agents run on the Bamboo server.

There are currently no local agents configured on this Bamboo instance.

Add Local Agent

**Remote Agents**

Remote agents run on computers other than the Bamboo server.

There are currently 2 remote agents online (2 elastic). A maximum of 25 agents are supported by your license.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Agent on i-3674e05f</td>
<td>Building - CONFMAIN-254</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-8d74e06e4</td>
<td>Building - CONF-87</td>
<td>View</td>
</tr>
<tr>
<td>atlassian(agent 1)</td>
<td>Offline</td>
<td>View</td>
</tr>
<tr>
<td>atlassian(agent 2)</td>
<td>Offline</td>
<td>View</td>
</tr>
<tr>
<td>atlassian(agent 3)</td>
<td>Offline</td>
<td>View</td>
</tr>
<tr>
<td>atlassian(agent 4)</td>
<td>Offline</td>
<td>View</td>
</tr>
</tbody>
</table>
Viewing your Elastic Agent Usage History

When you shut down an elastic instance, the agent process for that instance is killed. As such, the elastic agent will not appear in an offline status, but will be removed altogether from your available agents in Bamboo.

However, information about these elastic agents is recorded in Bamboo and can be viewed on the 'Elastic Agent History' page.

To view the history of an elastic instance that has been shut down,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agent History' link in the left navigation column. The 'Elastic Agent Usage History' screen will display.
3. To view the usage history of the elastic agent, click the agent name or the 'View' link next to the agent. The 'Elastic Agent History' page for the elastic agent will display (see screenshot). This page will show the following information:
   - 'Elastic Instance' — the elastic instance that the elastic agent ran in.
   - 'Last Start Time' — the last time that the elastic agent was started. This is based on the Bamboo server time.
   - 'Last Stop Time' — the last time that the elastic instance was stopped. This is based on the Bamboo server time.
   - 'Up Time' — the total time that the elastic agent was online.
   - 'Build History' — this table lists the builds run by the elastic agent and information about the build, such as the status, duration, test results, etc. You can view the full build results by clicking the build number.

Screenshot: Elastic Agent History

Elastic Agent History > Elastic Agent on i-54ff753d (elastic)

Elastic agent on instance i-54ff753d, image: ami-bbf512d2

You can view historical information for this elastic agent in this page. This agent is offline and will never come back alive.

<table>
<thead>
<tr>
<th>Elastic Instance</th>
<th>Instance i-54ff753d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Start Time</td>
<td>12/02/09 16:02</td>
</tr>
<tr>
<td>Last Shutdown Time</td>
<td>18/02/09 17:32</td>
</tr>
<tr>
<td>Up Time</td>
<td>8730 minutes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Build Number</th>
<th>Reason</th>
<th>Completed</th>
<th>Duration</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>CONFFUNC-UNIT16-1600</td>
<td>Dependant of CONFFUNC-MAIN-4662</td>
<td>1 day ago (03:03 PM, Wed, 18 Feb)</td>
<td>2 minutes</td>
<td>2282 passed</td>
</tr>
<tr>
<td>✓</td>
<td>CONFFUNC-INTRUNKJDK16-1535</td>
<td>Dependant of CONFFUNC-MAIN-4662</td>
<td>1 day ago (03:05 PM, Wed, 18 Feb)</td>
<td>1 minute</td>
<td>384 passed</td>
</tr>
<tr>
<td>✓</td>
<td>CONFFUNC-MAIN-4661</td>
<td>Scheduled build</td>
<td>1 day ago (02:31 PM, Wed, 18 Feb)</td>
<td>30 minutes</td>
<td>1184 passed</td>
</tr>
</tbody>
</table>

Configuring Elastic Agent Capabilities

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an instance of an Elastic Bamboo elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

Hence, you can customise the capabilities of your elastic agents by configuring the capabilities on the relevant elastic image.
You may want to configure the capabilities on your elastic image to force your builds to run on particular elastic agents (e.g. running slow acceptance tests on your most powerful elastic agents). You may also need to configure the capabilities on any custom elastic images that you have created and/or associated with your Bamboo installation.

To configure the capabilities on an elastic image,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Image Configurations' link in the left navigation column under the 'Elastic Bamboo' subheading. The 'Manage Elastic Image Configurations' screen will display.
3. Click the 'View' link for the image that you want to configure the capabilities for. The configuration screen will display, showing the capabilities of the image.
4. You can add new capabilities to the image via the 'Add Capability' panel at the bottom of the screen. Adding a new capability to an image is very similar to adding capabilities to non-elastic agents. Please see the following pages for further information:
   - Configuring a new Builder
   - Configuring a new JDK
   - Configuring a new Perforce Capability
   - Configuring a new Custom Capability
5. You can also edit, rename and delete a capability from an elastic image, similar to how you would edit, rename and delete a capability from a non-elastic agent. Please see the following pages for further information:
   - Editing a Capability
   - Renaming a Capability
   - Deleting a Capability
6. You can also view the agents and elastic image configurations with a particular capability and the plans with the related requirement by clicking the 'View' link for the capability.
7. Any changes that you have made to elastic agent capabilities will only be reflected in new agents started after the changes were made. You will need to restart any existing agents, if you want them to pick up your changes.

**Configuring capabilities for the default image**

Adding a builder, JDK or Perforce capability to the image does not install the actual builders, JDKs or Perforce modules on the image. Please take particular note of this, if you are adding capabilities to a custom image.
Disabling an Elastic Agent

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2). An elastic agent process runs in an instance of an Elastic Bamboo elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

If you would like to stop an elastic agent, you can disable it in Bamboo. This will abandon any build it is running and prevent it from running any further builds.

Please note, disabling an elastic agent will not shut down the elastic instance it is running on (i.e. you will still be charged for the instance uptime). You can permanently stop an elastic agent and instance by shutting down the elastic instance.

The Bamboo server also “supervises” your elastic agents. If the Bamboo server detects that an elastic agent is offline, it will automatically terminate the elastic instance.

To disable an elastic agent,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column.
3. The agents for your Bamboo instance will be displayed. Locate your elastic agent in the 'Remote Agents' section. The elastic agent name will be prefixed with 'Elastic Agent', e.g. 'Elastic Agent on i-2776e24e'.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Agent</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance i-2776e24e</td>
<td>Running</td>
<td>5 minutes</td>
<td>View</td>
<td>Shut Down</td>
</tr>
<tr>
<td>Elastic Agent on i-2776e24e</td>
<td>Online</td>
<td></td>
<td></td>
<td>Disable</td>
</tr>
</tbody>
</table>

4. Click the 'Disable' link in the 'Operations' column for the elastic agent. The elastic agent will display with a status of 'Online (Disabled)'.
   - If you wish to re-enable the elastic agent, click the 'Enable' link in the 'Operations' column for the elastic agent.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Agent</th>
<th>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance i-2776e24e</td>
<td>Running</td>
<td>6 minutes</td>
<td>View</td>
<td>Shut Down</td>
</tr>
<tr>
<td>Elastic Agent on i-2776e24e</td>
<td>Online (Disabled)</td>
<td></td>
<td></td>
<td>Enable</td>
</tr>
</tbody>
</table>

---

**Running Builds using Elastic Bamboo**

This page contains answers to common questions about running builds using Elastic Bamboo. If you are using Elastic Bamboo for the first time, we highly recommend that you read Getting Started with Elastic Bamboo for instructions on setting up Elastic Bamboo and running your first build.

On this page:

- What builds can I run on Elastic Bamboo?
- How do I run a build on an elastic agent?
- How do I automatically start/stop elastic agents for builds?
- How do I know whether my build was run on an elastic agent?
- How do I customise the capabilities of my elastic agents?
- How much does it cost to run a build?
- What is EBS and how does it affect my builds?

**What builds can I run on Elastic Bamboo?**

You can run any of your builds on elastic agents, providing that the capabilities of the elastic agent meets the requirements of the build. An elastic agent inherits the capabilities of the image it was created from. Hence, you can see which build plans can be run by elastic agents, by checking which plan’s requirements match your elastic image’s capabilities.

- You view your elastic image and the builds that it meets the requirements of, on the Agents and Plans matrix.

**How do I run a build on an elastic agent?**

An elastic agent operates in a similar fashion to a non-elastic agent. That is, Bamboo will determine if there is a build in the build queue that can be built by the elastic agent, and assigns the build to that agent if the build requirements are met.

If you do not have any free elastic agents running, you will need to start one to make it available to builds. An elastic agent is automatically started when you start an elastic instance. Read Starting an Elastic Instance for instructions on how to start an elastic instance.

We strongly recommend that you shut down your elastic instance (and hence your elastic agent), when it is not in use. Minimising redundant instance uptime will help optimise your Elastic Bamboo running costs. Read Shutting Down an Elastic Instance for instructions on how to shut down an elastic instance.

**How do I automatically start/stop elastic agents for builds?**
Bamboo does not automatically start or stop elastic agents based on demand from the build queue. If there is a build in the build queue that can be built using an elastic agent, Bamboo will not automatically start up one. Bamboo also will not automatically stop an elastic agent when a build is completed.

However, Elastic Bamboo can be controlled via the Bamboo Remote API. For example, you could implement cron jobs to intelligently start and stop elastic instances, so that elastic agents are available at key times for your builds.

**How do I know whether my build was run on an elastic agent?**

The name of the image and elastic agent that ran a build can be viewed as part of the build result. Please see the [Viewing a Build Result](#) page for more information.

**How do I customise the capabilities of my elastic agents?**

You may want to customise the capabilities of your elastic agents to suit your build strategy. For example, if you want to force certain builds to only run on elastic agents, you can add a custom capability of `elastic=true` to your elastic agents and add the same requirement to your builds.

To customise the capabilities for your elastic agents, you need to customise the capabilities of the image that they are created from. Read [Configuring Elastic Agent Capabilities](#) for instructions.

**How much does it cost to run a build?**

As Elastic Bamboo usage varies from customer to customer, we cannot provide a definitive cost estimate for running a build using Elastic Bamboo. We do provide high level guidelines for Elastic Bamboo costs, based on our own experience of using Elastic Bamboo at Atlassian, on the [Elastic Bamboo Costs](#) page.

You can significantly reduce the costs and time taken to run a build by configuring Elastic Bamboo to use Amazon's Elastic Block Store (EBS).

**What is EBS and how does it affect my builds?**

The [Amazon Elastic Block Store (EBS)](#) provides persistent storage volumes that can be attached to EC2 instances. Elastic Bamboo can use the EBS to store snapshots of relatively static build information, such as checkouts of source code and Maven repository data. You can choose a snapshot to create EBS volumes from. These volumes can then be attached to your elastic instances when they start up.

**Disabling Elastic Bamboo**

If you do not want to run builds in the Amazon EC2 anymore, you can disable Elastic Bamboo for your Bamboo installation. Your AWS account details will be preserved when you disable Elastic Bamboo, so you can just enable it if you want to start using it again.

⚠️ Please ensure that you do not require your elastic agents before disabling Elastic Bamboo, as they will be stopped immediately.

To disable Elastic Bamboo,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Configuration' link in the left navigation column under the 'Elastic Bamboo' sub-header. The 'Configure Elastic Bamboo' screen will display (see screenshot below).

**Elastic Bamboo Configuration**

Elastic Bamboo is a feature that allows Bamboo to dynamically source computing resources from the Amazon Elastic Compute Cloud (EC2). Please see the documentation on understanding Elastic Bamboo and working with Elastic Bamboo for more information on what Elastic Bamboo is and how you can use it.

You can configure your Amazon Web Services (AWS) account details on this page, as well as global settings for Elastic Bamboo.

### Configuration Details

<table>
<thead>
<tr>
<th>AWS Access Key ID</th>
<th>182GHMDP0J811P42Z2AT</th>
</tr>
</thead>
</table>

#### Elastic Bamboo Global Settings

- **Maximum Number of Concurrent Elastic Instances:** 1
- **Automatic Shutdown:** EC2 instances will be automatically terminated 100 seconds after the agent process ends

#### Elastic Instance Settings

- **Default Instance Type:** Small

  - **EBS volumes** are not automatically attached to new elastic instances.
  - AWS account identifier files are not automatically uploaded to new elastic instances.

3. Click the 'Disable' button. Elastic Bamboo will disabled and a confirmation message will display.

### Configuration Details

Amazon Elastic Compute Cloud (EC2) support is currently disabled.

Enable

---

**Configuring Email and Instant Messaging Notifications**

- Adding or Removing Notifications for a Plan
- Configuring Bamboo to send SMTP Email
- Configuring Bamboo to use Instant Messaging (IM)
  - Configuring Bamboo to use Google Talk for Instant Messaging
- Modifying Notification Templates
  - Freemarker and Notification Templates
Adding or Removing Notifications for a Plan

Notifications in Bamboo are triggered by a range of events for a plan, including build completion, build outcomes and comments being posted against build results. You can configure whether notifications are sent for a particular event and who they are sent to. Users can choose whether to receive their notifications via email, IM, both or neither.

You must have the ‘Edit’ permission for a plan, to add or remove notifications for it.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Completed Builds</td>
<td>Bamboo will send a notification whenever a build finishes for this plan, regardless of the build result. This trigger is recommended for any plans for which it is critical that people are always informed about the latest build activity. Many organisations start with this trigger, then change it later as they get more confidence in the continuous build process.</td>
</tr>
</tbody>
</table>
| Failed Builds And First Successful | This trigger is generally suitable for the majority of plans. Bamboo will send a notification whenever:  
  * a build fails for this plan.  
  * the plan is ‘fixed’ (that is, the plan’s latest build is successful and the previous build failed). |
| After X Failed Builds | This trigger enables you to specify the ‘Number Of Failed Builds’ after which Bamboo will send a notification. This is a useful way of limiting the number of notifications, if you are concerned about people receiving too many. You can also use this event to escalate build problems, e.g. notify a manager if a build fails five times. |
| Build Hung | Bamboo will send a notification whenever it believes that a build has hung, according to the hung build criteria (read more about configuring your Building Hanging Detection settings). You can use this event to ensure that the right people are informed, if a build becomes unresponsive. |
| Build Commented | Bamboo will send a notification whenever a comment is posted against a build result. The email notification will contain all comments against the build, whereas the IM notification will only contain the comment that triggered the notification. This feature can help improve collaboration between team members. Please note that you will not receive notifications for any comments that you post. |
| Build Error | Bamboo will send a notification whenever an error occurs with the build process (i.e. the activities that Bamboo performs to run a build). This event is not related to failures of the actual build itself (see the ‘Failed Builds And First Successful’ event above). For example, a notification will be sent if Bamboo encounters an error when connecting to the repository, or detecting changes. |
| Build Queued Without Capable Agents | Bamboo will send a notification whenever a build is queued and there are no agents capable of building it. You can use this event to ensure that people are notified when changes to the agents adversely affect your builds. |
| Build Queue Timeout | Bamboo will send a notification whenever a build has been waiting in a build queue for longer than the build queue timeout criteria (read more about configuring your Build Queue Timeout settings). You can use this event to ensure that the right people are informed, if a build is stuck in a queue for too long. |

For each plan, you can specify different recipients for each Notification Trigger. Note also that recipients need not be people with Bamboo user accounts.

Before you begin
You need to configure Bamboo's SMTP email and/or instant messaging capabilities before Bamboo can send notifications.

To add notifications for a plan,
If you are creating a new plan, start at step 5.

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the plan in the list and click this icon: 📚
4. The ‘Configuration’ tab will be displayed. Click the ‘Notifications’ sub-tab.
5. The ‘Notifications’ that you currently have set up will be displayed (see screenshot below). Add a new notification in the ‘Add Build Notification’ section as follows:
   - Select the ‘Event’:
     - ‘All Completed Builds’
     - ‘Failed Builds and First Successful’
     - ‘After X Failed Builds’
     - ‘Build Commented’
     - ‘Build Hung’
     - ‘Build Queued Without Capable Agents’
     - ‘Build Queue Timeout’
   - Specify the ‘Recipient’:
     - ‘Committers’ — A committer is the Bamboo user(s) who committed code to a particular build (i.e. someone who committed code after the previous build was checked out by Bamboo).
     - ‘Watchers’ — A plan’s watchers are the Bamboo users who have marked this plan as one of their favourites.
     - ‘Group’ — Type the name of the appropriate Bamboo group(s).
     - ‘User’ — Type the username of the appropriate Bamboo user; or click the following icon to select from a list of users: 🗿
     - ‘Email Address’ — This is useful if you need to send email notifications to a person who is not a Bamboo user. Type the appropriate email address. Please note, that if you specify the email address of an existing Bamboo user, the user will receive notifications, even if they have elected not to receive notifications in their user preferences.
     - ‘Instant Messaging Address’ — This is useful if you need to send IM notifications to a person who is not a Bamboo user. Type the appropriate IM address. Note that if you specify the IM address of an existing Bamboo user, the user will receive notifications, even if they have elected not to receive notifications in their user preferences.
   - Click the ‘Add’ button.
   - Repeat steps 5 and 6 until you have added all the build notifications that you wish to enable for this plan.
   - Click the ‘Done’ button if you are editing an existing plan; or if you are creating a new plan, click the ‘Next’ button and go to Specifying a Plan’s Post Actions.

Screenshot: ‘Build Notifications’
To remove notifications for a plan,

1. Click 'Home' to go to the Dashboard.
2. Click the 'All Plans' tab.
3. Locate the plan in the list and click this icon:
4. The 'Configuration' tab will be displayed. Click the 'Notifications' tab.
5. Locate the notifications that you wish to disable, click the corresponding 'Remove' link in the 'Operations' column for each notification.
6. Click the 'Done' button.

Modifying Notification Templates

On this page:

- Modifying a Notification Template
- Working with Freemarker
- Configuring Notifications Content via System Properties

Modifying a Notification Template

If you want to customise the layout and content of your Bamboo notifications, you can customise the templates for each of the notification types (i.e. HTML email, text email, instant message) and events (e.g. Build Commented). The notification templates are written in Freemarker.

To modify a notification template,

1. Locate the default notification templates in your Bamboo distribution in `/WEB-INF/classes/notification-templates/`
2. Copy the notification template that you wish to modify into the templates/notification-templates folder of your Bamboo home directory, e.g. `/HOME/templates/notification-templates`
   - The filename will have formatted as: `<event name><notification type>.ftl`, e.g. `AfterXFailedHTMLEmail.ftl`
3. Modify the copied template, as desired. Please see the section on Working with Freemarker below for tips on updating templates.
4. Save your changes to the template. Your updated template will be used in the next notification that the template applies to. You do not have to restart your Bamboo server.

   Please note, Bamboo does not validate notification templates. If you have incorrectly formatted the markup text in the template, Bamboo will still use the template to send out notifications. If this happens, your users may receive notifications with unreadable or missing information, as well as error messages. Errors will also be posted to your logs.

Working with Freemarker

The Bamboo notification templates are written in Freemarker. The Freemarker engine allows for dynamic content generation based on the Freemarker markup tags that are used in templates. This document does not describe the Freemarker language in detail. Please see the Freemarker Online Manual for full information on using this markup language.

Generating content via Freemarker involves merging a template (*.ftl file) with a context map. You can access any data in the context map from within the template using the Freemarker markup. For the notifications we have provided a specific subset of Bamboo objects that you can access. For example,

```freemarker
[if buildSummary.successful]
${buildSummary.buildResultKey} was successful.
```

If you had a successful Bamboo build with build result, BAM-1234-1, the above markup would return the following text in your notification: BAM-1234-1 was successful.

You can find more information on working with Freemarker, including Bamboo objects available from Freemarker templates, tips on writing Freemarker templates and examples in the Freemarker and Notification Templates document.
available objects in the context map as well as some Freemarker examples [Insert link here].

**Configuring Notifications Content via System Properties**

Some content in notifications can be configured via system properties, such as, the number of log lines to include in email notifications that display log information. The relevant system properties are described below. For instructions on how to configure a system property, please refer to the Configuring System Properties page.

Please note, the system properties below do not add content to notifications. You still need to ensure that your notification templates contain the relevant entities to display the content. For example, changing the `bamboo.notifications.logLinesToInclude` property will not add log information to your notifications. It only modifies the number of log lines displayed in notification templates that already include logs.

<table>
<thead>
<tr>
<th>System Property</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bamboo.notifications.logLinesToInclude</code></td>
<td>Specifies the number of log lines to include in email notifications that display log information.</td>
<td>100</td>
</tr>
</tbody>
</table>

**Freemarker and Notification Templates**

Notification templates in Bamboo can be modified to customise the format and content of your notifications. The templates are written in Freemarker. This page is intended to complement the Modifying Notification Templates page and contains information on the Bamboo objects available from Freemarker templates, tips on writing Freemarker templates and examples.

On this page:

- Accessing Bamboo Data
- Special Considerations When Working With Freemarker
  - Never assume data exists
  - Check the encoding of your information
  - Use white space carefully
- Freemarker Examples
  - List Files in a Commit
  - Provide Test Error Details

**Accessing Bamboo Data**

Each individual notification has a different subset of data that can be accessed from the Freemarker templates. You can find information on the objects available in our javadocs below.

- Build Completed Notification ("All Completed Builds" and "Failed and First Success")
- After X Failed Builds Notification
- Build Commented Notification
- Build Hung Notification
- Build Error Notification

**Special Considerations When Working With Freemarker**

**Never assume data exists**

Unfortunately Freemarker is not very forgiving if data does not exist or is null. Hence, you will need to check whether information exists before adding it to a page. The sample code below shows how you can validate for non-existent data.

```
[#if issue.jiraIssueDetails.summary?has_content][/#if]
[#if issue.jiraIssueDetails.summary??][/#if]
${issue.jiraIssueDetails.summary?if_exists}
${issue.jiraIssueDetails.summary!}
```

**Check the encoding of your information**

Freemarker has built-in utilities for escaping special characters. These could be characters that you deliberately do not want to be interpreted as HTML, or data that could potentially contain malicious content. The sample code below shows how you can escape characters in Freemarker.
You can find more information on these utilities in the official Freemarker documentation.

**Use white space carefully**

When editing text email content and instant message content, you need to be very careful with spacing and line breaks. Any spaces and line breaks that you have entered in the Freemarker template will also exist in the evaluated content. Freemarker provides you with some utilities to remove white space, so that you can still retain some formatting in the templates.

More information can be found the official Freemarker documentation.

**Freemarker Examples**

Below are some raw examples of additional information that you can add to your emails.

Please note, these examples are intended to demonstrate the use of Freemarker and how to access Bamboo objects. You will need to modify these examples to include your desired formatting and make it work with your data.

**List Files in a Commit**

```freemarker
[<#if buildSummary.commits.size() > 0>]
  [<#list buildSummary.commits as commit>]
    [<#if commit_index gte 3>[<#break>[<#if>]]// only shows 3 commits
      Author:  &lt;a href="[Bui.displayAuthorOrProfileLink commit.author/]">${commit.author.fullName?html}
    </a>
    Comment: ${{commit.comment?html}
    <br/>
    [<#if commit.guessChangeSetId()?has_content>Revision: ${commit.guessChangeSetId()?html}
    <br/>
    [/<#if>]
    [<#if commit.files?has_content>Files:
      [<#list commit.files as file>]
        ${file.cleanName} [<#if file.revision?has_content](${file.revision})
        <br/>
      [/<#list>]
    [/<#if>]
    [/<#if>]
  [/<#list>]
[/<#if>]
[<#else>]
  This build does not have any commits.
[/<#if>]
```

**Provide Test Error Details**

```freemarker
${commit.comment?html} // for data to be encoded to be displayed as html
${commit.author?url} // for data to be encoded for a url
```
Configuring Bamboo to send SMTP Email

Bamboo can send email notifications about build results. There are two steps to setting this up:

1. **Configure Bamboo to send SMTP email** (see below).
2. **Configure a plan to send SMTP email notifications about build results** (see Adding or Removing Notifications for a Plan).

To configure Bamboo to send SMTP email,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Mail Server' link in the left navigation column (under 'Communication').
3. This will display the 'Mail Server Details' page (see screenshot below). Click the 'Edit' button.
4. In the 'Name' field, type a display-name for the email address in step 5 (below).
5. In the 'From Address' field, type the email address from which Bamboo notifications will be sent.
6. In the 'From Address' field, type the text (if any) with which you would like the email Subject line to begin. E.g. if you type '[Bamboo]', then people will receive emails with Subjects like this:
   - [Bamboo] TEST build 1,001 has FAILED (77 tests failed, no failures were new) : Change made by jsmith
   - [Bamboo] TEST build 1,002 was SUCCESSFUL (with 77 tests) : Change made by jsmith
7. If you are not using JNDI¹:
   a. In the 'SMTP Server' field, type the name of the email server via which Bamboo notifications will be sent. E.g. 'mail.myserver.com'.
   b. In the 'Username' field, type the login name of the account which Bamboo will use to login to the SMTP server.
   c. In the 'Password' field, type the password for the account specified in step 7 (if any).
   d. Go to step 8.
8. If you are using JNDI¹, type the JNDI name in the 'JNDI Location' field. The JNDI Location will depend on your application server, and on the location of the 'mail' resource within the JNDI tree you specify. E.g. 'jaas.com/env/mail/BambooMailServer'.
9. Type a test email address in the 'Test Recipient Address' box.
10. Click the 'Test' button, and verify that a test email is received.
11. Click the 'Save' button.

¹ Note re JNDI: As an alternative to specifying mail details directly in Bamboo, you can configure them in your application server (e.g. in the server.xml file — see Locating Important Directories and Files), and then use JNDI to look up a preconfigured mail session. JNDI has the following advantages:

   - **Centralised management** - mail details are configured in the same place as database details, and may be configured through your application server administration tools.
   - **Better security** - mail details are not available to Bamboo administrators through the Bamboo interface, and aren't stored in Bamboo backup files.
   - **More SMTP options** - e.g. SSL. If you want to use SMTP over SSL you will need to use JNDI.

*Screenshot: 'Email Server Details'*
Now that you have configured Bamboo's SMTP email capability, you can specify notifications for a plan.

Configuring Bamboo to use Instant Messaging (IM)

Bamboo can send Instant Messaging (IM) notifications about build results. There are two steps to setting this up:

1. **Configure Bamboo to use Instant Messaging** (see below).
2. **Configure a plan to send IM notifications about build results** (see Adding or Removing Notifications for a Plan).

To configure Bamboo to use Instant Messaging,
1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘IM Server’ link in the left navigation column (under ‘Communication’).
3. This will display the ‘Instant Messaging Server Details’ page. Click the ‘Edit’ button.
4. In the ‘Host’ field, type the address of your IM server (for example, ‘chat.atlassian.com’).
5. In the ‘Port’ field, type the TCP port that your organisation uses for IM traffic (or leave this field blank to have Bamboo either perform a DNS lookup or use the default port).
6. In the ‘Username’ field, type the login name of the IM account from which Bamboo notifications will be sent.
7. In the ‘Password’ field, type the password for the account specified in step 6.
8. If your IM server uses SSL, select the ‘Requires an SSL Connection’ check-box.
9. Type a test IM user's address in the ‘Test Recipient Address’ box.
10. Click the ‘Test’ button, and verify that a test IM message is received.
11. Click the ‘Save’ button.

Next step
Now that you have configured Bamboo's IM capability, you can specify notifications for a plan.

Configuring Bamboo to use Google Talk for Instant Messaging
If your Bamboo server has access to the internet, it can use Google Talk to send IM notifications about build results.

To configure Bamboo to use Google Talk for Instant Messaging,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'IM Server' link in the left navigation column (under 'Communication').
3. This will display the 'Instant Messaging Server Details' page. Click the 'Edit' button.
4. In the 'Host' field, type 'gmail.com'.

   - If your IM Server uses an "@googlemail.com" account, type 'googlemail.com' in the 'Host' field.
5. Leave the 'Port' field blank. Bamboo will perform a DNS lookup to figure out which port to use.
6. In the 'Username' field, type the login name of the Google account from which IM notifications will be sent. Only the account name needs to be included e.g. atlassianbamboo NOT atlassianbamboo@gmail.com.

   - Do not select "Requires an SSL connection"
7. In the 'Password' field, type the password for the account specified in step 6.
8. In the 'Test Recipient Address' box, type a test IM user's address (e.g. NOT atlassianbamboo@gmail.com).
9. Click the 'Test' button, and verify that the message is successfully received.
10. Click the 'Save' button.

---

Google Talk does not allow IM messages to be received unless the receiver has approved the sender. Please ensure that the Gmail user specified in step 6 is approved by each Google Talk recipient. That is, ensure that the 'Host' and 'Username' have previously sent messages to each other via Google Talk.

---

Additional notes about using Google Talk:

- The Google Talk service is hosted at talk.google.com. The default port is 5222. (Note: be aware that your firewall might be blocking traffic to this port.)
- TLS is required.
- The only supported authentication mechanism is SASL PLAIN. For additional information, please see: http://code.google.com/apis/talk/open_communications.html

---

Managing Users and Permissions

About users and authors

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user. Depending on your organisation's needs, you can configure Bamboo to grant access to non-users. However, only Bamboo users can:

- view the 'My Bamboo' tab on the Dashboard.
- belong to a group.

About groups

Bamboo groups are used to specify which users will have global permissions and plan permissions. They can also be used to specify which users will receive notifications about a plan's build results. You can create and delete as many groups as you need. You will typically create at least one group per project.

A special group called bamboo-admin is automatically created when you install Bamboo. Members of this group have Bamboo administration rights.

About permissions

A plan permission is the ability to perform a particular operation in relation to a build plan. For each plan, different permissions can be granted to
particular groups and/or users. A global permission is the ability to perform a particular operation in relation to Bamboo as a whole.

**Further Information**

Read more about users, groups and security in Bamboo:

- Enabling or disabling Contact Details Display
- Enabling or disabling Public Signup
- Managing Groups
  - Adding Users to and removing them from Groups
  - Creating a Group
  - Deleting a Group
- Managing Permissions for Users and Groups
  - Allowing Anonymous Users to access Bamboo
  - Granting Global Permissions to Users or Groups
  - Granting Plan Permissions to Users or Groups
- Managing Users
  - Changing a User's Password or Details
  - Creating a User
  - Deleting or deactivating a User
  - Granting Administration Rights to a User
- Working with External User Repositories
  - Integrating Bamboo with Crowd
  - Integrating Bamboo with LDAP
    - Configuring the Caching of your LDAP Repository

**Managing Users**

**About users and authors**

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user. Depending on your organisation’s needs, you can configure Bamboo to grant access to non-users. However, only Bamboo users can:

- view the 'My Bamboo' tab on the Dashboard.
- belong to a group.

Read more about users:

- Creating a User
- Changing a User's Password or Details
- Deleting or deactivating a User
- Granting Administration Rights to a User

**Creating a User**

A user is someone who can login to Bamboo. An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user. Depending on your organisation's needs, you can configure Bamboo to grant access to non-users. However, only Bamboo users can:

- view the 'My Bamboo' tab on the Dashboard.
- belong to a group.

  To create a Bamboo user,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Users' link in the left navigation column.
3. The 'Manage Users' screen will be displayed. The 'Add User' section (as shown below) will be displayed at the bottom of the 'Manage Users' screen.
4. In the 'Username' field, type the user's login name.
   - Note that the Username cannot be changed after the user is created.
5. In the 'Password' and 'Confirm Password' fields, type the user's password.
   - The user can easily change their password later.
6. In the 'Full Name' field, type the user's display-name.
7. In the 'Email' field, type the user's email address. This address is where the user will receive password notifications.
8. (optional) In the 'Jabber address' field, type the user's Instant Messaging (IM) address. This address is where the user will receive any group-based notifications about build results.
   - If no IM address is specified, Bamboo will not be able to recognise the user's context when interacting via IM.
9. Select at least one group from the 'Groups' list. (To select multiple groups, press the <Ctrl> key.) Once you add a user to a group, the group will be shown in blue; groups to which the user does not belong are shown in white.
10. If the user is a Bamboo author, select 'Add Alias' (instead of 'None') in the 'Source Repository Alias' field. This will display the 'New alias' field. Type the user's login name for their source-code repository.
11. If you don't know the user's login name for their source-code repository, they can specify it themselves later.
12. Click the 'Save' button.

**Screenshot: 'Add User'**

### Add User

#### User Details

Enter the details of the user to add to Bamboo, then click Save.

- **Username:** jsmith
  - The name to use to login to Bamboo.
- **Password:** ********
- **Confirm Password:** ********
- **Full Name:** Jill Smith
- **Email:** jsmith@mycompany.com
- **Jabber Address:**
  - The address to which instant messages will be sent if they are enabled for a build.
- **Groups:**
  - bamboo-admin
  - Each user must be in at least one group.
- **Source Repository Alias:** Add Alias
  - The repository name the user commits with in repositories.
- **New alias:** jills

**Changing a User's Password or Details**

To change a user's password,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Users' link in the left navigation column.
3. The 'Manage Users' screen will be displayed. To locate the user, type part of their 'Username', 'Full Name' or 'Email' and click the 'Enter' button. This will display a list of matching users.
4. Click the 'Edit' link (in the 'Operations' column) that corresponds to the user.
5. The 'User Details' screen will be displayed. Type the new password in the 'Password' and 'Confirm Password' fields.
   - If you have configured SMTP email on your Bamboo server, the user will automatically receive an email containing their new password. The user can easily change their password later.
6. Click the 'Save' button.

To change a user's details (e.g. Email or Full Name),

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Users' link in the left navigation column.
3. The 'Manage Users' screen will be displayed. To locate the user, type part of their 'Username', 'Full Name' or 'Email' and click the 'Enter' button. This will display a list of matching users.
4. Click the 'Edit' link (in the 'Operations' column) that corresponds to the user.
5. The 'User Details' screen will be displayed. Enter the new details as described in Creating a User.
   - If you do not need to change the user's password, simply leave the 'Password' and 'Confirm Password' fields blank.
6. Click the 'Save' button.

Note
Users who have forgotten their passwords can click the 'Forgotten your password?' link on the Bamboo login screen. This will automatically generate a new password and email it to the user (provided the Bamboo server has been configured to send SMTP email).
Logged-in users can also change their own password and details, as described in Managing your User Profile in the Bamboo User's Guide:
Unable to render {children}. Page not found: 11. Editing your User Profile.

Deleting or deactivating a User

Deleting a user removes their Bamboo user account. Deactivating a user revokes their permission to login to Bamboo.

Note that deleting a Bamboo user will not delete their author data — that is, their author statistics and code check-in comments will still exist in Bamboo.

Also note that:
- You cannot delete a user who has created labels or comments about build results. You may want to deactivate them instead.
- You cannot delete the user account with which you are currently logged in to Bamboo.

To delete a Bamboo user,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Users' link in the left navigation column.
3. The 'Manage Users' screen will be displayed. Locate the relevant user in the list, and click the corresponding 'Delete' link in the 'Operations' column.

Screenshot: Delete User
Manage Users

You can use this page to view, update and delete users.

<table>
<thead>
<tr>
<th>Username</th>
<th>Email</th>
<th>Full Name</th>
<th>Groups</th>
<th>Repository Alias</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td><a href="mailto:admin@mycompany.com">admin@mycompany.com</a></td>
<td>admin</td>
<td>bamboo-admin</td>
<td></td>
<td>Edit</td>
</tr>
<tr>
<td>jsmith</td>
<td><a href="mailto:jsmith@mycompany.com">jsmith@mycompany.com</a></td>
<td>Jill Smith</td>
<td>project-x</td>
<td>jills</td>
<td>Edit, Delete</td>
</tr>
</tbody>
</table>

To deactivate a Bamboo user,

- please see Deactivating a Bamboo user

Granting Administration Rights to a User

In Bamboo, there are two types of administrators:

- **Global administrators** — that is, people with the ‘Admin’ global permission. These people can access the Bamboo ‘Administration’ menu. They can also administer every plan.
- **Plan administrators** — that is, people with the ‘Admin’ and ‘Edit’ plan permissions. These people can administer a particular plan.

To grant global administration rights to a user,

- Either grant the ‘Admin’ global permission to the user explicitly (as described in Granting Global Permissions to Users or Groups);
  OR:
  - Add the user to a group which has the ‘Admin’ global permission (as described in Adding Users to and removing them from Groups).

To grant plan administration rights to a user,

- Either grant the ‘Admin’ and ‘Edit’ plan permissions to the user explicitly (as described in Granting Plan Permissions to Users or Groups);
  OR:
  - Add the user to a group which has the ‘Admin’ and ‘Edit’ plan permissions (as described in Adding Users to and removing them from Groups).

Managing Groups

About groups

Bamboo groups are used to specify which users will have global permissions and plan permissions. They can also be used to specify which users will receive notifications about a plan's build results. You can create and delete as many groups as you need. You will typically create at least one group per project.

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have Bamboo administration rights.

Read more about managing groups for your users:

- Creating a Group
- Deleting a Group
- Adding Users to and removing them from Groups

Creating a Group

Bamboo groups are used to specify which users will have global permissions and plan permissions. They can also be used to specify which users will receive notifications about a plan's build results. You can create and delete as many groups as you need. You will typically create at least one group per project.
A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have **Bamboo administration rights**.

To create a group,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Groups' link in the left navigation column.
3. The 'Manage Groups' screen will be displayed. The 'Create Group' section (as shown below) will be displayed at the bottom of the 'Manage Groups' screen.
4. In the 'Group Name' field, type a name for your new group.
   - Note that the Group Name cannot be changed after the group is created.
5. Select relevant users from the 'Users in Group' list. Hold the <Ctrl> to select multiple users.
   - You can also add or remove users from the group later if required.
6. Click the 'Save' button.

Screenshot: Create Group

Create Group

<table>
<thead>
<tr>
<th>Group Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group Name:</strong></td>
</tr>
<tr>
<td><strong>Users in Group:</strong></td>
</tr>
<tr>
<td><strong>Save</strong></td>
</tr>
</tbody>
</table>

Deleting a Group

Bamboo groups are used to specify which users will have **global permissions** and **plan permissions**. They can also be used to specify which users will receive notifications about a plan's build results. You can create and delete as many groups as you need. You will typically create at least one group per project.
A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have Bamboo administration rights.

The **bamboo-admin** group cannot be deleted.

**To delete a group,**

1. Click the **Administration** link in the top navigation bar.
2. Click the **Groups** link in the left navigation column.
3. The **Manage Groups** screen will be displayed. Locate the relevant group in the list, and click the corresponding **Delete** link in the **Operations** column.

![Screenshot: Delete Group]

**Manage Group**

You can use this page to view and delete groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>Number of Users</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo-admin</td>
<td>28</td>
<td>Edit</td>
</tr>
<tr>
<td>crowd-developers</td>
<td>3</td>
<td>Edit</td>
</tr>
</tbody>
</table>

### Adding Users to and removing them from Groups

Bamboo groups are used to specify which users will have global permissions and plan permissions. They can also be used to specify which users will receive notifications about a plan's build results. You can create and delete as many groups as you need. You will typically create at least one group per project.

A special group called **bamboo-admin** is automatically created when you install Bamboo. Members of this group have Bamboo administration rights.

**To add users to a group,**

1. Click the **Administration** link in the top navigation bar.
2. Click the **Groups** link in the left navigation column.
3. The **Manage Groups** screen will be displayed. Locate the relevant group in the list, and click the corresponding **Edit** link in the **Operations** column.
4. The **Edit Group Details** screen will be displayed. Users who already belong to the group are shown in blue; users who do not currently belong to the group are shown in white. Press the <Ctrl> key and hold it while you select the user(s) whom you want to add to the group.
5. Click the **Save** button.

![Screenshot: Edit Group Details]
To remove users from a group,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Groups' link in the left navigation column.
3. The 'Manage Groups' screen will be displayed. Locate the relevant group in the list, and click the corresponding 'Edit' link in the 'Operations' column.
4. The 'Edit Group Details' screen will be displayed. Users who belong to the group are shown in blue. Press the <Ctrl> key and hold it while you deselect the user(s) whom you want to remove from the group.
5. Click the 'Save' button.

You cannot remove a user from the bamboo-admin group if they are the only member.

Managing Permissions for Users and Groups

About permissions

A plan permission is the ability to perform a particular operation in relation to a build plan. For each plan, different permissions can be granted to particular groups and/or users.

A global permission is the ability to perform a particular operation in relation to Bamboo as a whole.

Read more about managing permissions for users and groups:

• Granting Plan Permissions to Users or Groups
Granting Plan Permissions to Users or Groups

A plan permission is the ability to perform a particular operation in relation to a build plan. For each plan, different permissions can be granted to particular groups and/or users.

The following plan permissions are available:

<table>
<thead>
<tr>
<th>Plan permission</th>
<th>Description</th>
<th>Can be granted to</th>
</tr>
</thead>
</table>
| 'View' | Permission to:  
  - view this plan's build results  
  - add comments or labels to this plan's build results¹ | - a particular user  
- a particular group  
- all logged-in users  
- anonymous users² |
| 'Edit' | Permission to view and edit this plan's configuration, except for the plan's permissions. | - a particular user  
- a particular group  
- all logged-in users  
- anonymous users² |
| 'Build' | Permission to:  
  - manually start/stop a build for this plan.  
  - enable/disable this plan from submitting builds to the queue. | - a particular user  
- a particular group  
- all logged-in users  
- anonymous users² |
| 'Clone' | Permission to copy this plan when creating a new plan.  
(Note: only users with the 'Create Plan' global permission can create new plans.) | - a particular user  
- a particular group |
| 'Admin' | Permission to:  
  - edit this plan's permissions.  
  - delete this plan's build results and working files. | - a particular user  
- a particular group |

¹ Only logged-in users (not anonymous users) can label or comment on a build result.
² Anonymous users cannot access Bamboo at all unless they have been granted the 'Access' global permission. See Allowing Anonymous Users to access Bamboo.

How plan permissions are granted:

- People who have the 'Admin' global permission can 'bulk edit' permissions for multiple plans at the same time, as described below. Note that this will overwrite any pre-existing plan permissions.
- People who have the 'Admin' plan permission for one or more plans, but do not have the 'Admin' global permission, can only edit one plan at a time, as described in Specifying a Plan's Permissions.

The processes for granting and revoking permissions across multiple plans are as follows. Note that, for ongoing ease of management, it is recommended that you grant permissions to groups rather than to individual users.

To grant multiple plan permissions to a user,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Edit Plan Permissions' link in the left navigation column.
3. The first screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below). Select the plans whose permissions you wish to edit, then click the 'Next' button (at the bottom of the screen).
4. The second screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below).
5. In the 'Grant permission to' list at the bottom of the screen, select 'User'.
6. Type the username into the box, or click the following icon to select from a list of users:
7. Click the 'Add' button.
8. The user will be added to the list of users on the screen. Select the check-box for each permission that you wish to grant to this user.
9. Click the 'Save' button.

To grant multiple plan permissions to a group,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Edit Plan Permissions' link in the left navigation column.
3. The first screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below). Select the plans whose permissions you wish to edit, then click the 'Next' button (at the bottom of the screen).
4. The second screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below).
5. In the 'Grant permission to' list at the bottom of the screen, select 'Group'.
6. Type the group name into the box.
7. Click the 'Add' button.
8. The group will be added to the list of groups on the screen. Select the check-box for each permission that you wish to grant to this group.
9. Click the 'Save' button.

To grant multiple plan permissions to all Bamboo users,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Edit Plan Permissions' link in the left navigation column.
3. The first screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below). Select the plans whose permissions you wish to edit, then click the 'Next' button (at the bottom of the screen).
4. The second screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below).
5. Locate 'Logged in users' (under 'Other').
6. Select the check-box for each permission that you wish to grant to all Bamboo users.
7. Click the 'Save' button.

To grant multiple plan permissions to anonymous users,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Edit Plan Permissions' link in the left navigation column.
3. The first screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below). Select the plans whose permissions you wish to edit, then click the 'Next' button (at the bottom of the screen).
4. The second screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below).
5. Locate 'Anonymous users' (under 'Other').
6. Select the check-box for each permission that you wish to grant to all anonymous users.
7. Click the 'Save' button.

To revoke multiple plan permissions,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Bulk Edit Plan Permissions' link in the left navigation column.
3. The first screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below). Select the plans whose permissions you wish to edit, then click the 'Next' button (at the bottom of the screen).
4. The second screen of the 'Bulk Edit Plan Permissions Wizard' will be displayed (see below).
5. Locate the relevant user/group/all logged-in users/anonymous users.
6. Deselect the check-box for each permission that you wish to revoke from the user/group/all users/anonymous users.
7. Click the 'Save' button.
Bulk Edit Plan Permissions Wizard

Select Plans

All existing permissions defined for the plans you select will be deleted and replaced with the new permissions you specify.

You can apply a consistent set of permissions to multiple plans with this wizard. Simply select the plans you want to edit and then specify the permissions you wish to apply.

Plans:

- **Confluence (CONF)**
  - Unit Tests - JDK 1.4 (MAIN)
  - Stable RPC Unit Tests (STABRPC)
  - Stable Unit Tests (STABUNIT)
  - Atlassian User Stable Unit Tests (ATLUSRSTABUNIT)
  - Atlassian User Stable Integration Tests (ATLUSRSTABIT)
  - Unit Tests - JDK 1.5 (MAINJDK15)
  - Distribution (DIST)
  - Unit Tests - JDK 1.6 (MAINJDK16)
  - Crowd 1.1 JDK 1.6 (11JDK16)

- **Crowd (CWD)**
  - Main Build (MAIN)
  - Crowd JDK 1.6 (JDK16)
  - Crowd JDK 1.4 (JDK14)
  - Integration Tests (INT)
  - Build Plugin (BUILDPLUGIN)
  - Crowd 1.1 JDK 1.4 (11JDK14)
  - Crowd 1.1 - Integration Tests (11INT)
  - Crowd 1.1 - Main Build (11MAIN)

Next >>
Granting Global Permissions to Users or Groups

A global permission is the ability to perform a particular operation in relation to Bamboo as a whole. You can grant the following global permissions:

<table>
<thead>
<tr>
<th>Global permission</th>
<th>Description</th>
<th>Can be granted to</th>
</tr>
</thead>
</table>
| 'Access'          | Permission to view the Bamboo system. (Note that the ability to view build plans and build results is subject to individual plan permissions.) | - a particular user  
- a particular group  
- all logged-in users  
- anonymous users¹ |
| 'Create Plan'     | Permission to create new build plans. | - a particular user  
- a particular group  
- all logged-in users |
| 'Admin'           | Permission to:  
- access the Bamboo 'Administration' menu.  
- delete plans.  
The 'Admin' global permission also includes all plan permissions, for every plan. | - a particular user  
- a particular group |

¹ i.e. people who are not logged in to Bamboo.

The processes for granting and revoking global permissions are described below.

To grant global permissions to a user,
To grant global permissions to a group,

1. Click 'Home' to go to the Dashboard.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Global Permissions' link in the left navigation column to display the 'Global Permissions' screen (see screenshot below).
4. In the 'Grant permission to' list at the bottom of the screen, select 'Group'.
5. Type the group name into the box.
6. Click the 'Add' button.
7. The group will be added to the list of groups on the 'Permissions' tab. Select the check-box for each permission that you wish to grant to this group.
8. Click the 'Save' button.

To grant global permissions to all Bamboo users,

1. Click 'Home' to go to the Dashboard.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Global Permissions' link in the left navigation column to display the 'Global Permissions' screen (see screenshot below).
4. Locate 'All logged in users' (under 'Other').
5. Select the check-box for each permission that you wish to grant to all Bamboo users.
6. Click the 'Save' button.

To grant global permissions to anonymous users,

The 'Access' global permission is the only global permission that can be granted to anonymous users.

1. Click 'Home' to go to the Dashboard.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'Global Permissions' link in the left navigation column to display the 'Global Permissions' screen (see screenshot below).
4. Locate 'Anonymous users' (under 'Other').
5. Select the 'Access' check-box.
6. Click the 'Save' button.

To revoke global permissions,
1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘Administration’ link in the top navigation bar.
3. Click the ‘Global Permissions’ link in the left navigation column to display the ‘Global Permissions’ screen (see screenshot below).
4. Locate the relevant user/group/all logged-in users/anonymous users.
5. Deselect the check-box for each permission that you wish to revoke from the user/group/all users/anonymous users.
   - If you deselect all permissions for a user or group, that user or group will disappear from the ‘Permissions’ tab for this plan.
6. Click the ‘Save’ button.

### Screenshot: Global Permissions

You can edit your global application level permissions here. Permissions can be granted to specific users or groups. Please note these are global application permissions. For plan level permissions, please go to the plan configuration page.

### Bamboo Application Permissions

<table>
<thead>
<tr>
<th>Groups</th>
<th>Access</th>
<th>Create Plan</th>
<th>Admin</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo-admin</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other</th>
<th>Access</th>
<th>Create Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>All logged in users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anonymous users</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Grant permission to [User]

Add

Save Cancel

### Allowing Anonymous Users to access Bamboo

Allowing anonymous users to access your Bamboo system means that people who aren’t logged in to Bamboo will be able to perform most of the functions described in the Bamboo User's Guide (e.g. generating reports; viewing plans and build results) — subject to individual plan permissions.

Note that people who aren’t logged in to Bamboo do not have a ‘My Bamboo’ tab on their Dashboard.

**To allow anonymous users to access Bamboo,**

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘Administration’ link in the top navigation bar.
3. Click the ‘Global Permissions’ link in the left navigation column to display the ‘Global Permissions’ screen (see screenshot below).
4. Locate ‘Anonymous users’ (under ‘Other’).
5. Select the ‘Access’ check-box.
6. Click the ‘Save’ button.

Anonymous users will now be able to access your Bamboo system. However, they will only be able to view plans and build results for plans where the ‘Access’ plan permission has been granted to ‘Anonymous users’.

### Screenshot: Global Permissions
Enabling or disabling Contact Details Display

If you enable contact details display to your Bamboo system, the full contact details for a user, including email address, IM address, and group, will be visible to any visitors of Bamboo. The email addresses of administrators in the ‘Contact Administrators’ page will also be visible.

To enable contact details display,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Security Settings’ link in the left navigation column.
3. Select the ‘Enable contact details to be displayed?’ check-box.
4. Click the ‘Save’ button.

To disable contact details display,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Security Settings’ link in the left navigation column.
3. Deselect the ‘Enable contact details to be displayed?’ check-box.
4. Click the ‘Save’ button.

Enabling or disabling Public Signup

If you enable signup for your Bamboo system, visitors can create their own Bamboo user accounts.

To enable signup,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column.
3. Select the 'Enable Signup?' check-box.
4. Click the 'Save' button.
5. Log out of Bamboo and verify that the top navigation bar now contains a 'Signup' link (see screenshot below).

To disable signup,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column.
3. Deselect the 'Enable Signup?' check-box.
4. Click the 'Save' button.

Screenshot: 'Signup'

---

**Working with External User Repositories**

You can integrate external user repositories with Bamboo:

- Integrating Bamboo with Crowd
- Integrating Bamboo with LDAP
  - Configuring the Caching of your LDAP Repository

**Integrating Bamboo with Crowd**

Atlassian's [Crowd identity management system](#) can be integrated with Bamboo. Please refer to the [appropriate documentation](#) for the latest version of Crowd, which can be found in the [Crowd Administrator's Guide](#). If you are using an older version of Crowd, find the documentation via the [Crowd documentation homepage](#).

**External User Management Configuration**

If you are connecting Bamboo to an external user management system and do not have rights to update user attributes there, you will need to prevent users from being updated in Bamboo. In this case, you should ensure that the 'Read-only External User Management?' check-box is checked. The table below outlines the correct configuration for Bamboo, depending on your external user management setup:

<table>
<thead>
<tr>
<th>External User Management Setup</th>
<th>'Read-only External User Management?' check-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo integrated with — Crowd using the Crowd database (i.e. Internal Directories)</td>
<td>Unchecked</td>
</tr>
<tr>
<td>Bamboo integrated with — Crowd connected to a read-only LDAP</td>
<td>Checked</td>
</tr>
<tr>
<td>Bamboo integrated with — Crowd connected to a read-write LDAP</td>
<td>Unchecked</td>
</tr>
</tbody>
</table>
Integrating Bamboo with LDAP

Bamboo can be integrated with LDAP for the authentication and authorisation of LDAP users. The Integrating Bamboo with LDAP instructions below describe how you can set this up. Please note that Bamboo does not currently support multiple LDAP servers. If you need to connect to multiple LDAP servers, please consider using Atlassian's Crowd.

If you choose to integrate Bamboo with LDAP, please note that you cannot manage LDAP accounts or user groups from Bamboo. Please refer to the External User Management instructions on this page for details on how to manage your users and groups.

⚠️ In Bamboo versions prior to 1.2.2, it is not possible to perform XML backups of your Bamboo instance when integrated with LDAP. In Bamboo version 1.2.2 and later, exports are possible, but user data will not be included in the export.

On this page:
- Integrating Bamboo with LDAP
  - Before you begin
  - Step 1 - Backup your data
  - Step 2 - Configure Connection Details
  - Step 3 - Map LDAP Data Tree
  - Step 4 - Optional LDAP Settings
- External User Management
  - Step 1 - Configuring Bamboo for External User Management
  - Step 2 - Assigning LDAP Users to Bamboo Groups
- Troubleshooting

Integrating Bamboo with LDAP

Before you begin

You will need to confirm that your LDAP server is compatible and set up correctly before integrating it with Bamboo. Please check your LDAP server against the requirements below:

- **Check your LDAP server version** — Supported versions are v2 and v3. Supported LDAP servers include OpenLDAP, Microsoft Active Directory, Novell eDirectory, and any server that uses Java JNDI-LDAP mapping.
- **Check whether your LDAP or Active Directory server supports static groups** — Your LDAP or Active Directory server must support static groups. This means that the user DNs must be stored against a membership attribute inside an LDAP group. An example of a static group is shown below:

```
Dn: CN=Sales and Marketing,CN=Users,DC=ad,DC=atlassian,DC=com
objectClass: top; group;
objectClass: group;
cn: Sales and Marketing;
distinguishedName: CN=Sales and Marketing,CN=Users,DC=ad,DC=atlassian,DC=com;
name: Sales and Marketing;
...
member: CN=John Smith,CN=Users,DC=ad,DC=atlassian,DC=com
member: CN=Sally Smith,CN=Users,DC=ad,DC=atlassian,DC=com
...
```

The membership attribute in this case is member, but this is not required. Note that the full DNs of John Smith and Sally Smith are listed. If the values against member are not full DNs, but are just usernames, then you need to add the flag <useUnqualifiedUsernameForMembershipComparison>true</useUnqualifiedUsernameForMembershipComparison> to your LDAP configuration. Open Directory on OS X uses this configuration.

- **Ensure that you do not have an LDAP group called ‘bamboo-admin’** — The bamboo-admin group is reserved by Bamboo.
- **Ensure that you do not have duplicate users on your LDAP** — If you have users on your LDAP that are also on Bamboo, the LDAP users will take precedence over your local Bamboo users.
- **Ensure that you do not have duplicate groups on your LDAP** — If you have groups on your LDAP that are also on Bamboo, this may cause unpredictable behaviour when you attempt to integrate your LDAP server with Bamboo.

Step 1 — Backup your data

We strongly recommend that you backup your data before attempting LDAP integration.

Step 2 — Configure Connection Details
The LDAP server connection is specified by manually editing the file `atlassian-user.xml`.

**To configure your connection details,**

1. Edit the file `.../webapp/WEB-INF/classes/atlassian-user.xml` and configure the connection AD or LDAP.
2. Check your configuration against the example connection details shown below.

   ```xml
   <ldap key="ldapRepository" name="LDAP Repository@hecate.atlassian.com" cache="true">
     <host>hecate.atlassian.com</host>
     <port>389</port>
     <securityPrincipal>cn=admin,dc=atlassian,dc=private</securityPrincipal>
     <securityCredential>secret</securityCredential>
     <securityProtocol>plain</securityProtocol>
     <securityAuthentication>simple</securityAuthentication>
     <baseContext>dc=atlassian,dc=private</baseContext>
   </ldap>
   ...
   ``

3. Please ensure that the following line is also active in your `atlassian-user.xml` (it should be there by default):

   ```xml
   <hibernate name="Hibernate Repository" key="hibernateRepository" description="Hibernate Repository" />
   ``

**Step 3 — Map LDAP Data Tree**

**To map the LDAP Data Tree,**

1. To configure the mappings in `atlassian-user.xml` for either AD or LDAP, please see:
   - Mapping Active Directory
   - Mapping other LDAP servers
2. Check your configuration against the example connection details shown below.

   ```xml
   ...
   <baseUserNamespace>dc=staff,dc=perftest,dc=atlassian,dc=private</baseUserNamespace>
   <baseGroupNamespace>dc=groups,dc=perftest,dc=atlassian,dc=private</baseGroupNamespace>
   <usernameAttribute>cn</usernameAttribute>
   <userSearchFilter>(objectClass=inetorgperson)</userSearchFilter>
   <firstnameAttribute>givenname</firstnameAttribute>
   <surnameAttribute>sn</surnameAttribute>
   <emailAttribute>mail</emailAttribute>
   <groupnameAttribute>cn</groupnameAttribute>
   <groupSearchFilter>(objectClass=groupOfNames)</groupSearchFilter>
   <membershipAttribute>member</membershipAttribute>
   ...
   ``

**Filters:** `<userSearchFilter>` and `<groupSearchFilter>` may use the AD specific filter syntax. Operators such as `&` need to be escaped.

   ```xml
   <userSearchFilter>(&amp;{objectClass=inetorgperson} {memberOf=cn=bamboo-usr,ou=groups,DC=atlassian,DC=com})
   ``

**Step 4 — Optional LDAP Settings**

The following settings do not appear in the default `atlassian-user.xml` file. Their default values are as follows:
If you want to override these default values, you can specify any or all of them by adding them onto the end of the `atlassian-user.xml` file. For example, to add your own value for the `<initSize>` setting, you would add an extra line before the last line shown in 'Stage 3' above:

```xml
<initSize>10</initSize>
```

It is important that the connection pool timeout value be set to 0, as this will force Atlassian User (via the JNDI layer) to clean up lingering connections that have lived past one request. For more information about LDAP pools please see http://java.sun.com/products/jndi/tutorial/ldap/connect/config.html.

**External User Management**

You cannot manage LDAP accounts or user groups from Bamboo. Bamboo will continue to use local users and groups, even when LDAP is enabled. If you need to assign LDAP users to particular groups referenced by Bamboo (e.g. for permissions or notifications), the workaround is to assign your LDAP users to local Bamboo groups, and reference these groups rather than LDAP groups.

⚠️ Please note that once LDAP has been enabled, reverting back to local user management is not possible.

**Step 1 — Configuring Bamboo for External User Management**

Please ensure that 'Read-only External User Management' is turned OFF in Bamboo before assigning LDAP users to Bamboo groups. The Read-only External User Management check-box in Bamboo controls whether users and groups in Bamboo are editable. Setting this option to ON will make users and groups in Bamboo read-only (i.e. the implication being that you will be managing your users and groups externally).

⚠️ If you are using Crowd together with an LDAP, please read the documentation on Integrating Bamboo with Crowd to see our recommended settings for the 'Read-only External User Management' check-box.

**To disable Read-only External User Management,**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column.
3. De-select the 'Read-only External User Management' check-box.
4. Click the 'Save' button.

**Step 2 — Assigning LDAP Users to Bamboo Groups**

Once Bamboo is started with 'Read-Only External User Management' disabled, you can assign LDAP users to Bamboo groups. Please see Adding Users to and removing them from Groups.

⚠️ Please note, the 'View Users' and 'View Groups' screens in Bamboo currently will not list all of your LDAP users/groups (please see BAM-1963 for details).

**Troubleshooting**
To check whether the atlassian-user.xml file is correctly configured, please run the Paddle tool to debug the LDAP configuration in your atlassian-user.xml file. For further reference, please visit the Paddle usage page.

Configuring the Caching of your LDAP Repository

On this page:

- Disabling the Caching of Users
- Enabling the Caching of Users
- Configuring the LDAP Caches
  - Configuring Caches for Users
  - Configuring Caches for User Groups

Disabling the Caching of Users

By default, caching is activated for your LDAP users. We recommend that you do not disable caching of your LDAP users, as your LDAP repository may be overloaded by the high volume of requests by Bamboo.

To disable the caching of users,

1. Click the ‘Administration’ link in the top navigation bar.
2. Edit the file .../webapp/WEB-INF/classes/atlassian-user.xml
3. Set the property cache=”false” on your LDAP repository, as shown in the example below:

```xml
<ldap key="myLdapRepository" name="LDAP Repository@hecate.atlassian.com" cache="false">
  <host>hecate.atlassian.com</host>
  <port>389</port>
</ldap>
```

Enabling the Caching of Users

By default, caching is activated for your LDAP users. If you need to enable caching, follow the instructions below:

To enable the caching of users,

1. Edit the file .../webapp/WEB-INF/classes/atlassian-user.xml
2. Set the property cache=”true” on your LDAP repository as shown in the example below:

```xml
<ldap key="myLdapRepository" name="LDAP Repository@hecate.atlassian.com" cache="true">
  <host>hecate.atlassian.com</host>
  <port>389</port>
</ldap>
```

Configuring the LDAP Caches

Bamboo uses a number of caches for managing an LDAP repository, each of which can be configured differently. You must enable caching, as described above, before configuring the caches. The caches used by Bamboo are:

- Configuring Caches for Users
- Configuring Caches for User Groups

Each cache can be configured by following the instructions below:

To configure a cache,

1. Edit the file .../webapp/WEB-INF/classes/ehcache.xml.
2. Find the cache that you wish to edit. Examples of each of the caches are described in the Configuring Caches for Users and Configuring Caches for User Groups sections below.
3. Modify the cache, as desired. The following properties can be configured for each cache:
- **maxElementsInMemory** (mandatory) - Sets the maximum number of objects that will be created in memory
- **eternal** (mandatory) - Sets whether elements are eternal. If eternal, timeouts are ignored and the element is never expired.
- **timeToIdleSeconds** (optional) - Sets the time to idle for an element before it expires. i.e. The maximum amount of time between accesses before an element expires. This is only used if the element is not eternal. A value of 0 means that an Element can idle for infinity. The default value is 0.
- **timeToLiveSeconds** (optional) - Sets the time to live for an element before it expires i.e. The maximum time between creation time and when an element expires. This is only used if the element is not eternal. A value of 0 means that an Element can live for infinity. The default value is 0.

If you have caching turned on Bamboo will, by default, set the cache to eternal (elements will never expire), and set the maximum number of elements stored to 500. These can be configured to speed up user retrieval, reduce memory usage or reduce the load on the LDAP repository.

### Configuring Caches for Users

⚠️ In each of the examples below, replace `myLdapRepository` with the key of the repository specified in `atlassian-user.xml`

- LDAPUserManagerReadOnly.*.users
  
  LDAPUserManagerReadOnly.*.users stores the individual users, if you have difficulties with Bamboo picking up new user additions in the LDAP repository you will need to alter the configuration of this cache. In the example below, the users will expire after 5 minutes.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.users">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

- LDAPUserManagerReadOnly.*.users_ro
  
  LDAPUserManagerReadOnly.*.users_ro stores whether or not the users are read only. This will have no effect on the functionality of Bamboo, but you may wish to modify this cache for performance and memory tuning purposes.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.users_ro">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

- LDAPUserManagerReadOnly.*.repository
  
  LDAPUserManagerReadOnly.*.repository stores which repository the user belongs to. Bamboo does not yet support multiple repositories, so modifying this cache will have no effect on functionality. However, you may wish to modify this cache for performance and memory tuning purposes.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPUserManagerReadOnly.myLdapRepository.repository">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

### Configuring Caches for User Groups

⚠️ In each of the examples below, replace `myLdapRepository` with the key of the repository specified in `atlassian-user.xml`
• LDAPGroupManagerReadOnly.*.groups

LDAPGroupManagerReadOnly.*.groups stores the available groups in LDAP. If you wish Bamboo to pick up changes made to groups, then you will need to configure this cache appropriately.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.groups">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

• LDAPGroupManagerReadOnly.*.groups_hasMembership and LDAPGroupManagerReadOnly.*.groups_getGroupsForUser

The LDAPGroupManagerReadOnly.*.groups_hasMembership and LDAPGroupManagerReadOnly.*.groups_getGroupsForUser caches store the associations between users and groups. If you wish Bamboo to pick up changes made to group memberships then you will need to configure these caches appropriately.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.groups_hasMembership">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.groups_getGroupsForUser">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

• LDAPGroupManagerReadOnly.*.repositories

LDAPGroupManagerReadOnly.*.repositories stores which repository the group belongs to. Bamboo does not yet support multiple repositories, so modifying this cache will have no effect on functionality. However, you may wish to modify this cache for performance and memory tuning purposes.

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.repositories">
  maxElementsInMemory="500"
  eternal="false"
  timeToIdleSeconds="300"
  timeToLiveSeconds="300"
</cache>
```

Managing Data and Backups

Managing Data and Backups

- Enabling Expiry of Build Results
- Exporting Data for Backup
- Importing Data from Backup
- Locating Important Directories and Files
- Moving your Bamboo Data to a different Database
- Optimising or Re-indexing Data
- Specifying a Backup Schedule
- Specifying Bamboo’s Working Directory
- Viewing your Database Connection Details
Locating Important Directories and Files

On this page:

- **BAMBOO SERVER**
  - Bamboo home directory
  - Bamboo installation directory
- **BAMBOO AGENT**
  - Bamboo agent home directory

**BAMBOO SERVER**

When you installed your Bamboo server, you specified two directories:

- **Bamboo installation directory** — This is the directory where the Bamboo application files are installed. (The default location depends on your operating system: Windows, Unix/Linux, Solaris or Mac OS.)
- **Bamboo home directory** — This is the directory where your Bamboo configuration data and build results are stored. (The default location depends on your operating system: Windows, Unix/Linux, Solaris or Mac OS.) This directory can grow quite large when managing large quantities of plans and builds.

The most important contents of these two directories are described below.

**Bamboo home directory**

- `bamboo.cfg.xml` — This is Bamboo's core configuration file. It includes the configuration information for connecting to Bamboo's database.
- `xml-data/*` — This directory contains all files relating to source repositories and build results.
  - `xml-data/build-dir/*` — This is known as the **Working Directory**. This is where Bamboo temporarily puts the checked-out files it is building. The location of this directory was specified via the Setup Wizard, can be viewed as described in Viewing Bamboo's System Information, and can be changed as described in Specifying Bamboo's Working Directory.
  - `xml-data/builds/*` — This is known as the **Build Directory**. This is where Bamboo stores build results and artifacts (note that they will be deleted as described in Enabling Expiry of Build Results). The location of this directory was specified via the Setup Wizard, and can be viewed as described in Viewing Bamboo's System Information. Its contents can be backed up as per Exporting Data for Backup.
  - `xml-data/builds/PLAN_KEY/results` — Contains the build results for all the builds belonging to the 'PLAN_KEY' plan. Each build result is an individual XML file. Do not edit these files or the corresponding information in the database may become corrupt.
  - `xml-data/builds/PLAN_KEY/download-data` — Contains the artifacts and logs for each build belonging to the 'PLAN_KEY' plan.
- `xml-data/configuration/*` — This is known as the **Configuration Directory**. It contains server-wide configuration information. The location of this directory was specified via the Setup Wizard, and can be viewed as described in Viewing Bamboo's System Information. Its contents can be backed up as per Exporting Data for Backup.
- `database/*` — This directory contains Bamboo's embedded HSQL database. The database contains plan configurations and some build results data.
- `index/*` — This directory contains the build results index. Removing or modifying files in this directory may corrupt build history. Rebuilding the search index from Bamboo's global administration screen (see Optimising or Re-indexing Data) will completely regenerate the contents of this directory.

**Bamboo installation directory**

- `webapp/WEB-INF/classes/bamboo-init.properties` — This file tells Bamboo where to find the Bamboo home directory. The location of this directory is specified by the Bamboo administrator as described in the Bamboo Installation Guide, and can be viewed as described in Viewing Bamboo's System Information.
- `bamboo.sh`* — This is the startup file for Bamboo Standalone under Unix/Linux, Solaris and Mac OS.
- `bamboo.bat`* — This is the startup file for Bamboo Standalone under Windows.
- `bamboo.pid`* — This file, under Linux, contains the Process ID for the running instance of Bamboo.
- `conf/wrapper.conf`* — This file provides the means to configure Bamboo on startup, when using the Java Service wrapper under Linux or Windows.
- `scripts/*` — This directory contains operational scripts, including scripts for CVS and SVN triggers.
- `wrapper/*` — This directory contains the necessary files to start Bamboo using the Java Service wrapper (see the Mac and Linux installation guides).
- `logs/*` — This directory contains logs written by the Java Service wrapper. (Note: The Bamboo server logs are written to the root of the installation directory. Build logs are stored in the xml-data/builds/ sub-directories.)
- `webapp/*` — This directory contains all the Bamboo server application files.
- `webapp/WEB-INF/*` — This directory is used when deploying Bamboo plugins. It also contains other libraries required by Bamboo.
- `webapp/WEB-INF/classes/log4j.properties` — This is Bamboo's logging configuration file.
- `webapp/WEB-INF/classes/jetty.xml`* — This is the configuration file for Jetty, the application server that is bundled with Bamboo Standalone.
* This applies to the Bamboo Standalone distribution. The configuration may differ for the Bamboo EAR-WAR distribution.

**BAMBOO AGENT**

**Bamboo agent home directory**

When you installed your remote agents (if any), you specified the following directory:

- **Agent home directory** — This is the directory where the agent's configuration data is stored. The default name of this directory is `bamboo-agent-home`. (The default location depends on your operating system: Windows, Unix/Linux, Solaris or Mac OS.) This directory can grow quite large when managing large quantities of plans and builds.

The contents of the agent home directory are:

- `bamboo-agent.cfg.xml` — This contains configuration information about this remote agent. Most notably, it stores the agent id, which gets generated the first time this agent connects to the Bamboo server.
- `xml-data/`:
  - `xml-data/build-dir/` — This is where the agent will check out the files and perform builds (similar to the Bamboo server's directory)

**Specifying Bamboo's Working Directory**

The **working directory** is where Bamboo temporarily puts the checked-out files it is building. By default, this directory is located under the `xml-data` directory in the Bamboo home directory.

To change the location of Bamboo's working directory,

1. Shut down Bamboo.
2. Open the `<Bamboo-Home>/bamboo.cfg.xml` file in a text editor. Find the following line -

   ```xml
   <property name="buildWorkingDir"/>
   ```

3. Edit the Bamboo Working directory to point to a new folder on disk.
4. Save the changes and start Bamboo.
   Note: Bamboo will do a fresh checkout and perform a clean build of all your plans, once the directory is changed.

**Viewing your Database Connection Details**

When you installed Bamboo, you would have set up a database connection by following one of these processes:

- Connecting Bamboo to an external database
  - MySQL 4.1 and 5.0
  - Postgres 8+
  - Oracle 9i and 10g
  - Microsoft SQL Server
  - Unsupported databases
- Using Bamboo's embedded HSQL database

Once Bamboo is running, you can view the database configuration details as follows.

To view your database connection details,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Database Configuration' link in the left navigation column.

_Screenshot: 'Database Configuration'_
Moving your Bamboo Data to a different Database

When you installed Bamboo, you would have set up a database connection by following one of these processes:

- Connecting Bamboo to an external database
  - MySQL 4.1 and 5.0
  - Postgres 8+
  - Oracle 9i and 10g
  - Microsoft SQL Server
  - Unsupported databases
- Using Bamboo’s embedded HSQL database

You may later wish to use a different database. For example, the embedded HSQL database is suitable for evaluation purposes only — you would typically move to an external database before deploying Bamboo in production.

To move your Bamboo data to a different database,

1. Backup your Bamboo data as described in Exporting Data for Backup. Note the filename and path of the exported file for use in Step 8 below.
2. Shut down your old instance of Bamboo.
3. If your old instance of Bamboo was configured to start automatically (e.g. as a Windows service), disable it.
4. Install a new instance of Bamboo as described in the Bamboo Installation Guide. Specify a different Home Directory* and Installation Directory* from the directories used by your old instance of Bamboo. (If you use the same locations, your existing data will be deleted.)
5. Launch your new instance of Bamboo. You will see the Setup Wizard.
6. At Step 1 of the Setup Wizard, ensure that your new Configuration Directory*, Build Data Directory* and Build Working Directory* are in different locations to your old instance of Bamboo.
7. At Step 2 of the Setup Wizard, select your new database and follow the appropriate instructions for your chosen database:
   - MySQL 4.1 and 5.0
   - Postgres 8+
   - Oracle 9i and 10g
   - Microsoft SQL Server
   - Unsupported databases
8. At Step 3 of the Setup Wizard (see screenshot below), select Import existing data and specify the export file created in Step 1 above.
9. Wait while Bamboo imports your data. (You will not need to complete any more steps of the Setup Wizard.)
10. When the data import has finished, restart your new instance Bamboo.
11. Re-index your Bamboo data as described in Optimising or Re-indexing Data.
12. Verify that your build results and system settings look the same as before.

*For information about the contents of these directories, please see Locating Important Directories and Files

---

**Screenshot: Setup Wizard — Step 3: ‘Starting Data’**
Optimising or Re-indexing Data

About optimising
You may want to optimise your Bamboo build results data if you notice that search-intensive operations (e.g. reporting) are becoming slow.

Bamboo will still be accessible while the optimisation process is running.

About re-indexing
You will need to re-index your Bamboo build results data whenever you perform a data import. Re-indexing your data can also be helpful if your reports appear to be out-of-sync with your data.

Bamboo will not be accessible while the re-indexing process is running. This may take a few minutes to complete (see System Information for an estimate of how long it will take).

To re-index Bamboo’s build results data,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Indexing’ link in the left navigation column.
3. The ‘Optimise or Re-index Bamboo’ screen will appear. Select ‘Full re-index’ and click the ‘Perform’ button.

To optimise Bamboo’s build results data,

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Indexing’ link in the left navigation column.
3. The ‘Optimise or Re-index Bamboo’ screen will appear. Select ‘Optimise current index’ and click the ‘Perform’ button.

Enabling Expiry of Build Results

By enabling build expiry, you can choose how much build results data will be kept in your Bamboo system, and for how long it will be kept (e.g. for reporting purposes), before being automatically deleted.

If you disable build expiry, your build results will never be automatically deleted from Bamboo.

You can enable/disable build expiry for:

- **all plans** (as described below) — this is generally the easiest way to manage your build expiry. Your settings will apply to all plans that do not have individual expiry settings.
- **individual plans** (see Specifying Expiry for a Plan’s Build Results) — you would generally only do this if there is a specific reason to keep/delete a particular plan's build results.

Note that you can also expire build results manually — see Deleting a Build Result.
If you enable build expiry, ensure that you **back up** your build results data before its expiry date is reached.

On this page:

- Enabling Global Build Expiry
- Disabling Global Build Expiry

Enabling Global Build Expiry

To enable expiry for build results data,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Build Expiry' link in the left navigation column.
3. The 'Build Expiry' screen will be displayed. Click the 'Enable' button (note, this button will change to 'Edit' when build expiry settings have been added).
4. The 'Build Expiry Settings' screen will be displayed (see screenshot below).
5. In the 'Cron Expression' field, specify the cron expression which will govern when the build expiry event will be triggered. For example, specifying `0 0/30 9-19 ? * MON-FRI` signifies that the build expiry event will be triggered every half an hour from 9am to 7pm, Monday to Friday. See this FAQ for help constructing cron expressions.
6. In the 'Global Configuration' section, specify the criteria that will determine what build data is deleted. First, uncheck the 'Disable Global Build Expiry' checkbox to enable build expiry. The following fields will display:

    Please note, that the build expiry **event** (governed by the previously specified 'Cron Expression') will run regardless of whether you disable or enable build expiry of any data. For example, you may wish to disable build expiry globally, but still schedule a global build expiry event that triggers the deletion of build data from individual plans. See Specifying Expiry for a Plan's Build Results for details on how to override the global build expiry settings.

- 'What should be expired?': — select 'Artifacts' if you want to delete all user-defined artifacts but keep all other build results data. Select 'Build result' if you want to delete all build results data (including artifacts).
- Choose one of these three methods for specifying how much data to keep, using the 'Expiry period' and 'Minimum builds to keep' fields:
  - To keep all build results up to a certain age,
    - Please note, with this method, older plans could have **all** their build results deleted.
    - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your build results, e.g. specify '24 months' to keep all build results for the last two years.
    - In the 'Minimum builds to keep' field, specify '0'.
  - To keep a certain number of build results per plan,
    - In the 'Expiry period' field, specify '0'.
    - In the 'Minimum builds to keep' field, specify the number of build results you want to keep, e.g. specify '50' to keep the latest 50 build results for each plan.
  - To keep all build results up to a certain age, *and* a certain number of build results per plan,
    - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your build results data, e.g. specify '24 months' to keep all build results for the last two years.
    - In the 'Minimum builds to keep' field, specify the number of build results you want to keep, e.g. specify '50' to keep the latest 50 build results for each plan. (This means that, even if all of a plan's builds are over two years old, the last 50 build results will not be deleted.)
- If you wish to keep builds with particular labels, check the 'Don't expire builds with certain labels' checkbox. The 'Labels to keep' field will display:
  - In the 'Labels to keep' field, specify any labels for which you always want to keep labelled builds. (If you want to specify more than one label, use spaces to separate them.) For any label(s) that you specify, all builds that have a matching label will never be deleted, regardless of which of the three methods you followed for keeping and expiring data above.

    Please note, builds can either be labelled:
    - manually, as described in Labelling a Build Result in the Bamboo User's Guide; or
    - automatically, as described in Specifying Labels for a Plan's Build Results in the Bamboo Administrator's Guide.

7. Click the 'Save' button.
Disabling Global Build Expiry

To disable expiry for build results data,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Build Expiry' link in the left navigation column.
3. The 'Build Expiry' screen will be displayed (see screenshot below). Check the 'Disable Global Build Expiry' checkbox.

Please note, that the build expiry event (governed by the previously specified 'Cron Expression') will run regardless of whether you disable or enable build expiry of any data. For example, you may wish to disable build expiry globally, but still schedule a global build expiry event that triggers the deletion of build data from individual plans. See Specifying Expiry for a Plan's Build Results for details on how to override the global build expiry settings.
Specifying a Backup Schedule

You can configure Bamboo to automatically create a backup each night, rather than doing a manual export every time.

Before you begin:

- Bamboo will be unavailable while the backup process completes. Depending on the number of builds and tests, the export may take a long time to complete and may require large amounts of disk space. Please make sure you have enough disk space before proceeding.
- Bamboo will not export if it detects that plans are currently being built. Please ensure that you do not have plans building (see Viewing Bamboo’s Current Activity).
- There is a bug in Bamboo 1.2 which causes the dependency associations to not be exported along with the plans. If you are using Bamboo 1.2, please upgrade to Bamboo 1.2.1 for a fix.

To specify a backup schedule,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Scheduled Backups' link in the left navigation column.
3. The 'Scheduled Backups' screen will be displayed Click the 'Edit' button.
4. In the 'Backup path' field, specify the directory where you want to store your backups. Each backup will be stored as a single file.
   - Note that backups may require large amounts of disk space. Please make sure you have enough disk space before proceeding.
5. In the 'Backup file prefix' field, specify the first part of the filename for all your backup files.
6. In the 'Backup file date pattern' field, specify the date/time format for identifying your individual backup files. This will be appended to the 'Backup file prefix' to form the complete filename for your backup files, e.g.:
   - If your 'Backup file prefix' is 'bamboo_backup_' and your 'Backup file date pattern' is 'yyyy_MM_dd', then your backup file on 31st July 2007 would be named 'bamboo_backup_2007_07_31'.
7. In the 'Backup Frequency' field, specify how often (in days) you want Bamboo to create a backup, e.g.:
   - Specify '1' to create a backup every day.
   - Specify '7' to create a backup every week.
8. In the 'Backup Time' field, specify the time of day at which you want Bamboo to create a backup. Use 24-hour format, e.g. to create a backup at 8:30 PM you would specify '20:30'.
   - Note that Bamboo will be unavailable for the duration of the backup, so choose a time of day or night when usage is low.
9. Select the 'Backup Artifacts' check-box if you want to backup your build artifacts.
10. Click the 'Save' button.

Your first backup will run when your server’s clock matches the time specified in the 'Backup Time' field.

To disable a scheduled backup,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Scheduled Backups' link in the left navigation column.
3. The 'Scheduled Backups' screen will be displayed, showing your current backup schedule. Click the 'Edit' button.
4. Select the 'Disable scheduled backups' check-box.
   - Your schedule details will be retained, but no automatic backups will be performed.
5. Click the 'Save' button.

Exporting Data for Backup

- Bamboo will be unavailable while the export process completes. Depending on the number of builds and tests, the export may take a long time to complete and may require large amounts of disk space. Please make sure you have enough disk space before proceeding.
- Bamboo will not export if it detects that plans are currently being built. Please ensure that you do not have plans building (see Viewing Bamboo's Current Activity).
- There is a bug in Bamboo 1.2 which causes the dependency associations to not be exported along with the plans. If you are using Bamboo 1.2, please upgrade to Bamboo 1.2.1 for a fix.
- Bamboo uses the third party TrueZip library to create zip archives. TrueZIP currently implements the ZIP32 specification only. This limits the maximum ZIP file length to 4GB. Unfortunately, Bamboo exports will fail if the resulting ZIP file is over 4GB. As a workaround, please export Bamboo without artifacts or use a different backup strategy.
- Exporting Bamboo doesn't work with Bamboo versions prior to 1.2.2, if LDAP is turned on. If you want to export Bamboo data, please turn off LDAP. Please see Integrating Bamboo with LDAP for further details.

To export data for backup,
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Export' link in the left navigation column.
3. Type the absolute 'File Path' to which Bamboo is to export data. For example, 'c:/Documents and Settings/<me>/bamboo/bamboohome/manual_backups/export.zip'.
4. Select the 'Export Artifacts' check-box if you want to backup your build artifacts.
5. Click the 'Export' button.

**Importing Data from Backup**

**Before you begin**
- The import process will DELETE this instance and restore data from a previous export of Bamboo. This includes login data, hence you will need an administration login that is contained in the Bamboo data to be imported.
- Bamboo will be unavailable until the import process is complete, which may take some time.
- At the moment bamboo is not able to import instance zip files > 200MB on 32 bit JVMs. A 64 bit JVM with max heap space of 4G (-Xmx4096m) can handle the import of such an instance.

To import data from backup,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Import' link in the left navigation column.
3. Type the absolute 'File Path' from which Bamboo is to import data. For example, '/opt/bamboo/bamboohome/export.zip'.
4. Select the 'Backup Data' check-box (HIGHLY RECOMMENDED).
5. Specify the absolute 'File path of backup' to which Bamboo should backup data (note that this must be different from the 'File Path' above). For example, '/opt/bamboo/bamboohome/backup.zip'.
6. Click the 'Import' button.
7. After the import is complete,
   - check the paths of your builders and JDK.
   - index your data.
Configuring System Settings

- Configuring Logging in Bamboo
- Configuring System Properties
- Enabling Bamboo's Remote API
- Enabling GZIP Compression
- Specifying Bamboo's Title
- Specifying Bamboo's URL
- Updating your Bamboo License Details
- Viewing Bamboo's System Information

Viewing Bamboo's System Information

When you installed Bamboo, you provided information about how the system should be configured.

To view your Bamboo system information,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'System Info' link in the left navigation column.

This page contains useful information for you to send to Atlassian when requesting support.
## System Information

### System Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Date</td>
<td>Thursday, 01 Nov 2007</td>
</tr>
<tr>
<td>System Time</td>
<td>2150:37</td>
</tr>
<tr>
<td>Username</td>
<td>j2ee</td>
</tr>
<tr>
<td>User Timezone</td>
<td>America/Chicago</td>
</tr>
<tr>
<td>User Locale</td>
<td>English (United States)</td>
</tr>
<tr>
<td>System Encoding</td>
<td>ANSI_X3.4-1968</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux 2.6.9-55.0.9.ELsmp</td>
</tr>
<tr>
<td>Operating System</td>
<td>i386</td>
</tr>
<tr>
<td>Available Processors</td>
<td>4</td>
</tr>
<tr>
<td>Application Server</td>
<td>Apache Tomcat/5.5.20</td>
</tr>
</tbody>
</table>

### Java Runtime Information

| Java Version           | 1.5.0_09                                   |
| Java Vendor            | Sun Microsystems Inc.                      |
| JVM Version            | 1.0                                        |
| JVM Vendor             | Sun Microsystems Inc.                      |
| JVM Implementation     | 1.5.0_09-b01                               |
| Java Runtime           | Java(TM) 2 Runtime Environment, Standard Edition |
| Java VM                | Java HotSpot(TM) Server VM                 |

### Java VM Memory Statistics

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Memory</td>
<td>1416 MB</td>
</tr>
<tr>
<td>Free Memory</td>
<td>175 MB</td>
</tr>
<tr>
<td>Used Memory</td>
<td>1239 MB</td>
</tr>
</tbody>
</table>

### Bamboo Paths

- Current running directory: `/opt/j2ee/domains/bamboo.atlassian.com/jira/apache-tomcat-5.5.20`
- Configuration Path: `/home/j2ee/bamboo/ml-data/configuration`
- Build Path: `/home/j2ee/bamboo/ml-data/builds`
- Build Working Directory: `/home/j2ee/bamboo/ml-data/build-dir`
- Bamboo Home: `/home/j2ee/bamboo`

### File System Statistics

- Disk space free (for bamboo.home): 163 GB
- Size of index: 2 GB

### Instance Statistics

- Number of plans: 43
- Number of builds (approx): 26928
- Number of tests (approx): 33762965
- Full reindex time (approx): 1 day, 12 hours, 59 minutes

### Environment Variables

- `DOMAIN=bamboo.atlassian.com`
- `DOMAINS_DIR=opt/j2ee/domains`
- `HOST=jira`
For information about the 'Bamboo Paths' please see Locating Important Directories and Files.

Updating your Bamboo License Details

When you upgrade or renew your Bamboo license, you will receive a new license key. You will need to update your Bamboo server with the new license key.

To update your Bamboo license key,

1. Log into Bamboo 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.
4. This will display your existing Bamboo license details, and an empty box called 'License Key'. Paste your new license into this box.
5. Click the 'Save New License' button.

Licensing Questions?
Please see the Licensing FAQ.

Specifying Bamboo’s Title

Bamboo's name is the displayed title of this installation of Bamboo. It will appear throughout Bamboo (e.g. on the Dashboard), and in the window-title of your users' browsers.

To specify Bamboo’s title,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'General Configuration' link in the left navigation column.
3. This will display the 'General Configuration' page. In the 'Name' field, type the display title for your Bamboo server (e.g. "MyCompany's Bamboo").
4. Click the 'Save' button.

Specifying Bamboo’s URL

This is the base URL of this installation of Bamboo. All links created (for emails etc) will be prefixed by this URL.

To specify Bamboo’s URL,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'General Configuration' link in the left navigation column.
3. This will display the 'General Configuration' page. In the 'Base URL' field, type the URL address of your Bamboo server (for example, "http://keg:8080/bamboo").
4. Click the 'Save' button.
Configuring Logging in Bamboo

Bamboo generates two distinct sets of logs:

- **Build Logs**: The build logs are generated each time a plan is executed. All information specific to the build is stored in these logs. The build logs are located in the `<Bamboo-Home>/xml-data/builds/` sub-directories. The build logs can be downloaded as an artifact (see Viewing a Build's Artifacts).

- **atlassian-bamboo Logs**:
  - **atlassian-bamboo Logs for the Bamboo Server** — Bamboo records all server activity in the `atlassian-bamboo.log`. The location of the `atlassian-bamboo.log` file can be viewed in Bamboo's System Information under the 'Bamboo Paths' section. The location will generally be either the root `<Bamboo-Install>` directory or the directory you started Bamboo from. In case of a Tomcat webapp deployment, the logs are piped out to `catalina.out` file.
    - Please note, this log is different to the `bamboo.log` found in the `<Bamboo-Install>/logs` directory, which is the log written by the Java Service wrapper.
  - **atlassian-bamboo Logs for Remote Agents** — All agent activity is recorded in `atlassian-bamboo.log` file stored on the agent machine. These are generated in the running directory of the agent. The running directory can be viewed in the remote agent's system properties) under the 'Bamboo Paths' section.

See Locating Important Directories and Files for information on where to find other important files in Bamboo.

The information below relates to the Bamboo server (atlassian-bamboo) logs only. You cannot change the logging configuration for the build logs.

On this page:

- Configuring the Level of Logging
  - Configuring the Level of Logging on your Bamboo Server
  - Configuring the Level of Logging on your Bamboo Remote Agents
  - Configuring the Location of the atlassian-bamboo Logs

**Configuring the Level of Logging**

Bamboo uses the log4j library for logging during runtime. The logging levels can be changed by editing the `<Bamboo-Install>/webapp/WEB-INF/classes/log4j.properties` file. There are five logging levels available: 'DEBUG', 'INFO', 'WARN', 'ERROR' and 'FATAL'. Each logging level provides more logging information that the level before it:

```
DEBUG < INFO < WARN < ERROR < FATAL
```

i.e. DEBUG provides the most verbose logging and FATAL provides the least verbose logging.

**Configuring the Level of Logging on your Bamboo Server**

You can adjust the logging levels for the different Bamboo packages on the fly, using the runtime log4j configuration tool in the Bamboo administration console. The default log settings are still stored in the log4j.properties file. When you view the log settings page for the first time you will see the default log settings as defined in log4j.properties. All changes to the log settings via the runtime log4j configuration tool will not be persisted and are valid during bamboo runtime only.

**To change the level of logging on your Bamboo server,**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Log Settings' link in the left navigation column under the 'System' section. The 'Bamboo Log Settings' page will display showing the Bamboo packages being logged (see screenshot below).
   - To change the logging level of a package that is already being logged, locate the Bamboo package, select the desired logging level from the dropdown next to it and click 'Save'.
   - To start the monitoring a package in the Bamboo logs, enter the class name in the text box at the top of the page, select the desired logging level from the dropdown next to it and click 'Add'.
   - To stop logging a package, locate the Bamboo package and click the 'Delete' link next to it.

**Note**, you do not need to restart your Bamboo server for the logging changes to take effect.

*Screenshot: Bamboo Log Settings*
Configuring the Level of Logging on your Bamboo Remote Agents

The runtime log4j configuration tool in the Bamboo administration console can only be used to modify the logging levels for the Bamboo server. To configure the logging levels for your remote agents, you will need to update the `log4j.properties` file manually.

You can control the logging for each of your remote agents separately from the Bamboo server. To do this, simply repeat the process described below for multiple remote agents, so that each remote agent has a `log4j.properties` file that overrides the `log4j.properties` file on the Bamboo server.

To change the level of logging on your remote agent,

1. Configure a `log4j.properties` file for your remote agent. This can be any `log4j.properties` file. If you do not already have a `log4j.properties` file, you can take a copy of the `log4j.properties` file from the server, copy it to your remote agent and configure it as desired:
   - The rootLogger property in the `log4j.properties` file controls the verbosity of logs being generated at the top level. By default, the root level logging is set to 'INFO'. To change the root level logging, find the following lines in `<Bamboo-Install>/webapp/WEB-INF/classes/log4j.properties` and update the value of `log4j.rootLogger` to the desired logging level:

   ```
   # Change the following line to configure the bamboo logging levels (one of INFO, DEBUG, ERROR, FATAL)
   log4j.rootLogger=INFO, console, filelog
   #
   # Modify the logging level for any of the individual packages in the log4j.properties as desired, e.g.
   # log4j.category.webwork=WARN
   #
   # Save changes to the file.
   #
   2. Save changes to the file.
   #
   3. Update the log4j.configuration system property on your remote agent to point to the log4j.properties file. To do this, add the following command line parameter when starting your remote agent:
   -Dlog4j.configuration=/path/to/log4j.file
   where /path/to/log4j.file is the absolute path of your log4j.properties file. You can also update this setting in the bamboo-agent.cfg.xml file.
   #
   4. Restart your remote agent.

Configuring the Location of the atlassian-bamboo Logs

---

**Bamboo Log Settings**

On the fly switch between Production and Debug Logging. The settings are predefined in log4j.properties

**Add a Class to Log4j**

Add a new class and level to be included in the bamboo log

<table>
<thead>
<tr>
<th>Package</th>
<th>Current Level</th>
<th>New Level</th>
<th>Save</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.atlassian.bamboo.agent</td>
<td>DEBUG</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.atlassian.bamboo.buildqueue</td>
<td>DEBUG</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.atlassian.bamboo.buildqueue.manager.RemoteAgentManagerImpl</td>
<td>DEBUG</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.atlassian.bamboo.filter.AccessLogFilter</td>
<td>DEBUG</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.atlassian.bamboo.user.robobamboo.UserManagerImpl</td>
<td>WARN</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.opensymphony.xwork.util.LocalizedTextUtil</td>
<td>ERROR</td>
<td>ALL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[End of document]
To change the directory that the `atlassian-bamboo` logs are generated to, you must set the environment variable for the target location of the logs, as seen below:

```
log4j.appender.fileLog.file=/my/path/to/atlassian-bamboo.log
```

Important note: The new log file location applies to both the server and remote agents. If using an absolute path this may result in aggregated logs.

### Enabling GZIP Compression

You can enable GZIP compression in order to reduce the size of Bamboo's web pages. This is useful if Bamboo is being run over slow networks. There is a slight performance penalty, and note that GZIP may not work for languages other than English.

**To enable GZIP Compression,**

1. Click the `Administration` link in the top navigation bar.
2. Click the `General Configuration` link in the left navigation column.
3. Select the `Apply gzip compression to reduce the size of Bamboo’s web pages?` check-box.
4. Click the `Save` button.

### Enabling Bamboo's Remote API

The Bamboo remote API described in Bamboo Remote API has been deprecated in favour of the new Bamboo REST APIs. You only need to enable the `Accept remote API calls?` option if you want to allow access to the deprecated remote API. Access to the new REST API is enabled by default.

You access Bamboo's data from an external program by using Bamboo's REST-style remote API. The remote API is disabled by default. Bamboo will return an error if people try to use the remote API when it is disabled.

**To enable the remote API,**

1. Click the `Administration` link in the top navigation bar.
2. Click the `General Configuration` link in the left navigation column.
3. Select the `Accept remote API calls?` check-box.
4. Click the `Save` button. Bamboo will now accept remote calls. You do not have to restart the Bamboo server.

Looking for more API documentation? For documentation on the deprecated Bamboo remote API, see Bamboo Remote API. For documentation on the new Bamboo REST API, see Bamboo REST APIs.

### Configuring System Properties

The default settings on a number of Bamboo functions can be configured by setting the appropriate system properties. This page provides general instructions on how to set a system property in Bamboo.

You can use either of the two methods described below to configure a system property in Bamboo:

- Configuring a system property by command-line parameter
- Configuring a system property in `bamboo.cfg.xml`
Configuring a system property by command-line parameter

To configure a system property by command-line parameter,

- Add the parameter with a `-D` prefix and appropriate value, in your command line when starting Bamboo. For example, if you wanted to set the `bamboo.agent.heartbeatInterval` system property to 10 (seconds), you would add `-Dbamboo.agent.heartbeatInterval=10` as a command-line parameter.

Configuring a system property in bamboo.cfg.xml

To configure a system property in bamboo.cfg.xml,

- Add the parameter `bamboo.agent.heartbeatInterval` as a property in your `<bamboo-home>/bamboo.cfg.xml` file. For example, if you wanted to set the `bamboo.agent.heartbeatInterval` system property to 10 (seconds), you would add `<property name="bamboo.agent.heartbeatInterval">10</property>` to your `bamboo.cfg.xml` file.

Managing Bamboo Security

As a distributed application, Bamboo's security is important. This page contains links to security-related information in the Bamboo documentation.

**Security Advisories**

For information on how to report a security vulnerability in Bamboo and our policy on security advisories and patches, please read Bamboo Security Advisories. A full list of security advisories that we have previously issued is also available on that page.

**Bamboo Permissions**

For information on Bamboo's internal security model, i.e. user management and permissions, please see Managing Users and Permissions.

**Bamboo Configuration**

The following pages contain information on how to configure Bamboo features that can permit/forbid access to the Bamboo application. Please read each page carefully for further information.

- Configuring OAuth Consumers
- Elastic Bamboo Security

**Configuring OAuth Consumers**

An OAuth ‘consumer’ is an application that accesses (‘consumes’) data from another application. When you add an OAuth consumer in Bamboo, you are allowing the consumer application to access Bamboo’s data. For example, if you want your users to be able to add Bamboo 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.

Please see the information below for instructions on adding and removing OAuth consumer information for Bamboo. Additional information is also available in our Gadgets and Dashboards documentation.

On this page:

- Adding an OAuth consumer
- Removing an OAuth consumer

**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 'Admin' global permission.
2. Click the 'Administration' link in the top navigation bar.
3. In the left-hand menu, under the title 'Security', click the 'OAuth Consumers' link. The 'OAuth Administration' page will be displayed, showing a list of configured OAuth Consumers (if any exist).
4. Click the 'Add OAuth Consumer' link at the bottom of the list. The 'Add Consumer' page will be displayed:

   Screenshot: Adding an OAuth Consumer (click to view larger image)

5. You can either:
   - If the consumer is another Atlassian application (e.g. JIRA), type the consumer's root URL in the 'Consumer Base URL' field (e.g. "http://jira.mycompany.com"), so that Bamboo can automatically look up the consumer's details for you;
   - OR:
     - Enter the consumer's details yourself:
       a. 'Consumer Key' — Type the consumer's unique identifier (e.g. for iGoogle, type "www.google.com").
       b. 'Name' — Type a short name that is meaningful to you and your end-users (e.g. "iGoogle").
       c. 'Description' (optional) — Type a longer description if you wish.
       d. '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.
       e. 'Callback URL' (optional) — Type the URL of the page that Bamboo 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. You can edit any of these fields (apart from the 'Consumer Key') after the consumer has been added.

Removing an OAuth consumer

To remove an OAuth consumer,

1. Log in as a user with the 'Admin' global permission.
2. Click the 'Administration' link in the top navigation bar.
3. In the left-hand menu, under the title 'Security', click the 'OAuth Consumers' link. The 'OAuth Administration' page will be displayed, showing a list of configured OAuth Consumers (if any exist).
4. Locate the consumer that you wish to remove and click the 'Remove' link next to it. A confirmation message will display.
5. Confirm the removal of the consumer. Any request tokens created by this consumer application will be removed and the application will no longer be able to access Bamboo's data/resources.

Elastic Bamboo Security

Elastic Bamboo is a feature in Bamboo that allows Bamboo to dynamically source computing resources from the Amazon Elastic Compute Cloud (EC2). If you choose to enable Elastic Bamboo, the broker port (port 54663 by default) of your Bamboo server must be made available to remote agent instances created in the EC2.

Please be warned that this can expose your Bamboo installation to number of security vulnerabilities, if any of your remote agent instances are compromised. These include confidential data (e.g. source code, VCS credentials) being stolen, malicious code being injected into elastic agents, unauthorised access to build queues and false information being submitted to Bamboo servers.

To mitigate some of these security risks, Elastic Bamboo incorporates an SSH tunnelling implementation to provide a secure communication channel between your Bamboo server and the EC2. This tunnelling implementation encrypts traffic between the Bamboo server and elastic agents using SSL, which means that you do not need to compromise your firewall by opening it up to outside connections.

SSH tunnelling is not implemented for VCS (Version Control System) to EC2 traffic though. You will need to make your VCS available to the EC2 to use Elastic Bamboo. Please see the section on setting up your VCS for Elastic Bamboo, which contains guidelines on securing your VCS.

Screenshot: Elastic Bamboo Security Architecture
The sections below explain the default access rules for remote agent instances and how to change these rules, if desired.

**Default EC2 Access Rules**

When you first use Elastic Bamboo, i.e. start an elastic instance, an 'elasticbamboo’ security group will be set up for you on your AWS account. This security group is essentially a set of IP addresses that are permitted access to the EC2. By default, the security group will contain two rules — one to allow connections for Elastic Bamboo itself, and another to allow connections via SSH.

The EC2 security groups can be accessed via the AWS management console (see ‘Security Groups’ in the left-hand menu under ‘Configuration’).

**Screenshot: AWS Console - Security Groups**

![AWS Console - Security Groups](image)

**Changing the Default EC2 Access Rules**

If you wish to change the default access rules for Elastic Bamboo (e.g. remove SSH access, permit additional connections), you can do this by adding or removing entries from the ‘Allowed Connections’ for the ‘elasticbamboo’ security group. See the previous section on ‘Default EC2 Access Rules’ for instructions on how to access your EC2 security groups.

**Setting up your VCS for Elastic Bamboo**

We recommend that you take the following steps to ensure that your VCS is set up securely for Elastic Bamboo:

1. Make your VCS accessible to the public internet
2. Configure your AWS security group
3. Use VCS authentication and access control
4. Use encrypted connections to VCS

1. **Make your VCS accessible to the public internet**

As SSH tunnelling is not implemented for VCS to EC2 connections, you will need to make your VCS accessible to the public internet to use Elastic Bamboo. If your VCS is behind a firewall this will involve configuring an access point in your firewall. Please consult the documentation for your firewall software for details on how to do this.

2. **Configure your AWS security group**
Once you have made your VCS available to the public internet, the next step is to allow your VCS to connect to EC2. This involves adding the necessary access rule to the 'elasticbamboo' security group in your AWS account, to allow a connection from your VCS. Please see the section on *Changing the Default EC2 Access Rules* above for instructions on how to do this.

### 3. Use VCS authentication and access control

As you have made your VCS available to the public internet, we highly recommend that you secure access to your VCS by enabling the authentication and access control features on your VCS. The instructions for doing this vary from VCS to VCS. Please consult the documentation for your VCS for details.

### 4. Use encrypted connections to VCS

We also highly recommend that you use encrypted connections for your VCS (e.g. SSL). Again, the instructions for doing this vary from VCS to VCS. Please consult the documentation for your VCS for details.

---

## Configuring Plugins

**About Bamboo Plugins**

A Bamboo plugin is a program that provides a piece of Bamboo functionality. Bamboo comes with some preinstalled plugins, including:

- 'AutoFavourites' plugin
- 'JIRA' plugin
- 'NAnt Builder' plugin
- 'Clover' plugin

Additional plugins can be created (see the Bamboo Plugin Guide) or obtained from the library, and installed into your Bamboo system.

### Enabling the 'Auto-Favourites' Plugin

If the 'Auto-Favourites' plugin is enabled, then a plan will be automatically added to a user's 'My Bamboo' tab when the user checks-in code to the plan's source-code repository, and the user will become a 'watcher' of that plan (i.e. depending on how the plan's notifications are configured, they may receive notifications about the plan's build results).

To enable the 'Auto-Favourites' plugin,

1. Click the 'Administration' link in the top navigation bar.
2. Click the link 'Configure Auto-Favourites Plugin' in the left-hand column.
3. Tick the box 'Enable Auto-Favourite Plugin'.
4. Click the 'Save' button.

Each plan will now be automatically added to a user's favourites the first time the user checks-in code to the plan's source-code repository. Note that, if the user removes the plan from their favourites, it will not be automatically added again.

### Enabling the Clover Plugin

This page contains instructions on enabling and configuring the Clover plugin for a build plan in Bamboo.
One-click Clover integration for Ant will not work on Windows

We are aware of an issue with Ant that prevents one-click Clover integration (i.e. the 'Automatically integrate Clover into this build' option described below) from working on Windows (see BAM-4638 for further details). If you wish to use Clover with Ant on Windows, you will need to set it up manually as described in this FAQ.

To enable the 'Clover' plugin on a plan,

1. Go to the plan.
2. Click the 'Configuration' tab, then the 'Builder Configuration' sub-tab, then click 'Edit Plan'.

   ✪ You will need Administrator rights to do this.
3. Tick the option 'Use Clover to collect Code Coverage for this build' and fill out the following fields;
   - **Integration Options** — Select one of the following options:
     - 'Automatically integrate Clover into this build' — For this option, you have two sub options; 'Generate a Clover Historical Report' and 'Generate a JSON report'. The Clover Historical Report shows the current coverage results compared with previous Clover code coverage reports. The JSON report gives the Clover results in a format ready for embedding into applications or external report views.
     - 'Clover is already integrated into this build and a clover.xml file will be produced' — Use this option when you already have Clover-for-Ant or Clover-for-Maven configured to generate a report. You will also need to specify where the Clover XML report is being generated, under 'Clover XML Location'. For this, specify the name of the directory (including path) where Bamboo will look for the XML report output file from Clover. Please specify a file path relative to your plan's root directory, for example:


```
target/site/clover/clover.xml
```

   ✪ Do not specify an absolute path.

   - 'Clover XML Location' — Specify the location where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g. `/home/bamboouser.bamboo-home/xml-data/build-dir/MY_PLAN/`), i.e. please do not specify an absolute path.
4. Click the 'Save' button.

Screenshot: Enabling Clover on Your Plan

<table>
<thead>
<tr>
<th>Would you like to view Clover Code Coverage for this plan?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com" alt="Use Clover to collect Code Coverage for this build." /></td>
</tr>
<tr>
<td>Clover is a code coverage tool reports how well tested your code is and also highlights parts of code that require more testing. For more information visit <a href="https://www.atlassian.com/software/clover">Atlassian Clover</a></td>
</tr>
<tr>
<td><img src="https://example.com" alt="Automatically integrate Clover into this build." /></td>
</tr>
<tr>
<td>Clover Historical Report and JSON report give Clover results in a format ready for embedding into applications or external report views. For more information visit <a href="https://www.atlassian.com/software/clover">Atlassian Clover</a> and view the Online Documentation.</td>
</tr>
<tr>
<td><img src="https://example.com" alt="Generate a Clover Historical Report" /></td>
</tr>
<tr>
<td>(Will only work consistently if this plan is run on a single agent and no clean checkout is performed.)</td>
</tr>
<tr>
<td><img src="https://example.com" alt="Generate a JSON report" /></td>
</tr>
<tr>
<td>JSON makes it very easy to integrate Clover data into web-page. <a href="https://example.com">Learn how.</a></td>
</tr>
</tbody>
</table>

Clover License:

Specify your license, or get a [free 30 day trial](https://example.com)

Screenshot: Pointing Bamboo to Existing Clover Reports

<table>
<thead>
<tr>
<th>Would you like to view Clover Code Coverage for this plan?</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com" alt="Use Clover to collect Code Coverage for this build." /></td>
</tr>
<tr>
<td>Clover is a code coverage tool reports how well tested your code is and also highlights parts of code that require more testing. For more information visit <a href="https://www.atlassian.com/software/clover">Atlassian Clover</a></td>
</tr>
<tr>
<td><img src="https://example.com" alt="Automatically integrate Clover into this build." /></td>
</tr>
<tr>
<td>Clover Historical Report and JSON report give Clover results in a format ready for embedding into applications or external report views. For more information visit <a href="https://www.atlassian.com/software/clover">Atlassian Clover</a> and view the Online Documentation.</td>
</tr>
<tr>
<td>Clover XML Location:</td>
</tr>
<tr>
<td><img src="https://example.com" alt="target/site/clover/clover.xml" /></td>
</tr>
</tbody>
</table>

This is where Bamboo will look for the XML report output file from Clover. The Clover report generation must run as part of your build.

Related Links
Installing a new Plugin

To install a new plugin,

1. Copy the plugin (i.e. JAR file) into the following directory:
   - If you are using Bamboo Standalone: ../<Bamboo-install>/webapp/WEB-INF/lib/
   - If you are using the Bamboo EAR-WAR distribution: ../<Bamboo-Deploy-Location>/WEB-INF/lib/
   (e.g. if you are running Bamboo EAR-WAR on Tomcat, copy your plugin to the
     ../<Tomcat-Install>/webapps/<Bamboo-Deploy-Location>/WEB-INF/lib/ directory).
2. Restart Bamboo.

A number of plugins are available from the library. You can also create your own as described in the Bamboo Plugin Guide.

Using Bamboo with Other Applications

Using Bamboo with Other Applications

- Embedding Bamboo into Other Applications
  - Javascript Widgets
    - All Plans & My Favourite Plans
    - Latest Builds
    - Latest Status of a Plan
    - My Latest Changes
    - Plan Summary Graphs
- Integrating Bamboo with Other Applications
  - Integrating Bamboo with JIRA

Embedding Bamboo into Other Applications

- Javascript Widgets

Javascript Widgets

Bamboo has a number of widgets which can be used by external applications:

- All Plans & My Favourite Plans
- Latest Builds
- Latest Status of a Plan
- My Latest Changes
- Plan Summary Graphs

All Plans & My Favourite Plans

These widgets retrieve a summary of plans for a particular Bamboo instance. The summary is shown as a list and depicts the current status, the last completed builds and the reason for the last build. You can either show all the plans or just those that are in your favourites list (username and password required).

To use this widget

1. Include the style sheet in your html document

   <link rel="stylesheet" type="text/css" href="<bamboo-base-url>/styles/bamboo-widget.css" />

2. Place the following script tag in your html
For all plans

```html
<script type="text/javascript"
src="<bamboo-base-url>/js/jsBuildSummaryAll.action" >
</script>
```

For your favourite plans

```html
<script type="text/javascript"
src="<bamboo-base-url>/js/jsBuildSummaryFavourites.action?os_username=<your-user-name>&os_password=<your-password>"
</script>
```

3. Replace `<bamboo-base-url>` with the base url for your bamboo instance.

4. Replace `<your-user-name>` and `<your-password>` with the appropriate values.

5. **Style!** The style sheet provided just gives some basic style definitions. You can override these definitions to customise the widgets to suit your needs.

**Example**

Live example from http://opensource.bamboo.atlassian.com/

```html
<link rel="stylesheet" type="text/css" href="http://opensource.bamboo.atlassian.com/styles/bamboo-widget.css">
<script type="text/javascript" src="http://opensource.bamboo.atlassian.com/js/jsBuildSummaryAll.action"></script>
```

**Latest Builds**

This widget produces a list of the last 15 completed builds. A summary is provided for each outlining the build number, reason for the build, date, duration and test results.

**To use this widget**

1. Include the style sheet in your html document

   ```html
   <link rel="stylesheet" type="text/css" href="<bamboo-base-url>/styles/bamboo-widget.css" >
   ```

2. Place the following script tag in your html

   ```html
   <script type="text/javascript"
   src="<bamboo-base-url>/js/showRecentlyCompleted.action">
   </script>
   ```

3. Replace `<bamboo-base-url>` with the base url for your bamboo instance.

4. **Style!** The style sheet provided just gives some basic style definitions. You can override these definitions to customise the widgets to suit your needs.

**Example**

Live example from http://opensource.bamboo.atlassian.com/

```html
<link rel="stylesheet" type="text/css" href="http://opensource.bamboo.atlassian.com/styles/bamboo-widget.css">
<script type="text/javascript" src="http://opensource.bamboo.atlassian.com/js/showRecentlyCompleted.action"></script>
```
2. Place the following script tag in your HTML:

```html
<script type="text/javascript" src="<bamboo-base-url>/js/showLatestBuildStatus.action?buildKey=<plan-key>"></script>
```

3. Replace `<bamboo-base-url>` with the base url for your bamboo instance.

4. Replace `<plan-key>` with the key of the plan you want to summarise. 
   eg. TEST-DEF

5. Style - The style sheet provided just gives some basic style definitions. You can override these definitions to customise the widgets to suit your needs.

   Example

Live example from [http://opensource.bamboo.atlassian.com/](http://opensource.bamboo.atlassian.com/)

My Latest Changes

This widget allows you to view a list of your 10 most recent changes. It provides details of the changes you made (including the commit comments and links to related JIRA issues) as well as details of the build the change was included in (success or failure, how long ago and test results).

To use this widget

1. Include the style sheet in your HTML document

   ```html
   <link rel="stylesheet" type="text/css" href="<bamboo-base-url>/styles/bamboo-widget.css" />
   ```

2. Place the following script tag in your HTML

   ```html
   <script type="text/javascript" src="<bamboo-base-url>/js/myChanges.action?os_username=<your-user-name>&os_password=<your-password>"></script>
   ```

3. Replace `<bamboo-base-url>` with the base url for your bamboo instance.

4. Replace `<your-user-name>` and `<your-password>` with the appropriate values.

5. Style!! - The style sheet provided just gives some basic style definitions. You can override these definitions to customise the widgets to suit your needs.

   Example

```html
<link rel="stylesheet" type="text/css" href="http://localhost:8085/styles/bamboo-widget.css" />
<script type="text/javascript" src="http://localhost:8085/js/myChanges.action?os_username=admin&os_password=admin"></script>
```
Plan Summary Graphs

These widgets allows you to view either of the two summary graphs displayed on the plan summary page. The two graphs are:

- Build Duration & Number of Failures per Build
- Successful Builds & Average Duration Per Time Period

To use this widget

1. Place the following script tag in your html

   For Build Duration & Number of Failures per Build
   
   ```html
   <script type="text/javascript"
   src="<bamboo-base-url>/js/viewCombinedByBuildNumberChart.action?buildKey=<Plan-Key>&filterController.selectedFilterKey=<filter-key>"
   
   </script>
   ```

   For Successful Builds & Average Duration Per Time Period
   
   ```html
   <script type="text/javascript"
   src="<bamboo-base-url>/js/jsViewCombinedByTimePeriodChart.action?buildKey=<Plan-Key>&filterController.selectedFilterKey=<filter-key>"
   
   </script>
   ```

2. Replace <plan-key> with the key of the plan you want to summarise.
   eg. TEST-DEF

3. Replace <filter-key> with one of the following options:
   * LAST_25_BUILDS
   * LAST_7_DAYS
Integrating Bamboo with Other Applications

- Integrating Bamboo with JIRA

Integrating Bamboo with JIRA

Integrating Bamboo with Atlassian’s JIRA combines Bamboo's continuous integration capabilities with your issue tracker to give you a unified view of your software development project. Using JIRA and Bamboo together, you can see which issues are being actively coded, which builds have run for an issue, find the build that fixed the issue, download your distribution and much more.

What version of Bamboo and JIRA do I need?

Bamboo-JIRA integration requires the following software versions:

- Bamboo 2.4 or later
- JIRA 4.0 or later

If you are using an earlier version of Bamboo and/or JIRA, you can download an older version of the JIRA Bamboo plugin from the Atlassian Plugin Exchange. However, the instructions on this page refer to the latest version of the required plugins (which are bundled with JIRA 4.0 and Bamboo 2.4). We strongly advise you to upgrade JIRA to version 4.0 and Bamboo to version 2.4, if you wish to integrate Bamboo with JIRA.

Configuring Bamboo and JIRA to work together is a simple three step process:

1. Configure the Bamboo plugin on the JIRA Server
2. Configure the JIRA plugin on the Bamboo Server
3. Add your JIRA Server as an OAuth Consumer in Bamboo

Before you begin

There are security considerations that you should take into account when integrating Bamboo with JIRA:

- When you connect JIRA to Bamboo during the integration process below, you must specify a user that JIRA uses to log in to Bamboo (see 1. Configure the Bamboo plugin on the JIRA Server). Please be aware that when JIRA accesses Bamboo, this user's Bamboo permissions will be used (not the Bamboo permissions of the user who is currently logged into JIRA). For example, a user viewing Bamboo information in JIRA will see all builds available to the user specified in the JIRA-Bamboo setup, instead of the builds available under their own permissions.

- Similarly, when you connect Bamboo to JIRA during the integration process below, you must specify a user that Bamboo uses to log in to JIRA (see 2. Configure the JIRA plugin on the Bamboo Server). Please be aware that when Bamboo accesses JIRA, this user's JIRA permissions will be used (not the JIRA permissions of the user who is currently logged into Bamboo). For example, a user viewing JIRA information in Bamboo will see all issues available to the user specified in the Bamboo-JIRA setup, instead of the issues available under their own permissions.

To prevent information in JIRA and Bamboo being unintentionally displayed to the wrong users, please ensure that you specify a user with appropriate permissions when connecting JIRA to Bamboo and vice versa.
1. Configure the Bamboo plugin on the JIRA Server

The Bamboo Plugin for JIRA is shipped with JIRA, so you do not need to download and install it. You only need to tell JIRA where to find your Bamboo server(s).

To configure your Bamboo plugin,

To configure your Bamboo plugin:

1. Launch your JIRA instance, if it is not already running.
2. In the JIRA 'Administration' screen, go to 'Global Settings' and select the 'Bamboo Servers' link. The Bamboo servers currently set up for your JIRA instance will display (if any).
3. Click the 'Add Bamboo server' link. The 'Add Bamboo server' page will display.
   - You can configure multiple Bamboo servers, but only one server can be associated to one JIRA project at a time.
4. Enter the following fields:
   - 'Server name' — The name of your Bamboo server.
   - 'Description' (optional) — A description of your Bamboo server.
   - 'Host' — The URL of your Bamboo instance e.g. http://localhost:8085/bamboo/
   - 'User name' — The user name that JIRA will use to login to Bamboo.
   - 'Password' — The password for the user specified above.
   - 'Associated JIRA projects' — The keys of the JIRA projects (e.g. 'JRA') that will be associated with this Bamboo server. Separate multiple project keys with spaces.
5. Click the 'Add' button to add your Bamboo server.
6. You will also need to allow remote API access in JIRA by enabling the 'Allow remote API access' option, as described in that JIRA documentation.

Managing access to Bamboo information in JIRA
Access to Bamboo information in JIRA (i.e. builds information related to issues, versions and projects) is managed by the 'View Version Control' project permission in JIRA. If you have not added this permission to your JIRA project, you will not be able to view any Bamboo information for that project. Read the JIRA documentation on project permissions for instructions on how to add this permission.

Non-standard JIRA issue keys
Please note, that you cannot change the JIRA issue key format from the standard configuration, if you wish to integrate JIRA with Bamboo. See BAM-3492 for further details.

You should now be able view information on your builds in your JIRA instance (e.g. view the related builds on a particular issue). To complete the integration of Bamboo with JIRA, the JIRA Plugin for Bamboo in your instance of Bamboo needs to be configured so that you can view JIRA information in Bamboo.

2. Configure the JIRA plugin on the Bamboo Server

The JIRA Plugin for Bamboo is shipped with Bamboo, so you do not need to download and install it. You simply need to tell Bamboo where to find your JIRA instance and provide it with the necessary authentication details.

To enable the JIRA integration plugin,

1. Launch your Bamboo instance, if it is not already running.
2. Click the 'Administration' link in the top navigation bar.
3. Click the 'JIRA Server' link in the left navigation column under the 'Communication' section. The 'Update JIRA Server Details' page will display.
   - In the 'Host URL' field, type the URL address of your JIRA server (e.g. 'http://jira.atlassian.com').
   - In the 'Username' field, type the name of the JIRA account which your Bamboo server will use to login to your JIRA server.
   - This JIRA account does not require JIRA administration permission.
   - In the 'Password' field, type the corresponding password for the JIRA account.
4. (Optional) In the 'Test' section, type a JIRA issue key in the 'Issue Key' field (e.g. 'BAM-738') and click the 'Test' button. This should display the following message: 'Successfully retrieved JIRA issue from remote server'. If not, check that you can login to your JIRA server using the JIRA account and password you specified in step 4.
5. When the test is successful, click the 'Save' button.
3. Add your JIRA Server as an OAuth Consumer in Bamboo

The final step required to integrate Bamboo with JIRA is to add your JIRA server as an OAuth Consumer in Bamboo. An OAuth 'consumer' is an application that accesses ('consumes') data from another application. When you add a JIRA server as an OAuth consumer in Bamboo, you are allowing that JIRA instance to access Bamboo's data. You need to do this if you want to add Bamboo gadgets to your JIRA dashboard.

To add your JIRA Server as an OAuth Consumer in Bamboo,

- To add your JIRA Server as an OAuth Consumer in Bamboo, follow the instructions in Configuring OAuth Consumers.

Congratulations! You have successfully integrated Bamboo and JIRA.

You may wish to read about how to use these two applications together in the following pages:

- View the Bamboo builds that relate to a particular JIRA issue.
- View the Bamboo builds that relate to a JIRA project or version.
- View the JIRA issues for a build result.
- Add JIRA gadgets to display the status of your builds or a graphical summary of each build plan (please note, if you have added multiple Bamboo servers there will be one set of these Bamboo gadgets available for each server, e.g. 'Bamboo Status Gadget from http://172.20.5.83:8085').

Need help?

If you need further help, please raise a support request in our support system, in the Bamboo project.

Bamboo Installation & Upgrade Guide

Bamboo 101 Guide

- Bamboo 101

Bamboo Installation Guides

- Bamboo Installation Guide
  - Bamboo EAR-WAR Installation Guide
  - Bamboo Remote Agent Installation Guide
  - Bamboo Standalone Installation Guide (Linux)
  - Bamboo Standalone Installation Guide (Mac)
  - Bamboo Standalone Installation Guide (Windows)
  - Running the Setup Wizard

Bamboo Upgrade Guides

- Bamboo Upgrade Guides
  - Bamboo Generic Upgrade Guide

Bamboo Release Notes

- Bamboo Release Notes
  - Bamboo Release Summary
  - Bamboo 2.4 Release Notes
  - Bamboo 2.3 Release Notes
  - Bamboo 2.2 Release Notes
  - Bamboo 2.1 Release Notes
  - Bamboo 2.0 Release Notes
  - Bamboo 2.0 Beta Release Notes
  - Bamboo 1.2 Release Notes
  - Bamboo 1.1 Release Notes
  - Bamboo 1.0 Release Notes
  - Bamboo 1.0-Beta Release Notes

Bamboo Security Advisories

- Bamboo Security Advisories
Bamboo Installation Guide

Requirements

1. **JDK/JRE 1.5+.** We **strongly recommend** that you use Sun JDK 1.5 (Java 5) and above to run your Bamboo installation. You are likely to experience incompatibility problems running Bamboo under a different JDK and ensure that there are no other JDK’s in your PATH.
   
   🔄 Don't forget to point your `JAVA_HOME` system environment variable to your Sun JDK before installing Bamboo, as Bamboo will automatically **configure JDK capabilities based on the system environment variables on your machine.**

2. **A database.** NOTE: Bamboo ships with a built-in HSQL database, which is fine for evaluation purposes. For production environments it is recommended that you use an enterprise database, as described in Connecting Bamboo to an external database.

3. **Only if you are using the Bamboo EAR-WAR distribution:** A servlet container that supports Servlet 2.4 specification. Most modern containers should comply to this.

   ![Info](image)
   
   While some of our customers run Bamboo on SPARC-based hardware, Atlassian only officially supports Bamboo running on x86 hardware and 64-bit derivatives of x86 hardware.

Choose your Bamboo ‘Distribution’:

Bamboo is available in two ‘distributions’:

<table>
<thead>
<tr>
<th>Standalone distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Pre-packaged with the Jetty application server</td>
</tr>
<tr>
<td>• Requires virtually no setup</td>
</tr>
<tr>
<td>• Recommended for all users</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EAR-WAR distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Deploys into an existing application server</td>
</tr>
<tr>
<td>• Requires manual configuration</td>
</tr>
<tr>
<td>• Suitable only for system administrators</td>
</tr>
</tbody>
</table>

The Standalone distribution is recommended even for organisations with an existing application server environment.

**NEXT**

- Standalone Installation Guide — Windows
- Standalone Installation Guide — Linux
- Standalone Installation Guide — Mac

or

- EAR-WAR Installation Guide

**RELATED TOPICS**

- Running the Setup Wizard
- Upgrade Guide
- Release Notes
- Bamboo Documentation Home

**Bamboo EAR-WAR Installation Guide**

The Bamboo EAR/WAR edition is intended for deployment into an existing J2EE application server. It is assumed that you already know how to deploy a webapp on the application server of choice. If not, it is recommended to install the Bamboo Standalone edition.
The following instructions are only indicative of the process and examples are based on installing the Bamboo WAR file on Apache Tomcat. Deployment and configuration will differ based on your webserver.

Before you begin
Please review the System Requirements.

Step 1. Download and install Bamboo EAR-WAR

1. The Bamboo WAR file is available for download here. Click the ‘Show all’ link on the downloads page to display the WAR file.
2. Deploy onto your application server. In Tomcat there are two ways you can do this:
   a. Place the WAR file directly into the webapps folder of Tomcat. When Tomcat starts it will perform all the necessary extractions.
   b. Extract the WAR file to your chosen directory in the webapps folder.
      - Windows users must avoid Win XP's built-in unzip as it doesn't extract all the files. Use a third-party zip extractor like WinZip.
      - Solaris users will need to use GNU tar to handle the long filenames.
   c. For Tomcat Application servers only: Extract the WAR file into a directory of your choice - this directory is referred to as <bamboo-install>. By default the WAR file will extract to a folder called Bamboo-<version>. Note: The name of the directory in the webapps folder will form the URL required to access Bamboo (eg. Tomcat/webapps/bamboo-1.0/ will become http://host:port/bamboo-1.0/). You may wish to change the directory name for a more concise access URL.

Step 2. Set Bamboo Home

You will need to set your Bamboo Home Directory. You can do this in one of three ways:

1. set the bamboo.home property in the file /WEB-INF/classes/bamboo-init.properties to your chosen Bamboo home directory.
2. pass the Bamboo home directory to the application server as a java opt. (eg. -Dbamboo.home=C:/bamboo/bamboohome).
3. specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your {BAMBOO_HOME} directory.

Step 3. Set jms.broker.uri

If you are going to use Bamboo remote agents, set the following in the /WEB-INF/classes/bamboo-init.properties file:

```
.bamboo.jms.broker.uri=tcp://localhost:54663
```

- replace 'localhost' with the real host name or IP address of your Bamboo server.
- if port number 54663 is already in use, specify a different port number.

Step 4. Set Java OPTs

It is recommended that you set the following Java OPTs on your Application Server.

- -server — Ensures that the jvm starts up in server mode. (This will perform various optimisation tasks, which is beneficial for long-running applications.)
- -Xmx512m — Sets the maximum memory recommended for Bamboo.
- -XX:MaxPermSize=256m — Sets the maximum permgen memory recommended for Bamboo.
- -Djava.awt.headless=true — For Unix systems. This allows AWT to run in headless mode and is required if running Bamboo in non-graphical environments. For more details visit the Sun Developer Network.

In Tomcat you can set the above Java OPTs as follows

- Windows:
  1. Find the setenv.bat file.
  2. Assign the desired properties to the JAVA_OPTS variable:

```
set JAVA_OPTS=-server -XX:MaxPermSize=256m -Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true $JAVA_OPTS
```

- Linux-based systems:
1. Find the setenv.sh file
2. Assign the desired properties to the JAVA_OPTS variable:

```
JAVA_OPTS="-server -XX:MaxPermSize=256m -Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m
-Djava.awt.headless=true $JAVA_OPTS"
export JAVA_OPTS
```

**Step 5. Edit Bamboo Context Descriptor (Tomcat)**

If you have extracted the WAR file to a directory other than the default directory (e.g. for a Tomcat application server), you will need to deploy `<bamboo-install>`, by following the steps below:

1. Create a file called `bamboo.xml` in your Tomcat installation's `conf/Catalina/localhost` directory. (If you have set up a different hostname for your Tomcat instance, please use your specified hostname instead of localhost.)
   
   Please note, for Tomcat 6, you must create the Catalina and localhost directories.

2. Open your new `bamboo.xml` and add the following:

   ```xml
   <Context path="/bamboo" docBase="c:/applications/bamboo-war/" debug="0" reloadable="true">
   </Context>
   ``
   
   If you wish to change the context path to a different name, change both the context path and the name of the xml file.

3. For docBase, specify the `<bamboo-install>` absolute path that you noted down earlier.

**Step 6. Restart Server**

1. Shut down, and then restart your application server.
2. Bamboo should now be accessible on `http://host:port/bamboo`.

**Step 7. Configure Bamboo**

See Running the Setup Wizard.

---

**Bamboo Remote Agent Installation Guide**

**Before you begin:**

- **Not sure whether to install a remote agent?** See About Agents to understand how remote agents interact with your Bamboo server.
- **Ensure that you have specified the Broker URL**, as described in the Bamboo Setup Wizard and the Bamboo 2.0 Upgrade Guide.
- **Do you have sufficient agent licenses?** See Bamboo licensing for details.
- **Have you enabled the creation of remote agents**, as described in Disabling Remote Agents Support.
- **Ensure that you have Java Runtime Environment 5.0 or later** installed on the agent machine.
- **Have you implemented your own remote agent service wrapper?** You may not want to use the remote agent supervisor that is bundled with the remote agent. You can choose to install the legacy remote agent (pre-Bamboo 2.2) instead, which does not have a service wrapper.

**On this page:**

- Step 1. Download and install the Remote Agent
- Step 2. Launch the Remote Agent
  - Changing where the remote agent stores its data
  - Changing the logging on the remote agent
  - Suppressing the self-signed certificate of the server
  - Running Bamboo without the Remote Agent Supervisor
  - Running the remote agent with different start-up commands
  - (Windows only) Installing the remote agent as a Windows service
- Step 3. Configure the Remote Agent’s Capabilities
- Step 4. (Optional) Rename the Remote Agent
To install the Bamboo Remote Agent manually,

**Step 1. Download and install the Remote Agent**

1. Create a directory on the agent machine (e.g. `bamboo-agent-home`) to serve as the Bamboo agent home for the remote agent.
2. On your Bamboo server, click the ‘Administration’ link in the top navigation bar.
3. Click the ‘Agents’ link in the left navigation column.
4. This will display the ‘Agents’ screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system.
5. Click the ‘Install Remote Agent’ link. The following screen will be displayed (click to view full-size image):

   ![Image of Install Remote Agent screen](image1)

   **Running a Remote Agent**
   
   Once installed, you can run the remote agent by executing the following command line from the directory containing the `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar` file:

   ```
   
   The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.
   
   You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or suppress the self-signed certificate of the server.

   **Changing where the remote agent stores its data**

   By default, the remote agent will store its data in a directory called `bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter before the JAR file name:

   ```
   -Dbamboo.home=RemoteAgentHome
   ```

   where `RemoteAgentHome` is the path to the Bamboo agent home directory you created in step 1.1.
   
   Your command line will look something like this:

   ```
   ```

   The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.

   **Changing the logging on the remote agent**

   ```
   ```

   **Step 2. Launch the Remote Agent**

   Once installed, you can run the remote agent by executing the command line obtained in the previous step. This command will look something like the following:

   ```
   ```

   You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or suppress the self-signed certificate of the server.

   **Changing where the remote agent stores its data**

   By default, the remote agent will store its data in a directory called `bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter before the JAR file name:

   ```
   -Dbamboo.home=RemoteAgentHome
   ```

   where `RemoteAgentHome` is the path to the Bamboo agent home directory you created in step 1.1.
   
   Your command line will look something like this:

   ```
   ```

   The name of the jar file (e.g. `atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar`) will vary depending on the version of Bamboo you are running.
By default, the remote agent will use the same logging level as the Bamboo server. However, you can control the level of logging of your remote agent independently of your Bamboo server by setting up a separate logging configuration file.

Please see Configuring Logging in Bamboo for further details.

Suppressing the self-signed certificate of the server

If your Bamboo server uses SSL (https) with a self-signed certificate, you will need to carry out one of the following two options:

- Add the following parameter "-Dbamboo.agent.ignoreServerCertName=true" to the remote agent's command line, for example:
  
  java -Dbamboo.agent.ignoreServerCertName=true -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer/

  Please be aware that this reduces the security of your configuration, as the identity of your Bamboo server will not be authenticated by the remote agent.

- Use the keytool utility to add the self-signed certificate to the trusted certificates in your keystore. This is a more secure option, but is complex to set up. For detailed instructions of how to do this, please refer to the relevant Sun documentation.

Running Bamboo without the Remote Agent Supervisor

The remote agent supervisor is included in the remote agent JAR bundled with Bamboo. The appropriate remote agent supervisor for the operating system of your remote machine, will be automatically installed when you run the default remote agent start-up command line.

The remote agent supervisor cannot be installed on a small number of operating systems (i.e. the remote agent will start without the remote agent supervisor). If the remote agent supervisor fails to install, please check the operated systems list on the remote agent supervisor page. If your operating system is on the list and the remote agent supervisor still fails to install, please raise a support request in the Bamboo project.

If you need to run the remote agent without running the remote agent supervisor, you can execute the 'classic' version of the remote agent JAR.

The 'classic' agent jar is available from bamboo's agent installation page for download. Follow the steps below to run the 'classic' version of the remote agent:

1. Browse to

   http://<host>:8085/admin/agent/addRemoteAgent.action

2. Click the 'the direct agent JAR is available at bamboo-agent-2.2.2.jar.' link and save 'classic' agent jar.

3. Start the agent with

   java -jar bamboo-agent-2.2.2.jar http://<host>:8085/agentServer/

   The name of the jar file (e.g. bamboo-agent-2.2.2.jar) will vary depending on the version of Bamboo you are running.

Running the remote agent with different start-up commands
The remote agent supervisor is executed by default when you run the default remote agent start-up command line. The remote agent supervisor is implemented via a Java Service wrapper. The wrapper allows you to execute a number of general start-up commands when the remote agent is run. These commands are appended to the end of the default remote agent start-up command line, i.e.

```java
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where `<wrapper_command>` is one of the keywords described below:

- **console** — runs the remote agent in the foreground, i.e. display all of the commands on the screen. The agent home directory will be populated only if it is empty. This parameter is used by default.
- **start** — runs the remote agent in the background, i.e. no commands are displayed on screen. If you have installed the remote agent as a Windows service, this command will work with the service.
- **stop** — stops a remote agent that is running. If you have installed the remote agent as a Windows service, this command will work with the service.
- **status** — (non-Windows OS only) returns the status of the remote agent, e.g. “Remote agent is not running.”
- **install** — installs the files for the remote agent, but does not start it. This will overwrite any changes that have been made to the wrapper.conf file. The agent home directory will be populated, regardless of whether it is empty or not, i.e. existing files will be overwritten. You may wish to use this option, if you want to customise the remote agent files before starting it.

*The name of the jar file (e.g. atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar) will vary depending on the version of Bamboo you are running.*

(Windows only) Installing the remote agent as a Windows service

The remote agent supervisor is executed by default when you run the default remote agent start-up command line. The remote agent supervisor is implemented via a Java Service wrapper. The wrapper allows you to install or uninstall the remote agent as a service in Windows (i.e. start the Bamboo remote agent automatically when the machine boots). This is done by appending the appropriate wrapper commands to the end of the default remote agent start-up command line, i.e.

```java
java -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer <wrapper_command>
```

where `<wrapper_command>` is one of the keywords described below:

- **installntservice** — (Windows only) installs the remote agent as a Windows service.
- **uninstallntservice** — (Windows only) uninstalls the remote agent as a Windows service.

*The name of the jar file (e.g. atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar) will vary depending on the version of Bamboo you are running.*

If you have installed the NT service, you will be able to use the **start** and **stop** start-up console commands with the service.

**Step 3. Configure the Remote Agent's Capabilities**

Please see [Configuring Capabilities](#).

**Step 4. (Optional) Rename the Remote Agent**

Your new remote agent has been automatically given a default name (e.g. 'Remote Agent on mymachine'). If you wish to rename your new remote agent, please see [Editing an Agent's Details](#).

Legacy Remote Agent Installation Guide
If you have implemented your own remote agent service wrapper or have problems with the service wrapper used by the remote agent supervisor in Bamboo, you can install the legacy remote agent (pre-Bamboo 2.2) which does not have a service wrapper.

**Before you begin:**

- **Not sure whether to install a Remote Agent?** See About Agents to understand how Remote Agents interact with your Bamboo server.
- **Ensure that you have specified the Broker URL**, as described in the Bamboo Setup Wizard and the Bamboo 2.0 Upgrade Guide.
- **Do you have sufficient Agent licenses?** See Bamboo licensing for details.
- **Have you enabled the creation of Remote Agents**, as described in Disabling Remote Agents Support.
- **Ensure that you have Java Runtime Environment 5.0 or later** installed on the agent machine.

**Step 1. Download and install the Legacy Remote Agent**

1. Create a directory on the agent machine (e.g. bamboo-agent-home), to serve as the "Bamboo agent home" for the remote agent.
2. On your Bamboo server, click the 'Administration' link in the top navigation bar.
3. Click the 'Agents' link in the left navigation column.
4. This will display the 'Agents' screen, showing lists of all Local Agents and all Remote Agents that currently exist in your Bamboo system.
5. Click the 'Install Remote Agent' button. The 'Install Remote Agent' screen will be displayed.
6. Click the 'bamboo-agent-jar' link in the 'Running the agents without the service wrapper' section and save the JAR file to the directory you created in step 1.1.

**Step 2. Launch the Remote Agent**
Once installed, you can run the remote agent by executing the command line obtained in the previous step. This command will look something like the following:

```
```

You may wish to configure the remote agent machine to start the Bamboo remote agent automatically when the machine boots. Please consult your operating system documentation for instructions on how to do this.

You can also choose to run the remote agent with different command line parameters, to change where the remote agent stores its data or suppress the self-signed certificate of the server.

**Changing where the remote agent stores its data**

By default, the remote agent will store its data in a directory called `bamboo-agent-home`. If you wish to specify a different directory, add the following command line parameter:

```
-Dbamboo.home=RemoteAgentHome
```

where `RemoteAgentHome` is the path to the Bamboo agent home directory you created in step 1.1. Your command line will look something like this:

```
```

**Suppressing the self-signed certificate of the server**

If your Bamboo server uses SSL (https) with a self-signed certificate, you will need to carry out one of the following two options:

- **Add the following parameter** `-Dbamboo.agent.ignoreServerCertName=true` **to the remote agent’s command line**, for example:
  ```
  java -Dbamboo.agent.ignoreServerCertName=true -jar bamboo-agent-2.0-SNAPSHOT.jar http://bamboo-host-server:8085/agentServer/
  ```
  Please be aware that this reduces the security of your configuration, as the identity of your Bamboo server will not be authenticated by the remote agent.
- **Use the keytool utility to add the self-signed certificate to the trusted certificates in your keystore**. This is a more secure option, but is complex to set up. For detailed instructions of how to do this, please refer to the relevant Sun documentation.

**Step 3. Configure the Remote Agent’s Capabilities**

Please see Configuring Capabilities.

**Step 4. (Optional) Rename the Remote Agent**

Your new remote agent has been automatically given a default name (e.g. 'Remote Agent on mymachine'). If you wish to rename your new remote agent, please see Editing an Agent's Details.

**Bamboo Standalone Installation Guide (Linux)**

To install Bamboo Standalone on Linux,
Step 1. Download and installing Bamboo Standalone

Before you get started
Please ensure that you have read the Server Requirements section of the Bamboo Installation Guide

Bamboo Standalone for Linux is available for download here.

Linux Archive (.tar.gz)

1. To install Bamboo using the Linux archive version (atlassian-bamboo-x.x-standalone.tar.gz), you need to extract the files to a Bamboo installation directory of your choice. By default, the root directory of the tar file is “Bamboo”.
2. You will also need to set up your Bamboo home directory — this is the directory where Bamboo will store its configuration data. To do this, open the file named bamboo-init.properties in the <Bamboo installation directory>/webapp/WEB-INF/classes directory. In this file, insert the property “bamboo.home”, with an absolute path to your Bamboo home directory. Your file should look something like this:

```
bamboo.home=/test/bamboo-home
```

Alternatively, you can specify an environment variable ‘BAMBOO_HOME’ which specifies the absolute path to your {BAMBOO_HOME} directory. Bamboo will check if an environment variable is defined.

Step 2. Launch Bamboo Standalone on Linux

There are two ways you can launch Bamboo on Linux:

1. Launch via bamboo.sh startup script

You can start Bamboo with the default bamboo.sh file in your installation root directory. The bamboo.sh command accepts the following options (e.g. ./bamboo.sh start):

- start — this starts Bamboo.
- stop — this stops Bamboo.
- restart — this restarts Bamboo.
- status — this provides the current status of Bamboo.

2. Launch via Java Service Wrapper

The wrapper is platform specific, and doesn’t work on SunOS.

Alternatively, you can start Bamboo via a Java Service Wrapper, which provides services such as automatic restarting. To do this, you will need to use the start-bamboo command available in the /wrapper folder of the Bamboo installation. You will need to fire the command with one of the following options (e.g. ./start-bamboo start):

- console — this starts Bamboo in a console. The logs will scroll to standard out.
- start — this starts Bamboo.
- stop — this stops Bamboo.
- restart — this restarts Bamboo.
- status — this provides the current status of Bamboo.
- dump — stops Bamboo abruptly by killing the process

Once Bamboo has started, you can access it by going to your web browser and entering the address: http://localhost:8085/.

If you have installed Bamboo on a machine with multiple interfaces, and need to bind Bamboo to a single IP address, please see Binding Bamboo to one IP address.

Step 3. Configure Bamboo

See Running the Setup Wizard.
Bamboo Standalone Installation Guide (Mac)

To install Bamboo Standalone on Mac OS,

Step 1. Download and install Bamboo Standalone

Before you get started
Please ensure that you have read the Server Requirements section of the Bamboo Installation Guide

Bamboo Standalone for Mac OS is available for download here. You can choose an Installer (.dmg) or an Archive (.tgz).

Mac OS Installer (.dmg)

1. Launching the Bamboo Mac OS installer (atlassian-bamboo-x.x-standalone.dmg) will mount the Atlassian Bamboo installation volume. Launch the Bamboo Continuous Integration Server Installer.app to begin the installation wizard.
2. The installer requires you to specify two directories:
   - **Bamboo installation directory** — This is the directory where Bamboo's application files will be installed. The default is:
     
     ![Applications/Bamboo]

   - **Bamboo home directory** — This is the directory where Bamboo will store its configuration data. If the directory you specify doesn't exist, Bamboo will create the directory when it launches. The default is:
     
     ![Users/ current-user/ Bamboo-home]

You must use forward-slashes in your directory path. Backslashes are not recognised by Bamboo. Please ensure that the **Bamboo home directory** is not located inside the **Bamboo installation directory**

Mac OS Archive (.tgz)

1. To install Bamboo using the Mac OS archive version (atlassian-bamboo-x.x-standalone.tgz), you need to extract the files to a **Bamboo installation directory** of your choice. By default, the root directory of your tgz file is "Bamboo".
2. You will also need to setup your **Bamboo home directory** — this is the directory where Bamboo will store its root configuration data. To do this, open the file named bamboo-init.properties in the <Bamboo installation directory>/webapp/WEB-INF/classes directory. In this file, insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

   ```
bamboo.home=/test/bamboo-home
   ```

   Alternatively, you can specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your BAMBOO_HOME directory. Bamboo will check if an environment variable is defined.
3. If you are going to use Bamboo remote agents, set the following in the bamboo-init.properties file in the <Bamboo installation directory>/webapp/WEB-INF/classes directory:

   ```
bamboo.jms.broker.uri=tcp://localhost:54663
   ```

   - replace 'localhost' with the real host name or IP address of your Bamboo server.
   - if port number 54663 is already in use, specify a different port number.

Step 2. Launch Bamboo on Mac OS

There are two ways you can launch Bamboo on Mac OS:

1. **Launch via bamboo.sh startup script**

You can start Bamboo with the default bamboo.sh file in your installation root directory. The bamboo.sh command accepts the following options (e.g. ./bamboo.sh start):
1. **Start** Bamboo via Java Service Wrapper

Alternatively, you can start Bamboo via a Java Service Wrapper, which provides services such as automatic restarting. To do this, you will need to use the `run-bamboo` command available in the `/wrapper` folder of the Bamboo installation. You will need to fire the command with one of the following options (e.g. `./run-bamboo start`):

- `console` — this starts Bamboo in a console. The logs will scroll to standard out.
- `start` — this starts Bamboo.
- `stop` — this stops Bamboo.
- `status` — this provides the current status of Bamboo.

Once Bamboo has started, you can access it by going to your web browser and entering the address:

```
http://localhost:8085/
```

2. **Launch via Java Service Wrapper**

3. **Step 3. Configure Bamboo**

See Running the Setup Wizard.

**Bamboo Standalone Installation Guide (Windows)**

To install Bamboo Standalone on Windows,

**Step 1. Download and install Bamboo Standalone**

* **Before you get started**
  Please ensure that you have read the Server Requirements section of the [Bamboo Installation Guide](/doc/1.12installation/standalone-windows.html).

Bamboo Standalone for Windows is available for download [here](http://www.atlassian.com/software/bamboo). You can choose the **Windows Installer (.exe)** or the **Windows Archive (.zip)**.

**Windows Installer (.exe)**

1. Launch the Bamboo Windows installer (`atlassian-bamboo-x.x-standalone.exe`) to begin the installation wizard.
2. The installer requires you to specify two directories:
   - **Bamboo installation directory** — This is the directory where Bamboo's application files will be installed. The default is:
     ```
     C:/Program Files/Bamboo
     ```
   - **Bamboo home directory** — This is the directory where Bamboo will store its configuration data. If the directory you specify doesn't exist, Bamboo will create the directory when it launches. The default is:
     ```
     C:/Documents and Settings/<current-user>/Bamboo-home
     ```

   You must use forward-slashes in your directory path. Backslashes are not recognised by Bamboo. Please ensure that the **Bamboo home directory** is not located inside the **Bamboo installation directory**.

**Windows Archive (.zip)**

* **Warning: Some unzip programs cause errors**
  Some archive-extract programs cause errors when unzipping the Bamboo archive file. We highly recommend that you use the free 7Zip archive-extract program. If in doubt, download the '32-bit .exe' version.

1. To install Bamboo using the Windows archive version (`atlassian-bamboo-x.x-standalone.zip`), you need to extract the files to a **Bamboo installation directory** of your choice. By default, the root directory in your zip file is named "Bamboo".
2. You will also need to setup your **Bamboo home directory** — this is the directory where Bamboo will store its root configuration data. To
do this, edit the file named `bamboo-init.properties` in the `Bamboo/webapp/WEB-INF/classes` directory. In this file, insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

```
bamboo.home=C:/test/bamboo-home
```

Alternatively, you can specify an environment variable 'BAMBOO_HOME' which specifies the absolute path to your (BAMBOO_HOME) directory. Bamboo will check if an environment variable is defined.

3. If you are going to use Bamboo remote agents, set the following in the `bamboo-init.properties` file in the `<Bamboo installation directory>/webapp/WEB-INF/classes` directory:

```
bamboo.jms.broker.uri=tcp://localhost:54663
```

- replace 'localhost' with the real host name or IP address of your Bamboo server.
- if port number 5463 is already in use, specify a different port number.

### Step 2. Launch Bamboo

Once Bamboo is installed on your machine, you can launch the application either via the Start Menu (if you have used the self installer), or by running the batch files available in the root of the Bamboo installation directory. You can run Bamboo in two modes: either in the console, or as a Windows service.

#### Running Bamboo as a service in Windows Vista

Bamboo ships with a service wrapper in Windows and by default, the wrapper installs itself as the NT SYSTEM user. In Vista the temporary directory System Variable is not available to untrusted apps. In order to run Bamboo as service, you need to run Bamboo as a non-system user, as per this document.

Bamboo comes with the following batch files:

- `BambooConsole.bat` — this starts Bamboo in a Windows console.
- `InstallAsService.bat` — this installs Bamboo as a Windows service. Note that this will not start Bamboo.
- `StartBamboo.bat` — this starts your installed Bamboo Windows service.
- `StopBamboo.bat` — this stops your installed Bamboo Windows service.
- `UninstallService.bat` — this un-installs the Bamboo Windows service from your machine. Note that your Bamboo installation still remains.

Once Bamboo has started, you can access it by going to your web browser and entering the default address: `http://localhost:8085/`

#### Running Bamboo as a Service

The default behaviour in Windows is to start the service under the 'SYSTEM' user when Bamboo starts up. You can change this behaviour from the 'Services' console available via the 'Control Panel'. Please note, you may need to uninstall the service and re-install it again, if you have just upgraded or re-installed Bamboo.

#### Launching Bamboo on a Windows x64 Platform

If you wish to run Bamboo on a Windows x64 platform, you must launch Bamboo by running `BambooConsole.bat`. The win 32-bit based wrapper will throw an error, however Bamboo can still be used as normal.

If you wish to stop Bamboo, open the terminal that `BambooConsole.bat` is running in and press 'Ctrl-C'.

### Step 3. Configure Bamboo

See Running the Setup Wizard.

#### Bamboo Startup Error on win x64

If `bamboo` is launched via `bambooconsole.bat` on a Windows x64 platform the 32 bit Tanuki Service wrapper throws an error.
The reason is that the (free) wrapper is for x32bit systems only. A commercial x64 wrapper can be obtained from Tanuki.

### Running the Setup Wizard

#### Before you begin
If you are currently using Atlassian's Crowd with Bamboo and wish to import existing data into Bamboo (see Step 3. Starting Data below), you will need to disable Crowd before starting the Setup Wizard. You can then re-enable Crowd and restart Bamboo at the completion of the Setup Wizard.

#### Step 1. Installation Settings
When you launch Bamboo for the first time, you will need to provide some configuration information before you can start using it.
Welcome to Atlassian Bamboo!

Welcome to Bamboo Continuous Integration Server. Bamboo needs some information before it is fully installed.

Standard Installation Settings

Licensing

Server ID: AQ5P-F32-AQ5P-F32
License Key: *

Please enter your Bamboo license key above - either commercial or evaluation. Please contact sales if you require a license key.

System Paths and Directories

The configuration directory is where your settings for Bamboo will be stored. For example: /opt/dev/data/bamboo-home-functional-20091115/xml-data/configuration

The build directory is where Bamboo saves the build data such as results and artifacts. For example: /opt/dev/data/bamboo-home-functional-20091115/xml-data/builds

Specify the absolute path on the server. For example: /opt/dev/data/bamboo-home-functional-20091115/xml-data/build-dir

Broker URL: tcp://sapporo.sydney.atlassian.com:54663
The set on which your messaging broker will be set up. The messaging broker is used for communication with Bamboo remote agents. Remote agents will also use this path to communicate back to this server.

Step 2. Database

Here, you will choose what type of database Bamboo will use:

- **Server ID** — This is generated automatically by Bamboo.
- **License Key** — You are required to enter a valid license key before you can use Bamboo. You can generate your own Bamboo evaluation license from your Atlassian self-service account here. If you have any problems with this please email sales.
- **Configuration Directory** — This is where Bamboo will store its configuration files.
- **Build Data Directory** — This is where Bamboo will store its project data files.
- **Build Working Directory** — This is where Bamboo will check out project files from source control.

⚠️ You may find it simplest to keep the default settings for the above three directories. For more information please see Locating Important Directories and Files.

- **Broker URL** — This is the URL of the embedded messaging broker that Bamboo sets up to communicate with its remote build agents. This URL will be written to bamboo.cfg.xml as a property, but can also be manually specified as a system property. The system property, if specified, will override the bamboo.cfg.xml property.

```
tcp://localhost:54663
```

- Replace localhost with the real host name or IP address of your Bamboo server. You should not use localhost as the host name in the Broker URL, as remote agents are provided with the Broker URL on startup and use it to communicate to the server.
- If port number 54663 is already in use, specify a different port number.
Choose a Database Configuration

Choose where Bamboo should store its data

Select Database

- Embedded Database
- External Database

Embedded Database

The embedded database will allow Bamboo to operate without an external database.

External Database

If you wish to store your Bamboo data in an external database, choose it from the list of supported databases. This is recommended for production systems.

If your database is not listed in the menu, you may configure an 'Unsupported Database', but be aware that Bamboo may not be fully tested.

- 'Embedded Database' — choose this for quick and easy first-time installation of Bamboo.
  
  Note that the embedded HSQL database is suitable for evaluation purposes only. You should later move to an external database before deploying Bamboo in production, as described in Moving your Bamboo Data to a different Database.

  OR:

- 'External Database' — if you wish to use an external database, please see Connecting Bamboo to an external database.

Step 3. Starting Data

Here, you will tell Bamboo how to populate the 'home directory' that you setup when you installed Bamboo.

Starting Data

Select Starting Data for Bamboo

Would you like to:

- Create new Bamboo home
- Import existing data

Step 4. Bamboo Administrator

Here, you will enter the details of the first registered user to the Bamboo system. This user will have global administrative privileges over the entire installation of Bamboo and should not be removed.
Step 5. Server Configuration

The final page of the Setup Wizard allows you to enter some final configuration data for Bamboo.

- **Name** — for more details please see Specifying Bamboo’s Title.
- **Base URL** — for more details please see Specifying Bamboo’s URL.
- **‘Apply gzip compression to reduce the size of Bamboo’s web pages?’** — for more details please see Enabling GZIP Compression.
- **‘Accept remote API calls?’** — for more details please see Enabling Bamboo’s Remote API.

Once you have clicked “Complete Installation”, the setup process is done and you are now at the Bamboo dashboard.
Connecting Bamboo to an external database

Bamboo can be connected to an external database. For details and instructions please see:

- MySQL 4.1 and 5.0
- Postgres 8+
- Oracle 9i and 10g
- Microsoft SQL Server
- Unsupported databases

**Before you begin**
Please note: if you are already using Bamboo with the embedded HSQL database (or any other database), and you want to keep your data, please see Moving your Bamboo Data to a different Database.

### MySQL 4.1 and 5.0

**Before you begin**
Please ensure that your MySQL database server is set to 'utf8' character encoding, and not 'latin1' character encoding. For details please see:

- MySQL 4.1 documentation: Database Character Set
- MySQL 5 documentation: Database Character Set

First, you need to choose how you will connect to the MySQL database. Please follow the instructions for your chosen method:

- **Connecting via JDBC**
- **Connecting via a datasource**

JDBC is generally simpler, and is therefore the recommended method.

The external database needs to be created manually before bamboo can connect to it.

**Connecting via JDBC**

To connect Bamboo to a MySQL database, via JDBC,
1. Put the MySQL JDBC driver jar file (download here) into your application server's classpath:
   - For the Bamboo Standalone distribution, copy the jar file into the webapp/WEB-INF/lib directory.
   - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'MySQL' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Direct JDBC connection'.
4. The 'Setup JDBC Connection' screen will appear as shown in the screenshot below.
   - 'Driver Class Name' — Type the following: com.mysql.jdbc.Driver
   - 'Database URL' — Type the URL where Bamboo will access your database. For syntax, please see the MySQL documentation.
     - Include the autoReconnect=true flag, the useUnicode=true flag and the characterEncoding=utf8 flag, e.g.:
       jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8
     - If the autoReconnect flag is not set, the MySQL JDBC driver will eventually time out and Bamboo will no longer be able to communicate with the database.
   - 'User Name' — Type the username that Bamboo will use to access your database.
   - 'Password' — Type the password (if required) that Bamboo will use to access your database.
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

Screenshot 1: ‘Setup JDBC Connection (MySQL)’

**Setup JDBC Connection**

- **Driver Class Name**: com.mysql.jdbc.Driver
  - The class name of the database driver. Ensure that this class is in your class path.
- **Database URL**: jdbc:mysql://localhost/bamboo?autoReconnect=true
  - The url to access the database
- **User Name**: 
  - The username to access the database
- **Password**: 
  - The password (if required) that Bamboo will use to access your database.
  - □ Overwrite existing data
    - If you want Bamboo to overwrite any existing tables that may exist in the database.

Connecting via a datasource

To connect Bamboo to a MySQL database, via a datasource,
1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see the MySQL documentation.
   - In the JDBC URL that you configure in your application server, include the `autoReconnect=true` flag, the `useUnicode=true` flag and the `characterEncoding=utf8` flag, e.g.:
     
     ```
     jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8
     ```
   - If the `autoReconnect` flag is not set, the MySQL JDBC driver will eventually time out and Bamboo will no longer be able to communicate with the database.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'MySQL' from the list.
3. The 'Select Database ConnectionWizard' screen will appear. Select 'Connect via a datasource (configured in the application server)'.
4. The 'Setup Datasource Connection' screen will appear as shown in the screenshot below. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
   - If `java:comp/env/jdbc/DataSourceName` doesn't work, try `jdbc/DataSourceName` (and vice versa).
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

Screenshot 2: 'Setup Datasource Connection'

```
<table>
<thead>
<tr>
<th>JNDI name:</th>
</tr>
</thead>
</table>
| ![JNDI name field](image)

- Overwrite existing data
  - Check if you wish Bamboo to overwrite any existing tables that may exist in the database.

[Continue]
```

Postgres 8+

Before you begin

If connecting to a remote PostgreSQL server (i.e. if your PostgreSQL server is not installed locally on your Bamboo server host system), please ensure that your `data/postgresql.conf` and `data/pg_hba.conf` files are configured to accept remote TCP connections from the Bamboo server's IP address. Refer to the PostgreSQL documentation for the `listen_addresses` value in the `postgresql.conf` file, as well as documentation for the `pg_hba.conf` file, for enabling TCP connections to your PostgreSQL server. (Note that you will need to restart PostgreSQL once any changes to these files have been made.) See:

- PostgreSQL 8.0 documentation
- PostgreSQL 8.1 documentation
- PostgreSQL 8.2 documentation

First, you need to choose how you will connect to the Postgres database. Please follow the instructions for your chosen method:

- Connecting via JBDC
- Connecting via a datasource

JDBC is generally simpler, and is therefore the recommended method.

Connecting via JBDC

To connect Bamboo to a Postgres database, via JDBC,
1. Put the Postgres JDBC driver **jar file** (download here) into your application server’s classpath:
   - For the Bamboo Standalone distribution, copy the **jar** file into the `webapp/WEB-INF/lib` directory.
   - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. At Step 2 of the Bamboo Setup Wizard, choose **External Database** and select **Postgres** from the list.
3. The ‘Select Database Connection’ screen will appear. Select ‘Direct JDBC connection’.
4. The ‘Setup JDBC Connection’ screen will appear as shown in the screenshot below.
   - ** ‘Driver Class Name’** — Type the following: `org.postgresql.Driver`
   - **‘Database URL’** — Type the URL where Bamboo will access your database. For syntax, please see the Postgres JDBC driver documentation.
   - **‘User Name’** — Type the username that Bamboo will use to access your database.
   - **‘Password’** — Type the password (if required) that Bamboo will use to access your database.
5. Select the ‘Overwrite existing data’ checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

**Screenshot 1: ‘Setup JDBC Connection (Postgres)’**

**Setup JDBC Connection**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Class Name</td>
<td><code>org.postgresql.Driver</code></td>
</tr>
<tr>
<td>Database URL</td>
<td><code>jdbc:postgresql://localhost:5432/bamboo</code></td>
</tr>
<tr>
<td>User Name</td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Overwrite existing data</td>
<td><img src="on-off-6.png" alt="Checkbox" /></td>
</tr>
</tbody>
</table>

**Connecting via a datasource**

To connect Bamboo to a Postgres database, via a datasource,

1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see the Postgres JDBC driver documentation.
2. At Step 2 of the Bamboo Setup Wizard, choose **External Database** and select **Postgres** from the list.
3. The ‘Select Database Connection’ screen will appear. Select **Connect via a datasource (configured in the application server)**.
4. The ‘Setup Datasource Connection’ screen will appear as shown in the screenshot below. In the ‘JNDI name’ field, type the JNDI name of your datasource, as configured in your application server.
   - ![Alert](exclamation.png) If `java:comp/env/jdbc/DataSourceName` doesn’t work, try `jdbc/DataSourceName` (and vice versa).
5. Select the ‘Overwrite existing data’ checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

**Screenshot 2: ‘Setup Datasource Connection’**
Bamboo provides two ways to connect to an Oracle database — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

**Oracle 9i and 10g**

Regardless of what version of the Oracle database you are using, you should use the **Oracle 10g** JDBC drivers. (Note: Oracle 10g JDBC drivers will not work with Oracle 8.1.6. For further reference, see Oracle FAQ)

- We highly recommend using the **thin** drivers.

**Connecting via JBDC**

To connect Bamboo to a Oracle database, via JDBC,

1. Put the Oracle JDBC driver jar file into your application server's classpath:
   - For the Bamboo Standalone distribution, copy the jar file into the `webapp/WEB-INF/lib` directory.
   - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'Oracle' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Direct JDBC connection'.
4. The 'Setup JDBC Connection' screen will appear as shown in the screenshot below.
   - 'Driver Class Name' — Type the following: `oracle.jdbc.driver.OracleDriver`
   - 'Database URL' — Type the URL where Bamboo will access your database. For syntax, please see the Oracle documentation, e.g. `jdbc:oracle:thin:@localhost:1521:SID`
   - 'Username' — Type the username that Bamboo will use to access your database.
   - 'Password' — Type the password that Bamboo will use to access your database.
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Click 'Continue' to finish specifying your connection settings.

**Screenshot 1: “Setup JDBC Connection (Oracle)”**
Connecting via a datasource

To connect Bamboo to a Oracle database, via a datasource,

1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see the Oracle documentation.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'Oracle' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Connect via a datasource (configured in the application server)'.
4. The 'Setup Datasource Connection' screen will appear as shown in the screenshot below. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
   - If java:comp/env/jdbc/DataSourceName doesn't work, try jdbc/DataSourceName (and vice versa).
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Click 'Continue' to finish specifying your connection settings.

Screenshot 2: 'Setup Datasource Connection'

Microsoft SQL Server

⚠️ Before you begin
We strongly recommend, using the open source JTDS JDBC driver version 1.0.3 or above, for MS SQL Server integration.

Bamboo provides two ways to connect to an MS SQL Server database — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

- Connecting via JDBC
- Connecting via a datasource

Connecting via JDBC

To connect Bamboo to a MS SQL Server database, via JDBC,
1. Copy the MS SQL Server JDBC driver jar (download here) file into your application server's classpath:
   - For the Bamboo Standalone distribution, copy the jar file into the webapp/WEB-INF/lib directory.
   - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'MSSQL' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Direct JDBC connection'.
4. The 'Setup JDBC Connection' screen will appear as shown in the screenshot below.
   - 'Driver Class Name' — Type the following: net.sourceforge.jtds.jdbc.Driver
   - 'Database URL' — Type the URL where Bamboo will access your database. For syntax, please see this MS SQL Server documentation: e.g. jdbc:jtds:sqlserver://localhost:1433/<database>
   - 'Username' — Type the username that Bamboo will use to access your database.
   - 'Password' — Type the password that Bamboo will use to access your database.
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.

   **Mixed mode Windows Authentication.**
   On a typical MS SQL Server installation, 'Windows Authentication' mode is the default security mode. This means that if you try to connect to the database with a database user, MS SQL Server will throw the following error, "Login failed for user 'sa'. Reason: Not associated with a trusted SQL Server connection." To resolve this, you can either:
   - Enable 'Mixed Mode Authentication'
   - Log in with a Windows user account, which has permission to administer the Bamboo database.

   For further information, please consult this MSDN article.

6. Click 'Continue' to finish specifying your connection settings.

### Screenshot 1: ‘Setup JDBC Connection (MS SQL Server)’

#### Setup JDBC Connection

- **Driver Class Name:** net.sourceforge.jtds.jdbc.Driver
- **Database URL:** jdbc:jtds:sqlserver://localhost:1433/<database>
- **Username:**
- **Password:**
- **Overwrite existing data:**

[Continue]

### Connecting via a datasource

To connect Bamboo to a MS SQL Server, via a datasource,

...
Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the MS SQL Server URL to use, please see the MS SQL Server documentation.

At Step 2 of the Bamboo Setup Wizard, choose ‘External Database’ and select ‘MSSQL’ from the list.

The ‘Select Database Connection’ screen will appear. Select ‘Connect via a datasource (configured in the application server)’.

The ‘Setup Datasource Connection’ screen will appear as shown in the screenshot below. In the ‘JNDI name’ field, type the JNDI name of your datasource, as configured in your application server.

If `java:comp/env/jdbc/DataSourceName` doesn't work, try `jdbc/DataSourceName` (and vice versa).

Select the ‘Overwrite existing data’ checkbox if you wish Bamboo to overwrite any tables that already exist in the database.

Mixed mode Windows Authentication.
On a typical MS SQL Server installation, ‘Windows Authentication’ mode is the default security mode. This means that if you try to connect to the database with a database user, MS SQL Server will throw the following error, “Login failed for user ‘sa’. Reason: Not associated with a trusted SQL Server connection.” To resolve this, you can either,

- Enable ‘Mixed Mode Authentication’
- Log in with a Windows user account, which has permission to administer the Bamboo database.

For further information, please consult this MSDN article.

Click ‘Continue’ to finish specifying your connection settings.

**Unsupported databases**

First, you need to choose how you will connect to your database. Please follow the instructions for your chosen method:

- Connecting via JDBC
- Connecting via a datasource

JDBC is generally simpler, and is therefore the recommended method.

**Connecting via JDBC**

To connect Bamboo to an unsupported database, via JDBC,
1. Put the appropriate JDBC driver jar file into your application server's classpath:
   - For the Bamboo Standalone distribution, copy the jar file into the `webapp/WEB-INF/lib` directory.
   - For the Bamboo EAR-WAR distribution, the location will depend on which application server you are using.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'Unsupported Database' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Direct JDBC connection'.
4. The 'Setup JDBC Connection' screen will appear as shown in the screenshot below.
   - 'Driver Class Name' — Type the classname of your JDBC driver (consult your JDBC driver documentation for details).
   - 'Database URL' — Type the URL where Bamboo will access your database (consult your JDBC driver documentation for details).
   - 'User Name' — Type the username that Bamboo will use to access your database.
   - 'Password' — Type the password (if required) that Bamboo will use to access your database.
   - 'Hibernate Dialect' — Type the Hibernate dialect for your particular database:
     
     **NOTE:** the databases on this list are **not supported by Atlassian**. Using these databases is not recommended as there is no guarantee that they will operate correctly with Bamboo. Please consider using a supported database instead.

<table>
<thead>
<tr>
<th>Database</th>
<th>Dialect</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2</td>
<td>net.sf.hibernate.dialect.DB2Dialect</td>
</tr>
<tr>
<td>DB2 AS/400</td>
<td>net.sf.hibernate.dialect.DB2400Dialect</td>
</tr>
<tr>
<td>DB2 OS390</td>
<td>net.sf.hibernate.dialect.DB2390Dialect</td>
</tr>
<tr>
<td>Oracle 9/10g</td>
<td>net.sf.hibernate.dialect.Oracle9Dialect</td>
</tr>
<tr>
<td>Oracle (other versions)</td>
<td>net.sf.hibernate.dialect.OracleDialect</td>
</tr>
<tr>
<td>Sybase</td>
<td>net.sf.hibernate.dialect.SybaseDialect</td>
</tr>
<tr>
<td>Sybase Anywhere</td>
<td>net.sf.hibernate.dialect.SybaseAnywhereDialect</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>net.sf.hibernate.dialect.SQLServerDialect</td>
</tr>
<tr>
<td>SAP DB</td>
<td>net.sf.hibernate.dialect.SAPDBDialect</td>
</tr>
<tr>
<td>Informix</td>
<td>net.sf.hibernate.dialect.InformixDialect</td>
</tr>
<tr>
<td>Ingres</td>
<td>net.sf.hibernate.dialect.IngresDialect</td>
</tr>
<tr>
<td>Progress</td>
<td>net.sf.hibernate.dialect.ProgressDialect</td>
</tr>
<tr>
<td>Mckoi SQL</td>
<td>net.sf.hibernate.dialect.MckoiDialect</td>
</tr>
<tr>
<td>Interbase</td>
<td>net.sf.hibernate.dialect.InterbaseDialect</td>
</tr>
<tr>
<td>Pointbase</td>
<td>net.sf.hibernate.dialect.PointbaseDialect</td>
</tr>
<tr>
<td>FrontBase</td>
<td>net.sf.hibernate.dialect.FrontbaseDialect</td>
</tr>
<tr>
<td>Firebird</td>
<td>net.sf.hibernate.dialect.FirebirdDialect</td>
</tr>
</tbody>
</table>

5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

*Screenshot 1: 'Setup JDBC Connection (Unsupported Database)'*
Connecting via a datasource

To connect Bamboo to an unsupported database, via a datasource,

1. Configure a datasource in your application server (consult your application server documentation for details). For the syntax of the JDBC URL to use, please see your JDBC driver documentation.
2. At Step 2 of the Bamboo Setup Wizard, choose 'External Database' and select 'Unsupported Database' from the list.
3. The 'Select Database Connection' screen will appear. Select 'Connect via a datasource (configured in the application server)'.
4. The 'Setup Datasource Connection' screen will appear as shown in the screenshot below. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
   - If java:comp/env/jdbc/DataSourceName doesn't work, try jdbc/DataSourceName (and vice versa).
5. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
6. Go to Step 3 of the Setup Wizard.

Screenshot 2: 'Setup Datasource Connection'

Unable to render embedded object: File (bamboo-database-jndi-connection.png) not found.

Using Bamboo’s embedded HSQL database

For quick and easy installation, use the embedded HSQL database that ships with Bamboo.

Note that the embedded HSQL database is suitable for evaluation purposes only. You should move to an external database before deploying Bamboo in production. For details please see Moving your Bamboo Data to a different Database.

To use Bamboo’s embedded HSQL database,

1. At Step 2 of the Setup Wizard, choose 'Embedded Database'.
2. Click the 'Continue' button.
3. Go to Step 3 of the Setup Wizard.
Bamboo Release Notes

Latest Version

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Release Summary

The features of each Bamboo release, up to and including the latest version, can be found in the Bamboo Release Summary.

For full details on each of the Bamboo releases, please read the relevant release notes listed below:

- Previous Production Releases
- Previous Beta Releases

You may also be interested in the Bamboo Upgrade Guides for each release.

Previous Production Releases

<table>
<thead>
<tr>
<th>Release Version</th>
<th>Release Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo 2.3.1</td>
<td>Bamboo 2.3.1 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.3</td>
<td>Bamboo 2.3 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.2.4</td>
<td>Bamboo 2.2.4 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.2.3</td>
<td>Bamboo 2.2.3 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.2.2</td>
<td>Bamboo 2.2.2 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.2.1</td>
<td>Bamboo 2.2.1 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1.5</td>
<td>Bamboo 2.1.5 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1.4</td>
<td>Bamboo 2.1.4 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.2</td>
<td>Bamboo 2.2 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1.3</td>
<td>Bamboo 2.1.3 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1.2</td>
<td>Bamboo 2.1.2 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1.1</td>
<td>Bamboo 2.1.1 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.1</td>
<td>Bamboo 2.1 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.6</td>
<td>Bamboo 2.0.6 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.5</td>
<td>Bamboo 2.0.5 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.4</td>
<td>Bamboo 2.0.4 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.3</td>
<td>Bamboo 2.0.3 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.2</td>
<td>Bamboo 2.0.2 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0.1</td>
<td>Bamboo 2.0.1 Release Notes</td>
</tr>
<tr>
<td>Bamboo 1.2.4</td>
<td>Bamboo 1.2.4 Release Notes</td>
</tr>
<tr>
<td>Bamboo 1.2.3</td>
<td>Bamboo 1.2.3 Release Notes</td>
</tr>
<tr>
<td>Bamboo 2.0</td>
<td>Bamboo 2.0 Release Notes</td>
</tr>
</tbody>
</table>
6 October 2009
The Atlassian Bamboo team is proud to release Bamboo 2.4.

Hot on the heels of Bamboo 2.3, our latest release comes packed full of improvements to key Bamboo features. If you use JIRA 4.0, you’ll be happy to know that Bamboo 2.4 is fully compatible with Atlassian’s biggest ever JIRA release. You’ll be able to take advantage of JIRA’s dynamic dashboards with our new Bamboo gadgets, including revamped versions of our existing Bamboo portlets.

We’ve also overhauled the Clover plugin that is bundled with Bamboo. If you use Atlassian’s Clover, you will be able to view your Clover HTML reports in Bamboo or even view your Clover information in JIRA via a gadget. Getting Clover to work with Bamboo is also much simpler — integrate with a single-click.

Finally, we’ve added a number of useful tools for Bamboo administrators and developers. The Bamboo REST API has been extended and now incorporates a host of new services, including services with POST methods. Bamboo administrators can also take advantage of our new runtime log4j configuration feature to configure logging levels for Bamboo classes on the fly.

Upgrading to Bamboo 2.4 is free for all customers with active Bamboo software maintenance.

Highlights of this release:
Thank you for your feedback:

🌟 5 new features and improvements implemented
🌟 2 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Download Latest Version

Please keep logging your votes and issues. They help us decide what needs doing!

### Upgrading to Bamboo 2.4

You can download Bamboo from the [Atlassian website](https://www.atlassian.com/software/bamboo). If upgrading from a previous version, please read the [Bamboo 2.4 Upgrade Guide](https://www.atlassian.com/university/professional-training/online-courses/bamboo-2.4-upgrade).

### Highlights of Bamboo 2.4

#### Bamboo Gadgets in JIRA

Atlassian’s JIRA 4.0 has a wealth of useful functionality and we’ve improved the Bamboo-JIRA integration in this release so you can take advantage of it. Our existing Bamboo portlets have been upgraded to gadgets for JIRA 4 and we’ve added some new gadgets too. You’ll notice that the Bamboo gadgets not only look better than the old portlets, they also provide you with more information about your Bamboo instance.

- Read more about [Integrating Bamboo with JIRA](https://www.atlassian.com/software/bamboo). If you don’t have JIRA 4 already, give it a try!
Clover Enhancements

We have made a number of improvements to the Clover plugin for Bamboo in this release. You can now integrate Atlassian's Clover with Bamboo via a single click, access embedded HTML reports and view Clover information in the Bamboo gadget.

- Read more about configuring the Clover plugin for Bamboo.

REST Improvements
The REST interface for Bamboo has been extended for this release. You will now be able to access a range of new REST methods to help you access and update Bamboo information.

- Read more about the Bamboo REST APIs.

Runtime Log4j Configuration

We're bundling a feature in Bamboo 2.4 that our own support staff have found incredibly helpful — runtime log4j configuration. This handy Bamboo administration tool allows you to temporarily adjust the logging levels defined in your log4j.properties file. You can change the logging levels on existing packages as well as add new packages to be monitored on the fly.

- Read more about configuring logging in Bamboo.

### Bamboo Log Settings

On the fly switch between Production and Debug Logging. The settings are predefined in log4j.properties

#### Add a Class to Log4j

Add a new class and level to be included in the bamboo log

- [Add](#)

#### Edit current Levels

Change the log level

<table>
<thead>
<tr>
<th>Package</th>
<th>Current Level</th>
<th>New Level</th>
<th>Save Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.atlassian.bamboo.agent</td>
<td>DEBUG</td>
<td>ALL</td>
<td><a href="#">Save Delete</a></td>
</tr>
<tr>
<td>com.atlassian.bamboo.buildqueue</td>
<td>DEBUG</td>
<td>ALL</td>
<td><a href="#">Save Delete</a></td>
</tr>
<tr>
<td>com.atlassian.bamboo.user.BambooUserManagerimpl</td>
<td>WARN</td>
<td>ALL</td>
<td><a href="#">Save Delete</a></td>
</tr>
<tr>
<td>com.opensymphony.xwork.util.LocalizedTextUtil</td>
<td>ERROR</td>
<td>ALL</td>
<td><a href="#">Save Delete</a></td>
</tr>
</tbody>
</table>

### Plus over 20 fixes and improvements

- [JIRA Issues (24 issues)](#)
<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-4293</td>
<td>REST API - Get current (running) build details</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4335</td>
<td>UniqueAuthorNameValidator throws NPE when repository alias is added and table author contains {{null}} entries</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4349</td>
<td>Sort the view capabilities page lexicographically</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4492</td>
<td>Update BuildResultService in REST documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4513</td>
<td>Prevent Jira Bamboo Plugin from adding gadgets to browser if they dont exist</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4573</td>
<td>phpunit builder should (optionally) only check for a return code or make the number of lines for 'OK' message flexible</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4384</td>
<td>Unable to view Plan in REST</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4517</td>
<td>svnkit authentication fails due to authentication file access problems on all platforms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4013</td>
<td>Favourite toggle broken in all plans screen</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4241</td>
<td>rest/api/latest/chart should return useful information</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>BAM-4360</td>
<td>Ensure that all commands when mounting EBS volumes handle failure recursively</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4376</td>
<td>Queue re-ordering doesn't work in Safari 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4342</td>
<td>Plugins 2 xwork plugins throw ClassNotFoundException</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4350</td>
<td>Maven 2.1 will not respect path if not already the default M2_HOME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4411</td>
<td>Build reorder section does not show up on the BuildQueue screen</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4523</td>
<td>OAuth page looks like crap</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4542</td>
<td>Connection from remote agent fails if the agent time is in the past</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>BAM-4543</td>
<td>No cancel link from login page</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4550</td>
<td>Legacy clover tab should not show if there is no data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4551</td>
<td>Plan is limited to 15 (Critical)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4553</td>
<td>JIRA Plugin &quot;Latest plan status&quot; link doesn't work (seems to just reload the page) in FireFox 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4558</td>
<td>Clover Gadget doesn't handle the case where there are no clover enabled plans very well - no warning, just a blank spot where the select list should be.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4562</td>
<td>Viewing REST pages on IE8 actually crashes IE8. this only seems to happen when you are not logged in</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>BAM-4568</td>
<td>Bamboo Gadgets/Rest Do not accomodate for when the bamboo instance has no plans.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 2.4 Upgrade Guide

Upgrading from Bamboo 2.3 to 2.4

We strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

We also strongly recommend that you export your Bamboo data for backup before proceeding. Please note, that this may take a long time to complete depending on the number of builds and tests in your system. For full instructions please see Exporting Data for Backup.

⚠️ If you are using plugins, please make sure that your plugins are compiled against 2.4 before upgrading.

Please also note the following important points:

1. Developing for Bamboo 2.4

The page Changes for Bamboo 2.4 does not exist.

Upgrading from Bamboo prior to 2.3

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.

In particular, if you are upgrading from a version of Bamboo prior to 2.0, please ensure that you upgrade to Bamboo 2.0.6 first before upgrading to Bamboo 2.4.

⚠️ Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.3 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

6 August 2009

The Atlassian Bamboo team is proud to release Bamboo 2.3.

Bamboo 2.3 is all about making your life easier. We've added a host of new features to help you manage your builds better, including dependency blocking, new build notifications, queue reordering and bulk actions for plans.

If you are using Elastic Bamboo, you'll find that Bamboo 2.3 gives you more control over your elastic resources with the introduction of custom elastic images and instance scheduling. You can now specify the availability zone for your elastic images as well, if you want to take advantage of the new 'reserved instances' option from Amazon.

Finally, Bamboo 2.3 includes a number of improvements for the plugin developers. There's a brand new REST API that you can use to get information about projects, plans, builds and reports. We've also added Bandana support and downloadable plugin and web resources.

Upgrading to Bamboo 2.3 is free for all customers with active Bamboo software maintenance.

Highlights of this release:
- Dependency Blocking Strategies
- New Build Notifications and Queue Reordering
- Bulk Actions
- Multiple Elastic Images
- Elastic Instance Scheduling
- PHPUnit Builder
- Bamboo REST APIs
- Plugins Changes
- Plus over 80 fixes and improvements

Thank you for your feedback:
Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!

### Upgrading to Bamboo 2.3

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 2.3 Upgrade Guide.

### Highlights of Bamboo 2.3

#### Dependency Blocking Strategies

Bamboo 2.3 gives you greater control over your builds with the introduction of dependency blocking. Dependency blocking is a advanced feature of dependent build triggering that can be used to manage builds which have parents. You can ensure that a “tree” of dependent builds always runs in order of the tree hierarchy, even if child builds are triggered independently of their parents.

- Read more about Dependency Blocking Strategies.
New Build Notifications and Queue Reordering

Bamboo automatically assigns builds to the build queue when they are triggered and no agents are available to run them. In this release, you can now manually reorder builds that have been placed in the build queue. Prioritize a build in the queue if you need it to run urgently, or demote low-priority builds. We've also added two new build notifications in this release, 'Build Queue Timeout' and 'Build Queued Without Capable Agents', to help you keep on top of your builds.

- Read more about Reordering your Build Queue and Adding or Removing Notifications for a Plan.
Bulk Actions

We’ve also made it easier for you to configure multiple build plans via the new bulk actions in Bamboo 2.3. Bulk actions allow you to modify key plan information for multiple plans at once, like adding notifications, changing Subversion URLs and credentials and updating web repository URLs.

- Read more about Modifying Multiple (Bulk) Plans.
Multiple Elastic Images

Atlassian supplies a default image for use with Elastic Bamboo. In this release, you can now create and/or associate multiple custom images (Linux/Unix) with your Bamboo installation. This means that you can use separate images to start up differently configured elastic instances.

If you want to use EC2 Reserved Instances with Elastic Bamboo, you can also manually specify the availability zone for each of your images in this release.

- Read more about Creating a Custom Elastic Image and Managing your Elastic Image Configurations.

### Manage Elastic Image Configurations

You can manage the configurations of elastic images that you have associated with Bamboo on this page. You can also associate additional elastic images by creating new elastic image configurations on this page.

<table>
<thead>
<tr>
<th>Name</th>
<th>AMI ID</th>
<th>EBS Snapshot ID</th>
<th>Instance Type</th>
<th>Availability Zone Preference</th>
<th>Active Instances</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default (default)</td>
<td>ami-311f066</td>
<td>snap-36c6a6c5c6</td>
<td>High-CPU Medium</td>
<td>Default (chosen by EC2)</td>
<td>10</td>
<td>Start</td>
</tr>
<tr>
<td>Reserved Instance 1</td>
<td>ami-311f566</td>
<td>snap-ac5e6b6c5c6</td>
<td>Small</td>
<td>us-east-1b</td>
<td>0</td>
<td>Start</td>
</tr>
</tbody>
</table>

### Create Elastic Image Configuration

**Elastic Image Configuration Details**

- **Name:** Ubuntu Image 1
- **Description:** Custom Ubuntu Image with Maven 2
- **AMI ID:** ami-56e5c6e6
- **EBS Snapshot ID:** Specify the EBS Snapshot ID to mount volumes from, when starting new elastic instances.
- **Instance Type:** High-CPU Medium
- **Availability Zone:** Default (chosen by EC2)

### Elastic Instance Scheduling

Bamboo 2.3 makes it easy for you to automatically streamline your build resources by configuring schedules for your elastic instances. You can specify exactly how many elastic instances you want to be active at a particular time and Bamboo will automatically start up or shut down elastic instances as needed.

- Read more about Scheduling your Elastic Instances.
PHPUnit Builder

We have added to our stable of builders in Bamboo by bundling the **PHPUnit builder** plugin with Bamboo. You can now configure build plans to run using this popular testing framework.

- Read more about configuring a PHPUnit builder for a plan.

Bamboo REST APIs

Bamboo 2.3 exposes a new **REST API** for developers. You can use the REST API to retrieve information about projects and plans as well as available actions. You can also retrieve information about build results and reports via the REST API. Results can be returned in either XML or JSON format.

- Read more about Bamboo REST APIs.
Plugins Changes

In further improvements for Bamboo developers, we have introduced a number of features to help you build Bamboo plugins more easily.

Firstly, Bamboo 2.3 now includes Bandana support. Bandana is our XML-based framework for persistence that is easy to use in plugins. You can use Bandana to store and retrieve data via contexts and key-value pairs.

- Read more about Bamboo Persistence using Bandana

In addition, you can now define downloadable plugin resources and web resources for your plugins. If you want to include static images, Javascript or CSS with your plugin, you can use downloadable plugin resources or web resources to make them available.

- Read more about Bamboo Persistence using Bandana, Downloadable Plugin Resources and Web Resources.

Plus over 80 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (86 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>BAM-3836</td>
</tr>
<tr>
<td>BAM-3635</td>
</tr>
<tr>
<td>BAM-3634</td>
</tr>
<tr>
<td>BAM-3614</td>
</tr>
<tr>
<td>BAM-3612</td>
</tr>
<tr>
<td>BAM-3287</td>
</tr>
<tr>
<td>BAM-2683</td>
</tr>
<tr>
<td>BAM-1677</td>
</tr>
<tr>
<td>BAM-1182</td>
</tr>
<tr>
<td>BAM-4327</td>
</tr>
<tr>
<td>BAM-4326</td>
</tr>
<tr>
<td>BAM-4324</td>
</tr>
<tr>
<td>BAM-4253</td>
</tr>
<tr>
<td>BAM-4195</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>BAM-4133</td>
</tr>
<tr>
<td>BAM-4093</td>
</tr>
<tr>
<td>BAM-4087</td>
</tr>
<tr>
<td>BAM-4084</td>
</tr>
<tr>
<td>BAM-4081</td>
</tr>
<tr>
<td>BAM-3890</td>
</tr>
<tr>
<td>BAM-3766</td>
</tr>
<tr>
<td>BAM-3710</td>
</tr>
<tr>
<td>BAM-3697</td>
</tr>
<tr>
<td>BAM-1844</td>
</tr>
<tr>
<td>BAM-1412</td>
</tr>
<tr>
<td>BAM-1393</td>
</tr>
<tr>
<td>BAM-932</td>
</tr>
<tr>
<td>BAM-781</td>
</tr>
<tr>
<td>BAM-743</td>
</tr>
<tr>
<td>BAM-734</td>
</tr>
<tr>
<td>BAM-560</td>
</tr>
<tr>
<td>BAM-4330</td>
</tr>
<tr>
<td>BAM-4325</td>
</tr>
<tr>
<td>BAM-3958</td>
</tr>
<tr>
<td>BAM-3788</td>
</tr>
<tr>
<td>BAM-2503</td>
</tr>
<tr>
<td>BAM-3784</td>
</tr>
<tr>
<td>BAM-3781</td>
</tr>
<tr>
<td>BAM-4257</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BAM-4289</td>
</tr>
<tr>
<td>BAM-4264</td>
</tr>
<tr>
<td>BAM-4263</td>
</tr>
<tr>
<td>BAM-4259</td>
</tr>
<tr>
<td>BAM-4095</td>
</tr>
<tr>
<td>BAM-3674</td>
</tr>
<tr>
<td>BAM-3650</td>
</tr>
<tr>
<td>BAM-3643</td>
</tr>
<tr>
<td>BAM-3495</td>
</tr>
<tr>
<td>BAM-4297</td>
</tr>
<tr>
<td>BAM-4292</td>
</tr>
<tr>
<td>BAM-4288</td>
</tr>
<tr>
<td>BAM-4262</td>
</tr>
<tr>
<td>BAM-4256</td>
</tr>
<tr>
<td>BAM-4244</td>
</tr>
<tr>
<td>BAM-4235</td>
</tr>
<tr>
<td>BAM-4180</td>
</tr>
<tr>
<td>BAM-4161</td>
</tr>
<tr>
<td>BAM-4142</td>
</tr>
<tr>
<td>BAM-4121</td>
</tr>
<tr>
<td>BAM-4119</td>
</tr>
<tr>
<td>BAM-4071</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>BAM-4069</td>
</tr>
<tr>
<td>BAM-4055</td>
</tr>
<tr>
<td>BAM-4016</td>
</tr>
</tbody>
</table>
BAM-3971 Improve command interface to allow better variable substitution

Resolved

BAM-3839 "Remove All Artifacts" link should require confirmation

Resolved

BAM-3819 Exceptions in logs when accessing REST method for build without build result

Closed

BAM-3770 guessChangeSetId returning null will cause emails to throw exceptions

Resolved

BAM-3757 After X Failed Notifications "Updated By" is broken.

Resolved

BAM-3722 AuthorName is not html escaped when serving Ajax responses

Resolved

BAM-3716 JIRA Bamboo plugin throwing NullPointerException for Confluence

Resolved

BAM-3703 Wrong defined M2_HOME for elastic bamboo image

Resolved

BAM-3689 Elastic agents do not automatically shut down if the Bamboo server is restarted

Resolved

BAM-3673 Expire builds throws SQLIntegrityConstraintViolationException

Resolved

BAM-3664 Build results REST XML invalid

Resolved

BAM-3523 Elastic agent not unregistered when instance is being killed manually

Resolved

BAM-3121 Changing the way the "Updated by" text field is being constructed

Resolved

BAM-2184 Disallow directory listing in Jetty.

Resolved

BAM-1469 Disabled agents are reenabled on restart

Resolved

BAM-4168 Web repository module can be added without a web repository url

Resolved

BAM-3660 Builds that took less than 1 sec to execute should be reported as 'less than a second' instead of 'Unknown'

Resolved

BAM-3242 Entering incorrect project key gives big stack trace

Resolved

BAM-2965 Missing custom build data when examined from a CustomBuildCompleteAction

Resolved

BAM-3787 View Plan > Tests: "Most Recent" column hard to understand. Most Recent What?

Resolved

BAM-3780 User Profile > Edit: "Jabber Address" label isn’t clear.

Resolved

BAM-3575 Weird Panelling on General Configuration Updated

Resolved

Bamboo 2.3.1 Release Notes

12 August 2009
The Atlassian Bamboo team is proud to announce the release of Bamboo 2.3.1. Bamboo 2.3.1 is a minor bug fix release. Most customers will not notice any changes from our last major release — Bamboo 2.3

SVNKit has been upgraded to version 1.3 in this release. Any new workspaces created will have a format that is compatible with Subversion 1.6, by default. This format is also compatible with Subversion 1.5. You can now also manually set the version of any new Subversion workspaces
created by Bamboo via the system property `bamboo.svn.wc.format`. Please see this FAQ for further details.

Bamboo 2.3.1 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.3 yet?**
Take a look at all the new features in the Bamboo 2.3 Release Notes and see what you are missing out on!

[Download Latest Version](#)

### Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.3.1 Upgrade Guide.

### Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (2 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

### Bamboo 2.3.1 Upgrade Guide

**Upgrading from Bamboo 2.3 to 2.3.1**

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.3 to 2.3.1.

**Upgrading from Bamboo 2.2.x or earlier**

In addition to the above, please read the Bamboo 2.3 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

### Bamboo 2.3 Upgrade Guide

**Upgrading from Bamboo 2.2 to 2.3**

We strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

We also strongly recommend that you **export your Bamboo data for backup** before proceeding. Please note, that this may take a long time to complete depending on the number of builds and tests in your system. For full instructions please see Exporting Data for Backup.

If you are using plugins, please make sure that your plugins are compiled against 2.3 before upgrading.

Please also note the following important points:

1. **Bamboo developers — Changes for 2.3**

If you are a Bamboo developer, please take note of the 'Changes for Bamboo 2.3' document when upgrading to 2.3. We have made **significant changes to Bamboo's remote API** to improve it. However, it is likely that a number of existing Bamboo plugins will not work as a result.

We **strongly recommend** that you take note of the changes linked above and update your plugins accordingly.

2. **Remote agents automatically upgraded**

Please note that your remote agents **do not** need to be manually upgraded for this release. They will be automatically upgraded when you upgrade your Bamboo instance.
Upgrading from Bamboo prior to 2.2

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.

In particular, if you are upgrading from a version of Bamboo prior to 2.0, please ensure that you upgrade to Bamboo 2.0.6 first before upgrading to Bamboo 2.3.

Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.2 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!

9 March 2009
The Atlassian Bamboo team is proud to release Bamboo 2.2.

Bamboo 2.2 introduces a range of exciting new features and improvements. Harness the flexible online processing potential of the Amazon Elastic Compute Cloud (EC2) to power your builds with the new Elastic Bamboo feature. This provides you with the power to reduce your build times as well as the flexibility in capacity to minimise costs.

Bamboo emails have been redesigned in this release. HTML templates are now supported and can be easily customised to your liking. We have also added new notification events in Bamboo, to allow you to collaborate more easily via build comments or detect when your builds hang.

Finally, we’ve implemented a few refinements to Bamboo builds. These include improving the performance of artifact transfer from remote agents and adding the ability to use the same repository snapshot revision for dependent builds.

Please note, this release contains a number of important security fixes. Please see Bamboo Security Advisory 2009-03-09 for further details.

Upgrading to Bamboo 2.2 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

- Elastic Bamboo
- Customisable Email Templates
- Build Comment Notification Event
- Hanging Build Detection Event
- Faster Artifact Transfer
- Dependent Builds
- Agent Improvements
- Plus over 80 fixes and improvements

Thank you for your feedback:

🌟 over 68 new features and improvements implemented
🌟 over 170 votes fulfilled

Your votes and issues help us keep improving our products, and are much appreciated.

Please keep logging your votes and issues. They help us decide what needs doing!

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 2.2 Upgrade Guide.

Highlights of Bamboo 2.2
Elastic Bamboo

Cloud computing comes to Bamboo with the introduction of the Elastic Bamboo feature. You can now configure your Bamboo application to create remote agents in the Amazon Elastic Compute Cloud (EC2). We've also incorporated a number of useful tools with this feature, that allow you to start up your builds more quickly via build snapshots, run Elastic Bamboo builds from behind a firewall and control Elastic Bamboo via the Bamboo REST API.

- Read more about Working with Elastic Bamboo.

Customisable Email Templates

Bamboo emails have been given a facelift in this release. Multi-part (MIME) format is now supported allowing you to use HTML in your email templates (e.g. for Bamboo notifications). We've set up Freemarker templates in Bamboo as well, making it easy for you to customise the look and feel of your emails.

- Read more about configuring your notification templates.
Build Comment Notification Event

The new build comment notification event makes it even easier to collaborate in Bamboo. You can set up Bamboo to notify selected users and groups when a comment is posted against a build. Users can be notified by email, instant message or even RSS feed, depending on their preference.

- Read more about notifications in Bamboo.

Hanging Build Detection Event

The hanging build detection event helps you identify and resolve build issues. It provides a detailed summary of failed builds, including code changes, test results, and error messages. You can take action directly from the summary to improve your build process.
We have also added a notification event for hung builds. Ensure that the right people are informed when a build hangs, by setting up notifications using this event. Users can be notified by email, instant message or even RSS feed, depending on their preference.

- Read more about configuring the hanging build event.

<table>
<thead>
<tr>
<th>JIRA FUNC-JDK16-3437 may have hung.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This build has been running for 294 minutes, which is 150% longer than usual. It has been 231 minutes since Bamboo received a log message for this build.</td>
</tr>
<tr>
<td>Running on agent Core 1</td>
</tr>
</tbody>
</table>

**Code Changes**

- trahilly
  - JIRA-1500 User custom field conversion.
- andreask
  - JIRA-16120: Major cleanup of the javascript.
- andreask

**Last Logs**


**Faster Artifact Transfer**

We have dramatically improved the speed of artifact transfer from remote agents in this release. Remote builds with large artifacts will complete much more quickly, particularly over high-latency network links.

- Read more about viewing a build’s artifacts.

**Dependent Builds**

Builds are now more consistent when triggering a build after another build finishes. If a child build uses the same source as the parent build, the child build will now be forced to check out the same revision of source code as the parent build.

- Read more about triggering a build when another build finishes.

**Agent Improvements**

We have introduced a remote agent supervisor in this release to monitor and automatically restart your remote agents, if necessary. You should notice an improvement in the uptime of your remote agents with this change.

- Read more about the remote agent supervisor in the Bamboo Remote Agent Installation Guide.
Plus over 80 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (114 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>BAM-3607</td>
</tr>
<tr>
<td>BAM-3605</td>
</tr>
<tr>
<td>BAM-3591</td>
</tr>
<tr>
<td>BAM-3561</td>
</tr>
<tr>
<td>BAM-3554</td>
</tr>
<tr>
<td>BAM-3553</td>
</tr>
<tr>
<td>BAM-3521</td>
</tr>
<tr>
<td>BAM-3509</td>
</tr>
<tr>
<td>BAM-3508</td>
</tr>
<tr>
<td>BAM-3504</td>
</tr>
<tr>
<td>BAM-3480</td>
</tr>
<tr>
<td>BAM-3479</td>
</tr>
<tr>
<td>BAM-3462</td>
</tr>
<tr>
<td>BAM-3458</td>
</tr>
<tr>
<td>BAM-3457</td>
</tr>
<tr>
<td>BAM-3445</td>
</tr>
<tr>
<td>BAM-3432</td>
</tr>
<tr>
<td>BAM-3412</td>
</tr>
<tr>
<td>BAM-3408</td>
</tr>
<tr>
<td>BAM-3386</td>
</tr>
<tr>
<td>BAM-3339</td>
</tr>
<tr>
<td>Issue ID</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BAM-3335</td>
</tr>
<tr>
<td>BAM-3329</td>
</tr>
<tr>
<td>BAM-3292</td>
</tr>
<tr>
<td>BAM-3281</td>
</tr>
<tr>
<td>BAM-3280</td>
</tr>
<tr>
<td>BAM-3250</td>
</tr>
<tr>
<td>BAM-3246</td>
</tr>
<tr>
<td>BAM-3243</td>
</tr>
<tr>
<td>BAM-3240</td>
</tr>
<tr>
<td>BAM-3220</td>
</tr>
<tr>
<td>BAM-3194</td>
</tr>
<tr>
<td>BAM-3189</td>
</tr>
<tr>
<td>BAM-3183</td>
</tr>
<tr>
<td>BAM-3179</td>
</tr>
<tr>
<td>BAM-3177</td>
</tr>
<tr>
<td>BAM-3176</td>
</tr>
<tr>
<td>BAM-3172</td>
</tr>
<tr>
<td>BAM-3168</td>
</tr>
<tr>
<td>BAM-3166</td>
</tr>
<tr>
<td>BAM-3155</td>
</tr>
<tr>
<td>BAM-3141</td>
</tr>
<tr>
<td>BAM-3138</td>
</tr>
<tr>
<td>BAM-3125</td>
</tr>
<tr>
<td>BAM-3122</td>
</tr>
<tr>
<td>BAM-3120</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BAM-3108</td>
</tr>
<tr>
<td>BAM-3105</td>
</tr>
<tr>
<td>BAM-3104</td>
</tr>
<tr>
<td>BAM-3091</td>
</tr>
<tr>
<td>BAM-3090</td>
</tr>
<tr>
<td>BAM-3089</td>
</tr>
<tr>
<td>BAM-3083</td>
</tr>
<tr>
<td>BAM-3043</td>
</tr>
<tr>
<td>BAM-2994</td>
</tr>
<tr>
<td>BAM-2991</td>
</tr>
<tr>
<td>BAM-2989</td>
</tr>
<tr>
<td>BAM-2983</td>
</tr>
<tr>
<td>BAM-2951</td>
</tr>
<tr>
<td>BAM-2946</td>
</tr>
<tr>
<td>BAM-2933</td>
</tr>
<tr>
<td>BAM-2852</td>
</tr>
<tr>
<td>BAM-2835</td>
</tr>
<tr>
<td>BAM-2803</td>
</tr>
<tr>
<td>BAM-2721</td>
</tr>
<tr>
<td>BAM-2713</td>
</tr>
<tr>
<td>BAM-2664</td>
</tr>
<tr>
<td>BAM-2612</td>
</tr>
<tr>
<td>BAM-2606</td>
</tr>
<tr>
<td>BAM-2605</td>
</tr>
<tr>
<td>BAM-2560</td>
</tr>
</tbody>
</table>
BAM-2541 "Comment Added" Notification Condition

BAM-2530 Scheduled Builds With no changes say, "This is an initial or manual build"

BAM-2494 Bamboo home is logged as blank in the logs during start-up

BAM-2479 Build Action status on "Currently Building ..." Screen doesn't update

BAM-2475 Bamboo re-index code, doesn't handle cases where the buildresults XML file is null

BAM-2466 Swap order of source directory and pattern columns in artifacts admin UI

BAM-2450 Email Bamboo admins when builds start failing due to issues with the repository.

BAM-2402 Reimplement "latest" artifact URLs as 302 redirects, rather than client-side refreshes

BAM-2399 Document the ability to link to the latest version of an artifact

BAM-2298 Allow for global variable substitution in the private key field, of the source repository page.

BAM-2227 Dependencies should pass down meta data about the dependency chain

BAM-2209 Only "stop build" is available from the Build Actions drop down menu even when the build has completed

BAM-2198 The URL filter fails if there are special characters in the testcase name

BAM-2168 Make Global System Variables accessible on the post actions page.

BAM-2080 Elastic Bamboo implementation, M2

BAM-2059 Directory clean on repository change may fail in distributed agent environment

BAM-2047 trivial typo on plan Notifications screen: 'commited' should have a double 't'

BAM-2018 Emails sent based on Build Notification

BAM-1999 Add artifacts section to build notification emails.

BAM-1991 "BUILD SUCCESSFUL" appearing shortly before a final "BUILD FAILED" message is misinterpreted as a successful build

BAM-1839 Agent bootstrapper restart if server goes down / communication error occurs

BAM-1831 HTML Emails

BAM-1736 Incorrect message in error page when configuring notification for invalid groups/users

BAM-1706 Ability to specify SMTP port in order to connect to SMTP server.

BAM-1592 Ability to disable/delete IM Server
BAM-1497  Next build arrow image has stray pixels  
Resolved

BAM-1413  Support for MSBuild  
Resolved

BAM-1403  Add more detail to the broken build e-mail - compiler error or broken test  
Resolved

BAM-1396  Have a time limit on builds  
Resolved

BAM-1394  Add "Build Actions" menu to Build Results page  
Resolved

BAM-1375  Labeller plugin assumes that build log contains instances of SimpleLogEntry rather than LogEntry  
Resolved

BAM-1360  Bamboo should explicitly build projects when user triggers a build ON manual build strategy  
Resolved

BAM-1355  Ability to remove Mail Server of IM Server Configuration  
Resolved

BAM-1299  Improve the Notification Framework in Bamboo, to register listeners  
Resolved

BAM-1177  Run builders on Amazon cloud  
Closed

BAM-1107  Broadcast IM comments  
Resolved

BAM-947  Dependant builds of the same source tree should build the same source as the parent  
Resolved

BAM-849  Ajaxy panels don't behave properly when a full page is returned from bamboo.  
Resolved

BAM-846  Customisable emails.  
Resolved

BAM-810  UI is left in a confused state on build failure  
Resolved

BAM-695  Build email doesn't contain enough information in the case of build failure  
Resolved

BAM-614  When a build fails I only want 1 error for artifacts  
Resolved

BAM-509  Timeout builds when they take too long  
Resolved

**Bamboo 2.2 Upgrade Guide**

**Upgrading from Bamboo 2.1 to 2.2**

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

> If you are using plugins, please make sure that your plugins are compile against 2.2 before upgrading.

Please also note the following important points:

1. **Remote agent changes**

   The default remote agent JAR shipped with Bamboo 2.2 has been upgraded. You will need to upgrade the remote agent JAR files for all of your remote agents, as described below.
If you want to use the new remote agent supervisor service wrapper, you will need to download and install the new remote agent JAR for all of your remote agents, as described in Step 1 of Bamboo Remote Agent Installation Guide.

If you do not want to use the new remote agent supervisor service wrapper (e.g. you have implemented your own service wrapper), you will need to download and install the legacy remote agent JAR for all of your remote agents, as described in Step 1 of Legacy Remote Agent Installation Guide. This remote agent JAR does not include the remote agent supervisor service wrapper.

Please note, your pre-Bamboo 2.2 remote agent JAR files will not work if you upgrade to Bamboo 2.2. You must install one of the two JARs described above.

2. Issue with remote agent home directory on Windows

An outstanding issue exists when installing remote agents on servers running Windows. If your remote agent home directory has space characters (e.g. /remote agent home), you will not be able to install remote agents. The process will crash when you attempt to run the remote agent (see BAM-3604 for further details).

We recommend that you remove all space characters from your remote agent home directory. Instructions for changing your remote agent home are described in the Bamboo Remote Agent Installation Guide.

3. Bamboo Developers — Changes for 2.2

If you are a Bamboo developer, please take note of the changes described in Changes for Bamboo 2.2 when upgrading to 2.2.

Upgrading from Bamboo prior to 2.1

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.

⚠️ Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.2.4 Release Notes

9 July 2009

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.2.4.

We have added an SVNkit java command-line client to Bamboo in this release (see BAM-4057). This client will provide valuable assistance in analysing Subversion-related connectivity issues. For detailed instructions on using this client, please see this FAQ.

This point release also contains 5 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.

Bamboo 2.2.4 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.2 yet?

Take a look at all the new features in the Bamboo 2.2 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.2.4 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (5 issues)</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-4055</td>
<td>Krystian Brazulewicz [Atlassian]</td>
<td>Brian Matzon</td>
<td>!</td>
<td>!</td>
<td>Resolved</td>
<td>Fixed 20C</td>
</tr>
</tbody>
</table>
Bamboo 2.2.4 Upgrade Guide

Upgrading from Bamboo 2.2.3 to 2.2.4

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.2.3 to 2.2.4.

Upgrading from Bamboo 2.1.x or earlier

In addition to the above, please read the Bamboo 2.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.2.3 Release Notes

Bamboo 4.0 has been released. Read the full Bamboo 4.0 Release Notes and Upgrade Guide. Don’t have Bamboo 4.0? Take a look at the features of Bamboo’s latest major version and try it out!

4 June 2009

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.2.3.

You can now disable your Bamboo server’s automatic capability detection upon agent restart, so that you do not have to reconfigure your agent capabilities every time you restart it. See this FAQ for details.

This point release also contains 14 bug fixes and improvements, including 2 critical fixes, which can be viewed below. Click a specific issue to see details of the fix, and to download patches where relevant.

Bamboo 2.2.3 is of course free to all customers with active Bamboo software maintenance.

Don’t have Bamboo 2.2 yet?

Take a look at all the new features in the Bamboo 2.2 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.2.3 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (14 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo removes quotes</td>
<td></td>
<td>Brydie</td>
<td>Ulrich Kuhnhardt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May 05, 2009</td>
<td>May 28, 2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3882</td>
<td>(*) from maven goal settings</td>
<td>McCoy [Atlassian]</td>
<td>[Atlassian]</td>
<td></td>
<td></td>
<td></td>
<td>Fixed</td>
<td>2009</td>
<td>2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3833</td>
<td>Perforce configuration doesn't use system variables during validation for client and port</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
<td>Fixed</td>
<td>Apr 22, 2009</td>
<td>Jun 02, 2009</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bamboo 2.2.3 Upgrade Guide**

**Upgrading from Bamboo 2.2.2 to 2.2.3**

The settings.xml file that is shipped with Bamboo no longer contains references to the Atlassian internal maven proxies. If you were relying on these proxies (for your EBS volumes), you can either edit the file to reference your own maven proxies or rely on the default ones.

Please also follow the Bamboo Generic Upgrade Guide.

**Upgrading from Bamboo 2.1.x or earlier**

In addition to the above, please read the Bamboo 2.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.2.2 Release Notes**

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

13 May 2009

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.2.2.

You can now control the logging for each of your remote agents independently from your Bamboo server in this release. For example, you may wish to change the logging on a particular remote agent to a more detailed level, if you are trying to troubleshoot a problem. For more information, please read Configuring Logging in Bamboo.

This point release also contains 9 bug fixes and improvements, including 3 critical fixes, which can be viewed below. Click a specific issue to see details of the fix, and to download patches where relevant.
Bamboo 2.2.2 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.2 yet?
Take a look at all the new features in the Bamboo 2.2 Release Notes and see what you are missing out on!

**Download Latest Version**

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.2.2 Upgrade Guide.

**Updates and Fixes in this Release**

<table>
<thead>
<tr>
<th>JIRA Issues (9 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-3774</td>
</tr>
<tr>
<td>BAM-3717</td>
</tr>
<tr>
<td>BAM-3703</td>
</tr>
<tr>
<td>BAM-3674</td>
</tr>
<tr>
<td>BAM-3643</td>
</tr>
<tr>
<td>BAM-3121</td>
</tr>
<tr>
<td>BAM-2965</td>
</tr>
<tr>
<td>BAM-2683</td>
</tr>
</tbody>
</table>

**Bamboo 2.2.2 Upgrade Guide**

**Upgrading from Bamboo 2.2.1 to 2.2.2**

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.2.1 to 2.2.2.

**Upgrading from Bamboo 2.1.x or earlier**

In addition to the above, please read the Bamboo 2.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.2.1 Release Notes**

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.

Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

18 March 2009
The Atlassian Bamboo team is proud to announce the release of **Bamboo 2.2.1**.

The .Net plugin has been upgraded in this release and should now work correctly. In addition, this point release contains more than 10 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.

Bamboo 2.2.1 is of course free to all customers with **active Bamboo software maintenance**.

**Don't have Bamboo 2.2 yet?**
Take a look at all the new features in the **Bamboo 2.2 Release Notes** and see what you are missing out on!

![Download Latest Version]

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the **Bamboo 2.2.1 Upgrade Guide**.

**Updates and Fixes in this Release**

**JIRA Issues (11 issues)**

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-3593</td>
<td>Help text for SSH to Instance link should be more informative if private key is missing</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Ulrich Kuhnhardt [Atlassian]</td>
<td>Fixed</td>
<td>Feb 25, 2009</td>
<td>Mar 0 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3590</td>
<td>Link from EB configuration to Agents configuration when Remote Agents are disabled uses Base URL, rather than relative URL</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Fixed</td>
<td>Feb 24, 2009</td>
<td>Feb 2 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3322</td>
<td>Gray out plan names if they are disabled in the plans matrix page</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Fixed</td>
<td>Dec 08, 2008</td>
<td>Jun 2 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3207</td>
<td>Post build regex pattern labeler should allow multiple capturing groups to be combined into one label</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Greg Baysden</td>
<td>Fixed</td>
<td>Nov 16, 2008</td>
<td>Mar 0 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2957</td>
<td>If perforce files end up open for edit, syncs dont work but Bamboo still builds</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Fixed</td>
<td>Jul 31, 2008</td>
<td>Mar 0 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1795</td>
<td>Using a url to browse for a user or author that does not exist results in freemarker errors</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Fixed</td>
<td>Oct 07, 2007</td>
<td>Sep 0 2009</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Upgrading from Bamboo 2.2 to 2.2.1

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.2 to 2.2.1.

Upgrading from Bamboo 2.1.x or earlier

In addition to the above, please read the Bamboo 2.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.1 Release Notes

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

5 August 2008

The Atlassian Bamboo team is proud to release Bamboo 2.1.

Bamboo 2.1 introduces a suite of new features which help you monitor the status of your JIRA issues and Bamboo builds side by side, when you integrate Bamboo with JIRA. This includes enhancements to issue viewing and linking functionality in Bamboo, enhanced views in both JIRA and Bamboo, and an easier setup process to integrate JIRA and Bamboo.

Upgrading to Bamboo 2.1 is free for all customers with active Bamboo software maintenance. The Bamboo plugin for JIRA is free for all customers.

Highlights of this release:

- Link Issues and Builds
- Specify the Issues that are Fixed by a Build
- Track the Builds for your Projects and Versions
- View Issues under Development
- Post Change Detection Plugin Point
- Plus over 30 fixes and improvements

Please keep logging your votes and issues. They help us decide what needs doing!

**Download Latest Version**

Upgrading to Bamboo 2.1

You can download Bamboo from the Atlassian website. To obtain the full benefits of this release, you will also need to install the latest JIRA Bamboo plugin, which is available for free here. If upgrading from a previous version, please read the Bamboo 2.1 Upgrade Guide.

Highlights of Bamboo 2.1

1

Link Issues and Builds

Bamboo now provides you with more ways to link JIRA issues to your builds, when you integrate JIRA with Bamboo. Bamboo will still automatically link an issue to your build when you specify it in your commit message, but it will now also pick up related JIRA issue keys that have been included in build comments and labels. If you want to manually link a particular JIRA issue to a build, we have included a new user interface to let you do that too.
Specify the Issues that are Fixed by a Build

We have also enhanced the issue to build linking to allow you to specify which issues are fixed by a build. This handy function will make it more convenient for your developers to flag when a particular JIRA issue is fixed in a project version. The build artifacts are then automatically made available as links from your JIRA issue, allowing you to download them straightaway in JIRA.

• Read more about editing issue links for a build.
Track the Builds for your Projects and Versions

Real-time tracking of the builds for a project or version has been included in this release of Bamboo. View the status of the builds for a project or a version at a glance in JIRA and drill down for details of each issue and build.

- Read more about viewing builds for your project and viewing builds for your project version.
View Issues under Development

Bamboo 2.1 now also provides you with a detailed view of the issues related to builds in Bamboo. Find out which issues are linked to completed builds, to track which issues were worked on recently.

- Read more about viewing issues linked to a build

Post Change Detection Plugin Point

As part of the Bamboo 2.1 release, we have extended our plugin framework by introducing the post change detection plugin point. This allows you to customise Bamboo actions before an build is queued, giving you greater flexibility to manage your build process.

Plus over 30 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (47 issues)</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-3293</td>
<td>Bamboo Upgrade Guide may call for reconfiguring external user repositories in wrong order</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-3055</td>
<td>External User Management is marked as readonly but it is still attempting to write to Crowd.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2992</td>
<td>Add the agent information of the agent executing the build to the Build Context.</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2974</td>
<td>A Version Tab Panel in JIRA Bamboo plugin for &quot;builds&quot;</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2973</td>
<td>A Project Tab Panel in JIRA Bamboo plugin for &quot;builds&quot;</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
BAM-2972  Plan Summary Tab for issues built in plan  
BAM-2971  Mark which builds have a JIRA issue is fixed in  
BAM-2970  Manually edit JIRA issue links with builds  
BAM-2936  Exports fail on Oracle  
BAM-2900  Builds list only displays relative date  
BAM-2897  JIRA content is not escaped in Bamboo  
BAM-2889  Intelligent cleaning of source directory when using “Force Clean Build”  
BAM-2888  RSS does not display the correct information when no changes found  
BAM-2887  Accesskey + S for submitting forms no longer worked in FireFox 3  
BAM-2874  Nullpointer Exception while exporing non-existant users.  
BAM-2872  Strange horizontal scrolling on build results pages  
BAM-2870  Upgrade Nant jar  
BAM-2869  Bamboo fails with OOM error, due to memory leak in xstream library  
BAM-2866  Changes to perforce client spec (without changing Bamboo) can cause odd behaviour  
BAM-2865  Bamboo integration with crowd doesnt play nicely with caches.  
BAM-2851  Add Pre-Build-queued action plugin point  
BAM-2850  Better Logic for deletion of build directory (for force clean build)  
BAM-2849  Warn users, they shouldn't point their perforce root directory to system folders.  
BAM-2807  A mechanism to allow remote agent capabilities to be supplied on start-up  
BAM-2749  Update version of Jira Soap Client Bamboo Uses  
BAM-2656  Add Types (fixes, relates to, etc) to Jira Issues  
BAM-2654  Testing JIRA setup tests saved setup rather than edited setup  
BAM-2649  Bamboo writes to the crowd directory when adding user aliases  
BAM-2645  Improve the way Bamboo reacts when a Jira Issue number doesn't exist.  
BAM-2631  Editing of Build to JIRA issue links
BAM-2604 Reindex message lies

BAM-2561 Use of $(bambooHome) in bamboo.cfg.xml

BAM-2501 Bundled JIRA Soap Service libraries don’t match jirasoapservice-v2 wsdl

BAM-2341 Shutdown logging should not be so verbose

BAM-2242 The JIRA Bamboo plugin shows incorrect version number.

BAM-2171 URL rewrite issue with JIRA's Bamboo plugin

BAM-1955 The shipped Bamboo source code assembly is missing the bamboo-ldap-web-app module

BAM-1937 REST API Login.action does not work if anonymous user access to plans is disabled.

BAM-1791 Indexing Rework

BAM-1619 Incorrect link generated by JIRA (Bamboo Integration) Portlet

BAM-1615 FileNotFoundException due to Bamboo fragmenting indexes in ../<Bamboo-Home>/index

BAM-1323 Indexes are slow to reindex

BAM-1228 Links to JIRA is potentially confusing

BAM-984 Repeated NullPointerExceptions on Shutdown; shutdown fails to complete

BAM-954 Changes View -> Clicking on the revision number of the file should lead to an annotated file view as of this revision

BAM-953 Changes tab should feature a Changeset link

BAM-142 ShutdownHook doesn't work correctly in Tomcat

**Bamboo 2.1 Upgrade Guide**

*Upgrading from Bamboo 2.0 to 2.1*

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

> If you are using plugins, please make sure that your plugins are compile against 2.1 before upgrading.

Please also note the following important points:

1. **Reindex during upgrade**

Please note that Bamboo will reindex when attempting to upgrade. For large instances this may take several hours. We strongly recommend that you do not upgrade during critical time periods for your development environment.

2. **Database changes**

Please note that during the upgrade, Bamboo will automatically remove the table `BUILDRESULTSUMMARY_JIRAISSUE` and replace it with
3. Issues upgrading Bamboo to version 2.1 with an Oracle database

There is an issue upgrading Bamboo with an Oracle database to Bamboo 2.1. Please upgrade to Bamboo 2.1.1, which contains the fix for this problem - read the release notes and upgrade guide for further details.

4. Bamboo Developers — Changes for 2.1

If you are a Bamboo developer, please take note of the changes described in Changes for Bamboo 2.1 when upgrading to 2.1.

Upgrading from Bamboo prior to 2.0

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.

Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.1.5 Release Notes

2 December 2008

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.1.5.

This point release contains more than 5 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.

Bamboo 2.1.5 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.1 yet?
Take a look at all the new features in the Bamboo 2.1 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.1.5 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (19 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-3241</td>
</tr>
<tr>
<td>BAM-3227</td>
</tr>
<tr>
<td>BAM-3208</td>
</tr>
<tr>
<td>BAM-3197</td>
</tr>
<tr>
<td>BAM-3154</td>
</tr>
<tr>
<td>BAM-3054</td>
</tr>
</tbody>
</table>

Add a directive to
<table>
<thead>
<tr>
<th>Ticket</th>
<th>Description</th>
<th>Assignee</th>
<th>Resolution</th>
<th>Status</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1986</td>
<td>Variable substitution should not assume that the result will be contained within a single command argument</td>
<td>Krystian Brazulewicz [Atlassian]</td>
<td></td>
<td></td>
<td>Dec 04, 2007</td>
<td>Nov 28, 2008</td>
</tr>
</tbody>
</table>

**Bamboo 2.1.5 Upgrade Guide**

**Upgrading from Bamboo 2.1.4 to 2.1.5**

Bamboo 2.1.5 contains a workaround to a Subversion-related issue in Bamboo 2.1.4, where any checked out code was automatically upgraded to SVN client format 1.5. If you want to prevent this automatic upgrade from occurring (e.g. you are using a pre-1.5 Subversion client to access code checked out by Bamboo), you can disable this automatic upgrade of checked out code by running Bamboo with the following system property:

```
-Dbamboo.svn.compatibility.14=true
```
Please also follow the Bamboo Generic Upgrade Guide.

**Upgrading from Bamboo 2.0.x or earlier**

In addition to the above, please read the Bamboo 2.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.1.4 Release Notes**

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

11 November 2008

The Atlassian Bamboo team is proud to announce the release of **Bamboo 2.1.4**.

This point release also over 5 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.

---

**For Bamboo 2.1.x users**

If you are currently using Bamboo 2.1.x, we **strongly recommend** that you upgrade to Bamboo 2.1.4. This release contains an important fix to an LDAP issue (**BAM-3180**) that may cause problems for your system.

Bamboo 2.1.4 is of course free to all customers with **active Bamboo software maintenance**.

**Don't have Bamboo 2.1 yet?**

Take a look at all the new features in the **Bamboo 2.1 Release Notes** and see what you are missing out on!

cta: https://www.atlassian.com/software/bamboo/download

**Download Latest Version**

---

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.1.4 Upgrade Guide.

**Updates and Fixes in this Release**

<table>
<thead>
<tr>
<th>JIRA Issues (9 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>0x0</td>
</tr>
<tr>
<td>0x0</td>
</tr>
</tbody>
</table>
Bamboo 2.1.4 Upgrade Guide

Upgrading from Bamboo 2.1.3 to 2.1.4

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.1.3 to 2.1.4.

Bamboo compatibility with Subversion pre-1.5

We have upgraded the SVNKit library in Bamboo. As a result, any source code checked out by Bamboo will be automatically upgraded to be compatible with Subversion 1.5. This does not adversely affect any pre-1.5 Subversion servers. However, if you use a pre-1.5 Subversion client to access code checked out by Bamboo, then any Bamboo builds on that code may fail. Please refer to BAM-3241 for further details.

Please avoid using a pre-1.5 Subversion client to access code checked out by Bamboo.

Bamboo 2.1.5 contains a workaround to this issue. We highly recommend that you upgrade your Bamboo version.

Upgrading from Bamboo 2.0.x or earlier

In addition to the above, please read the Bamboo 2.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.1.3 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.

Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

16 October 2008

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.1.3.

This point release also includes over 10 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.

Bamboo 2.1.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.1 yet?

Take a look at all the new features in the Bamboo 2.1 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.1.3 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues</th>
<th>(16 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue Key</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BAM-3087</td>
<td>Sentence incomplete in Plan Details -&gt; Build Plan Key description</td>
</tr>
</tbody>
</table>

**Bamboo 2.1.3 Upgrade Guide**

**Upgrading from Bamboo 2.1.2 to 2.1.3**

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.1.2 to 2.1.3.

**Upgrading from Bamboo 2.0.x or earlier**

In addition to the above, please read the Bamboo 2.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.1.2 Release Notes**
24 September 2008
The Atlassian Bamboo team is proud to announce the release of **Bamboo 2.1.2**.

We have improved the availability and reliability of remote agents in this release, by adding a failover to reconnect agents when the network drops out. You should also notice a significant performance improvement in Bamboo 2.1.2, if you are using a Perforce repository, as we have dramatically reduced the CPU usage (60%-70% less usage) for Perforce polling.

This point release also includes over 20 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.

Bamboo 2.1.2 is of course free to all customers with active Bamboo software maintenance.

**Don’t have Bamboo 2.1 yet?**
Take a look at all the new features in the [Bamboo 2.1 Release Notes](#) and see what you are missing out on!

![Download Latest Version](#)

### Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the [Bamboo 2.1.2 Upgrade Guide](#).

### Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-3066</td>
<td>Bamboo server hangs if Container initialization fails</td>
<td>Lucas Guminski</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Sep 16, 2008</td>
</tr>
<tr>
<td>BAM-3047</td>
<td>Support for distinct client and server broker URLs</td>
<td>Adrian Hempel</td>
<td>Adrian Hempel</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Sep 08, 2008</td>
</tr>
<tr>
<td>BAM-3045</td>
<td>Move Plan Wizard has wrong information and unresolved links / actions</td>
<td>Brydie McCoy</td>
<td>Ulrich Kuhnhardt</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Duplicate" /></td>
<td></td>
<td></td>
<td>Sep 07, 2008</td>
</tr>
<tr>
<td>BAM-3042</td>
<td>Valid Broker URLs are ruined by attempt to add &quot;maxInactivityDuration&quot;</td>
<td>Adrian Hempel</td>
<td>Adrian Hempel</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Sep 04, 2008</td>
</tr>
<tr>
<td>BAM-3039</td>
<td>Upgrade Bamboo's crowd-client library</td>
<td>Mark Chaimgungkalanont</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Sep 03, 2008</td>
</tr>
<tr>
<td>BAM-3008</td>
<td>Bamboo throws an exception, if a user tries to view a comment for a buildresult that doesn't exist</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 29, 2008</td>
</tr>
<tr>
<td>BAM-3007</td>
<td>System Info truncates memory information rather than rounds it</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 28, 2008</td>
</tr>
<tr>
<td>BAM-3004</td>
<td>Exception thrown if Crowd is not writable and Read only external user management flag not checked</td>
<td>Mark Chaimgungkalanont</td>
<td>Edwin Wong</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 27, 2008</td>
</tr>
<tr>
<td>BAM-3001</td>
<td>Bamboo shouldn't try to add users in Crowd, when it doesn't have directory rights to add principals</td>
<td>Mark Chaimgungkalanont</td>
<td>Mick van der Most van Spijk</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 25, 2008</td>
</tr>
<tr>
<td>BAM-3000</td>
<td>Bamboo sends out false notifications, if the build has failed after X builds</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 25, 2008</td>
</tr>
<tr>
<td>BAM-2999</td>
<td>Some builds assigned to a remote agent behind a firewall are never built</td>
<td>Adrian Hempel</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 21, 2008</td>
</tr>
<tr>
<td>BAM-2987</td>
<td>Can’t assign LDAP users against local Bamboo groups.</td>
<td>Mark Chaimgungkalanont</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 13, 2008</td>
</tr>
<tr>
<td>BAM-2980</td>
<td>Change the perforce polling command to use p4 changes -m2</td>
<td>Mark Chaimgungkalanont</td>
<td>Ajay Sridhar</td>
<td><img src="#" alt="Resolved" /></td>
<td><img src="#" alt="Fixed" /></td>
<td></td>
<td></td>
<td>Aug 12, 2008</td>
</tr>
</tbody>
</table>

Mark Chaimungkalanont on Jul 29, 2008
<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Assignee</th>
<th>Commenter</th>
<th>Resolution</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2947</td>
<td>Failing to pump input stream</td>
<td>Chaimungkalanont</td>
<td>Joshua Grigonis</td>
<td>Fixed</td>
<td>2008</td>
</tr>
<tr>
<td>BAM-2943</td>
<td>Perforce session expiry not detected by Perforce library</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td>Jul 28, 2008</td>
</tr>
<tr>
<td>BAM-2937</td>
<td>Perforce polling uses up too much CPU</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>Resolved</td>
<td>Jul 25, 2008</td>
</tr>
<tr>
<td>BAM-2892</td>
<td>HTML is not escaped in RSS title</td>
<td>Lucas Guminski</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td>Jul 21, 2008</td>
</tr>
<tr>
<td>BAM-2881</td>
<td>If comment for commit contains xml tag then api/rest/getBuildResultsDetails.action return invalid XML</td>
<td>Lucas Guminski</td>
<td>Jacek Jaroczynski</td>
<td>Resolved</td>
<td>Jul 17, 2008</td>
</tr>
<tr>
<td>BAM-2806</td>
<td>“Triggered” is spelt incorrectly</td>
<td>Brydie McCoy</td>
<td>Stuart Miller</td>
<td>Fixed</td>
<td>Jun 22, 2008</td>
</tr>
<tr>
<td>BAM-2750</td>
<td>Excessive error logging when does not parse testing-results.xml correctly</td>
<td>Mark Chaimungkalanont</td>
<td>Eric Dalglish</td>
<td>Resolved</td>
<td>Jun 12, 2008</td>
</tr>
<tr>
<td>BAM-2691</td>
<td>Warning when moving plans not updated with 2.0 changes (link + text)</td>
<td>Brydie McCoy</td>
<td>Kirk Wylie</td>
<td>Fixed</td>
<td>Jun 03, 2008</td>
</tr>
<tr>
<td>BAM-2651</td>
<td>Reorganise Plug-in developer documentation</td>
<td>Brydie McCoy</td>
<td>Adrian Hempel</td>
<td>Resolved</td>
<td>May 23, 2008</td>
</tr>
<tr>
<td>BAM-2050</td>
<td>The anonymous user signup page doesn't validate user's email address/Full name</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Resolved</td>
<td>Jan 03, 2008</td>
</tr>
<tr>
<td>BAM-2030</td>
<td>Full names and Email addresses are set to null, if editing local users with external user management enabled.</td>
<td>Mark Chaimungkalanont</td>
<td>Ajay Sridhar</td>
<td>Resolved</td>
<td>Dec 17, 2007</td>
</tr>
<tr>
<td>BAM-1963</td>
<td>The ViewGroups page can only show a maximum of 100 groups, when integrated with LDAP</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Resolved</td>
<td>Nov 28, 2007</td>
</tr>
<tr>
<td>BAM-1722</td>
<td>The ViewLists page does not list all available groups.</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Resolved</td>
<td>Sep 23, 2007</td>
</tr>
</tbody>
</table>

**Bamboo 2.1.2 Upgrade Guide**

**Upgrading from Bamboo 2.1.1 to 2.1.2**

Please follow the Bamboo Generic Upgrade Guide.

- No additional upgrade tasks are required to upgrade from Bamboo 2.1.1 to 2.1.2.

**Upgrading from Bamboo 2.0.x or earlier**

In addition to the above, please read the Bamboo 2.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.1.1 Release Notes**

- Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.
  Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

**12 August 2008**

The Atlassian Bamboo team is proud to announce the release of **Bamboo 2.1.1**. This point release includes fixes for two critical issues (refer to the JIRA issues below for details).
Bamboo 2.1.1 is of course free to all customers with active Bamboo software maintenance.

Don’t have Bamboo 2.1 yet?
Take a look at all the new features in the Bamboo 2.1 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.1.1 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (2 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

Bamboo 2.1.1 Upgrade Guide

Upgrading from Bamboo 2.1 to 2.1.1

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.1 to 2.1.1.

Upgrading from Bamboo 2.0.x or earlier

In addition to the above, please read the Bamboo 2.1 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.0 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.

Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!

Atlassian Software Systems presents Bamboo 2.0

Upgrading to Bamboo 2.0 is free for all customers with active Bamboo software maintenance as at 14th April 2008. This release introduces the ability to run distributed builds. You will find this particularly useful if you need to run your builds in different geographic locations, or on different platforms. Simply install the new Bamboo Agent on your additional build servers, and your main Bamboo 2.0 server will be able to manage them. We have also provided a number of plugin points in case you need to control your distributed builds programmatically.

Highlights of Bamboo 2.0:

- Distributed builds
- Capability matching
- Memory usage improvements
- Parallel VCS updates and checkouts
- Ability to force a ‘clean build’
- Quiet Period functionality supported for Subversion & Perforce
- Bamboo Plugin for Confluence
- Support for Oracle and MS SQL Server databases
- Status Summary screens
- Plus over 160 other fixes and improvements

Thank you for your feedback:
Your votes and issues help us keep improving our products, and are much appreciated.

Upgrading to Bamboo 2.0

Bamboo 2.0 can be downloaded from the Bamboo Download Centre. Before upgrading, please refer to the Bamboo 2.0 Upgrade Guide.

Highlights of Bamboo 2.0

1

Distributed builds

In response to the most popular feature on your wish-list, Bamboo 2.0 introduces agents — that is, services which execute builds. These can either run on the Bamboo server ('local agents') or on other machines ('remote agents'), which is particularly useful if you need to run your builds in different geographic locations, or on different platforms.

A single build queue manages the distribution of builds to appropriate agents, using capability matching (see below).

The activity of all agents can be seen on the Bamboo dashboard:
Capability matching

To enable you to control exactly which agents may execute builds for particular plans, Bamboo 2.0 introduces capability matching:

- A capability is a feature of an agent. A capability can be a:
  - builder (e.g. Maven)
  - JDK
  - custom capability (a key-value property which defines a particular characteristic of an agent, e.g. 'operating.system=WindowsXP' or 'fast.builds=true')
  - Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

See Configuring Capabilities.

- A requirement is an agent capability required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See Specifying a Plan's Capability Requirements.

For more details please see these diagrams.
Note that for ease of conversion, the Bamboo 2.0 upgrade process will automatically create appropriate agent capabilities and assign appropriate requirements to all your pre-existing build plans (see the Bamboo 2.0 Upgrade Guide).

3

Memory usage improvements

The underlying engine for Bamboo has been revamped to decrease memory usage. You will notice a distinct improvement in the performance of your builds, especially if you have very large logs.

4

Parallel VCS updates and checkouts

No more waiting! Plans can now perform checkouts and updates to your version control system in parallel, rather than serially. Hence, the time taken to run plans will be improved.

5

Ability to force a 'clean build'

You can now instruct Bamboo to delete the old working files and perform a new checkout of the entire source code directory, before commencing a build.

See the documentation for more details.

6

Quiet Period functionality supported for Subversion & Perforce

By popular request, Quiet Period parameters can now be specified for Subversion and Perforce when configuring a source repository for a build plan. You can choose to set how long Bamboo should wait after a commit before triggering a build, and the number of times it retries before initiating a build. Read more about configuring Subversion and Perforce source repositories.

7

Bamboo Plugin for Confluence

Atlassian brings collaboration to the next level with the introduction of the Bamboo plugin for Confluence. Here's some of the build information that your wiki users will be able to have at their fingertips:

- the most recent status of any given build plan.
- the current status of all builds in a project.
- the recent build history of a plan.
- the recent build history of a user across all projects.
- the recent build history of all plans in a project.
- Bamboo charts, including duration of builds, build failures, numbers of test, percentage of test failures and more!

Read more about the Bamboo Plugin for Confluence.
Support for Oracle and MS SQL Server databases

By popular request, Bamboo's supported databases now include Oracle and MS SQL Server.

Status Summary screens

See the status of your builds at a glance! Set up a build status monitor for your development team and display Bamboo's new status summary screens. These screens show the status of your builds in a color-coded and easy to view format.

Plus over 160 other fixes and improvements

See them here.

Bamboo 2.0 Upgrade Guide

On this page:

- Upgrading from Bamboo version 1.1.x or earlier to 2.0
  - Upgrading from Bamboo 1.1.x
  - Upgrading from Bamboo 1.0.x
  - Upgrading from Bamboo 1.2.x to 2.0
    - 1. Adding a Broker URL property.
    - 2. Changes to Server Configuration
      - JDK support
      - Database changes
      - Plugins
    - 3. Changes to Build Queues and Build Plans
      - Conversion of Build Queues to Agents
      - Conversion of Builders to Capabilities
      - Conversion of JDKs to Capabilities
    - 4. Changes to Repositories
      - Conversion of Perforce P4 Client Application Location to a Capability
      - Minimum repository version requirement for CVS and Perforce
    - 5. Changes to Jetty (Bamboo Standalone Only)
  - Upgrading from Bamboo 2.0 Beta to 2.0

Upgrading from Bamboo version 1.1.x or earlier to 2.0
If you are using a version of Bamboo prior to version 1.2, you will need to upgrade Bamboo to version 1.2 before you can upgrade to version 2.0. Note that the upgrade process from version 1.0.x is different from the upgrade process from version 1.1.x. Please follow the appropriate instructions below:

**Upgrading from Bamboo 1.1.x**

You will need to:

1. Upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
2. Then upgrade to Bamboo 2.0, as per the 'Upgrading from Bamboo 1.2.x to 2.0' instructions below.

**Upgrading from Bamboo 1.0.x**

You will need to:

1. Upgrade to 1.1.2 first — please see the Bamboo 1.1.2 Upgrade Guide. (This step is necessary as there is an issue with the upgrade process from the 1.0.x series that we're currently looking into.)
2. Then upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
3. Then upgrade to Bamboo 2.0, as per the 'Upgrading from Bamboo 1.2.x to 2.0' instructions below.

---

Please read this if you have a datasource configured

Currently, Bamboo upgrade tasks fail if user has a datasource configured. To get around this issue please follow instructions on this page before upgrading to Bamboo 2.0

---

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

You will need to re-index your Bamboo instance post upgrade, please consult step 6 of the Bamboo Generic Upgrade Guide for more details.

**Upgrading from Bamboo 1.2.x to 2.0**

1. **Adding a Broker URL property.**

Bamboo uses a messaging broker to communicate with its remote build agents. To ensure this works properly, a URL must be specified. This URL is where Bamboo will set up its embedded broker. Remote agents will also be provided with this URL on startup.

To specify the broker URL, please add a `<property name="bamboo.jms.broker.uri">tcp://HOSTNAME:54663</property>` in your `bamboo.cfg.xml` file, located inside the Bamboo home directory. For example:

```xml
<property name="bamboo.jms.broker.uri">tcp://HOSTNAME:54663</property>
```

where HOSTNAME is the canonical name of your Bamboo server.

Please note, as remote agents use this URL to communicate to the server, you should take care not to specify localhost as the host name in the broker URL.

If no broker URL is found in `bamboo.cfg.xml`, Bamboo will default the broker URL to `tcp://HOSTNAME:54663` in the `bamboo.cfg.xml` file, as seen in the example above. Bamboo will also append the parameter `wireFormat.maxInactivityDuration=0` by default to any broker URL coming from `bamboo.cfg.xml`.

2. **Changes to Server Configuration**

   **JDK support**

Bamboo 2.0 requires JDK 1.5 (i.e. JDK 1.4 is no longer supported). Please note that this does not affect the actual builds: it is only the Bamboo server itself that must be running JDK 1.5.

   **Database changes**

The release of 2.0 will include some changes to column names in the database as follows:
In the BUILD_DEFINITION table, the column XML_DATA will be changed to XML_DEFINITION_DATA.
In the BUILDRESULTSUMMARY_CUSTOMDATA table, the column CUSTOM_INFO_DATA will be changed to CUSTOM_INFO_VALUE.

These fields have also had types changed to CLOB to increase their maximum lengths.

**Plugins**

If you are using external or custom plugins, please make sure that your plugins compile against Bamboo 2.0 before upgrading.

We've made significant changes to the internals of the application for Bamboo 2.0. If you've installed an external plugin for 1.2.4, it's likely that it will be broken. Please take care when upgrading.

3. Changes to Build Queues and Build Plans

Bamboo 2.0 introduces the concepts of agents and capabilities. To preserve the functionality of your existing plans, JDKs, Builders and Build Queues, the following will automatically happen during the upgrade:

**Conversion of Build Queues to Agents**

Prior to Bamboo 2.0, you could have multiple build queues. In Bamboo 2.0, there is now only one build queue, but multiple agents (see diagram).

As part of the upgrade process,

- Each of your build queues will be converted to a **local agent**.
- If, prior to the upgrade, the build queue accepted builds from all plans, the agent will be given the following **capability** (and every plan will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: ALLOW_ANY_BUILDS
- Or if, prior to the upgrade, the build queue only accepted builds from specific plans, the agent will be given the following **capability** (and the relevant plans will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: <name of old queue>

If you wish to change this after the upgrade, please see Configuring Agents and Capabilities and Specifying a Plan's Capability Requirements.

**Conversion of Builders to Capabilities**

Prior to Bamboo 2.0, your builders (e.g. Maven) were defined globally. In Bamboo 2.0, builders are now defined as agent capabilities and specified as plan requirements.

As part of the upgrade process,

- Each of your builders will be converted to a **local server capability** (that is, it will apply to every local agent).
- Every plan will continue to have the same builder that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

**Conversion of JDKs to Capabilities**

Prior to Bamboo 2.0, your JDKs (e.g. JDK 1.5) were defined globally. In Bamboo 2.0, JDKs are now defined as agent capabilities and specified as plan requirements.

As part of the upgrade process,

- Each of your JDKs will be converted to **local server capabilities** (that is, it will apply to every local agent).
- Upon conversion, the labels of each of your JDKs will upgraded to the Bamboo 2.0 JDK label format, (i.e. 'JDK 9.9.9_99').
- Upon conversion, two more generic versions of the labels will be created for each JDK, (i.e. 'JDK 9.9' and 'JDK').
- Every plan will have its requirements upgraded, to keep the association with the same JDK that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

4. Changes to Repositories

Bamboo 2.0 introduces the concepts of agents and capabilities. To preserve the functionality of your existing Repositories, the following will automatically happen during the upgrade:
Conversion of Perforce P4 Client Application Location to a Capability

With the introduction of remote agents in Bamboo 2.0, the location of the Perforce P4 client application now needs to be specified as a capability. To create build plans using Perforce as repository, a local server capability must be created for the P4 client application location. In addition, agent-specific remote capabilities must be created for each remote agent using Perforce.

As part of the upgrade process,

- A local server Perforce capability will be created for the Perforce P4 client application location. The upgrade task reads this information from the system's environment variables. If the Perforce P4 client application location has not been specified as an environment variable, the local server capability will need to be set up manually.

The upgrade task will not create agent-specific Perforce capabilities for any remote agents. These capabilities will need to be set up manually.

Please see Configuring a new Perforce Capability for further details on creating Perforce capabilities.

Minimum repository version requirement for CVS and Perforce

Due to internal changes, Bamboo is no longer compatible with the following:

- CVS server version 1.11.1p2 and below.
- Perforce server version 2005.1 and below.

If you are planning on upgrading to Bamboo 2.0, please consider upgrading your repository server version.

5. Changes to Jetty (Bamboo Standalone Only)

Jetty has been upgraded from version 5 to version 6 in Bamboo 2.0. This means that if you have set up Bamboo to use the jetty.xml file, it will no longer work. You will need to update the configuration to be compatible with Jetty 6. An example Jetty 6 jetty.xml file can be found at Getting Bamboo Standalone to use the jetty.xml file.

Upgrading from Bamboo 2.0 Beta to 2.0

If you are already using the latest Bamboo 2.0 Beta, no additional upgrade tasks are required. Your Beta license key will continue to function until it expires. We encourage you to consider purchasing a license, if you wish to continue using Bamboo 2.0.

Bamboo 2.0.6 Release Notes

Bamboo 4 has been released. Read the full Bamboo 4 Release Notes and Upgrade Guide. Don't have Bamboo 4? Take a look at the features of Bamboo's latest major version and try it out!

8 July 2008
Atlassian Software Systems is proud to announce the release of Bamboo 2.0.6. This point release includes over 10 major 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.

Bamboo 2.0.6 is of course free to all customers with active maintenance.

This release includes significant fixes to issues with remote agents.

Don't have Bamboo 2.0 yet?
Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.0.6 Upgrade Guide.

Updates and Fixes in this Release

<p>| JIRA Issues (15 issues) |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Assignee</th>
<th>Status</th>
<th>Fixed On</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2846</td>
<td>Cannot select any test on the live activity screen</td>
<td>Chaimungkalanont</td>
<td>Resolved</td>
<td>Jul 03, 2008, Jul 03, 2008</td>
</tr>
<tr>
<td>BAM-2840</td>
<td>Auto detection of SVN problems which requires a clean</td>
<td>Chaimungkalanont</td>
<td>Resolved</td>
<td>Jul 01, 2008, Jul 02, 2008</td>
</tr>
<tr>
<td>BAM-2830</td>
<td>Perforce sync command is memory inefficient</td>
<td>Chaimungkalanont</td>
<td>Resolved</td>
<td>Jun 30, 2008, Jul 06, 2008</td>
</tr>
<tr>
<td>BAM-2818</td>
<td>Errors causes agents to stop</td>
<td>Sridhar</td>
<td>Resolved</td>
<td>Jun 25, 2008, Sep 01, 2009</td>
</tr>
<tr>
<td>BAM-2810</td>
<td>Errors on local agent do not get reported</td>
<td>Sridhar</td>
<td>Resolved</td>
<td>Jun 23, 2008, Jul 01, 2008</td>
</tr>
<tr>
<td>BAM-2755</td>
<td>Ant output in Activity log is triple spaced in Firefox</td>
<td>Tim Whittington</td>
<td>Resolved</td>
<td>Jun 12, 2008, Jul 18, 2008</td>
</tr>
<tr>
<td>BAM-2743</td>
<td>Remote agents may return even when already marked as dead</td>
<td>Chaimungkalanont</td>
<td>Resolved</td>
<td>Jun 10, 2008, Sep 01, 2009</td>
</tr>
<tr>
<td>BAM-2621</td>
<td>The &quot;clear error log&quot; button on dashboard, reverts dashboard without style sheets</td>
<td>Sridhar</td>
<td>Resolved</td>
<td>May 13, 2008, Jul 01, 2008</td>
</tr>
<tr>
<td>BAM-2452</td>
<td>Build continues after agent lost and recontacted server</td>
<td>Barney</td>
<td>Resolved</td>
<td>Apr 06, 2008, Oct 02, 2008</td>
</tr>
<tr>
<td>BAM-2412</td>
<td>Build continues after agent lost and recontacted server</td>
<td>Sridhar</td>
<td>Resolved</td>
<td>Mar 27, 2008, Sep 01, 2009</td>
</tr>
<tr>
<td>BAM-1937</td>
<td>REST API Login.action does not work if anonymous user access to plans is disabled.</td>
<td>Hempel</td>
<td>Resolved</td>
<td>Nov 15, 2007, Jun 26, 2008</td>
</tr>
<tr>
<td>BAM-1568</td>
<td>Jabber Details and Notification Preferences are not exported for LDAP and Crowd Users</td>
<td>McCoy</td>
<td>Resolved</td>
<td>Aug 07, 2007, Aug 21, 2008</td>
</tr>
</tbody>
</table>

**Bamboo 2.0.6 Upgrade Guide**

*Upgrading from Bamboo 2.0.x to 2.0.6*  
Please follow the Bamboo Generic Upgrade Guide.

Note: No further upgrade tasks are required to upgrade from Bamboo 2.0.x to 2.0.6, but please ensure that you have read the Bamboo 2.0.1 Upgrade Guide which contains information on minor database changes.
**Upgrading from Bamboo 1.2.x or earlier**

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.0.5 Release Notes**

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

25 June 2008

Atlassian Software Systems is proud to announce the release of Bamboo 2.0.5. This point release includes 5 major 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.

Bamboo 2.0.5 is of course free to all customers with active maintenance.

**Don't have Bamboo 2.0 yet?**

Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

**Download Latest Version**

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.0.5 Upgrade Guide.

**Updates and Fixes in this Release**

<table>
<thead>
<tr>
<th>JIRA Issues (4 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
<th>Due</th>
</tr>
</thead>
</table>

**Bamboo 2.0.5 Upgrade Guide**

**Upgrading from Bamboo 2.0.x to 2.0.5**

Please follow the Bamboo Generic Upgrade Guide.

We have made additional optimisation improvements for SVN support in this release. To get these improvements, you will also need to upgrade your remote agents' startup jar with the latest version from the Bamboo server, as follows:

1. Upgrade your Bamboo server to version 2.0.5.
2. Shut down all your remote agents.
3. Replace the start up jar on each of your remote agents with the latest version from the Bamboo server. This is available from Administration -> Agents -> Install Remote Agent.
4. Start your Bamboo remote agents.

⚠️ No further upgrade tasks are required to upgrade from Bamboo 2.0.x to 2.0.5, but please ensure that you have read the Bamboo 2.0.1 Upgrade Guide which contains information on minor database changes.

**Upgrading from Bamboo 1.2.x or earlier**

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the
Bamboo 2.0.4 Release Notes

**Bamboo 4.2 has been released. Read the full Bamboo 4.2 Release Notes and Upgrade Guide. Don't have Bamboo 4.2? Take a look at the features of Bamboo's latest major version and try it out!**

11 June 2008

Atlassian Software Systems is proud to announce the release of Bamboo 2.0.4. This point release more than 15 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.

Bamboo 2.0.4 is of course free to all customers with active maintenance.

**Don't have Bamboo 2.0 yet?**
Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

*Download Latest Version*

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.0.4 Upgrade Guide.

**Updates and Fixes in this Release**

<table>
<thead>
<tr>
<th>JIRA Issues (16 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
</tr>
</thead>
</table>
Bamboo 2.0.4 Upgrade Guide

Upgrading from Bamboo 2.0.x to 2.0.4

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0.x to 2.0.4, but please ensure that you have read the Bamboo 2.0.1 Upgrade Guide which contains information on minor database changes.

Upgrading from Bamboo 1.2.x or earlier

Customers using PostgreSQL
Due to a recent fix to our 2.0.4 upgrade tasks, if you are using a PostgreSQL database the upgrade will fail. Please follow the instructions in this JIRA issue prior to running Bamboo 2.0.4.

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.0.3 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.

Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

23 May 2008
Atlassian Software Systems is proud to announce the release of Bamboo 2.0.3. This point release includes six 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.

Bamboo 2.0.3 is of course free to all customers with active maintenance.

Don't have Bamboo 2.0 yet?
Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.0.3 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (9 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>BAM-2572</td>
</tr>
<tr>
<td>JIRA Issue</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>BAM-2548</td>
</tr>
<tr>
<td>BAM-2505</td>
</tr>
<tr>
<td>BAM-2468</td>
</tr>
<tr>
<td>BAM-2296</td>
</tr>
<tr>
<td>BAM-1913</td>
</tr>
</tbody>
</table>

### Bamboo 2.0.3 Upgrade Guide

**Upgrading from Bamboo 2.0.x to 2.0.3**

Please follow the Bamboo Generic Upgrade Guide.

> No additional upgrade tasks are required to upgrade from Bamboo 2.0.x to 2.0.3, but please ensure that you have read the Bamboo 2.0.1 Upgrade Guide which contains information on minor database changes.

**Upgrading from Bamboo 1.2.x or earlier**

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

### Bamboo 2.0.2 Release Notes

**Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.**

*Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!*

**5 May 2008**

Atlassian Software Systems is proud to announce the release of Bamboo 2.0.2. This point release includes five 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.

Bamboo 2.0.2 is of course free to all customers with active maintenance.

Significant fixes for this point release include, a resolution to a memory leak problem caused by Subversion Client Manager (see BAM-2543) and a fix to an SVN Externals exception (see BAM-2544).

**Don't have Bamboo 2.0 yet?**

Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

**Download Latest Version**

### Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.0.2 Upgrade Guide.

### Updates and Fixes in this Release

**JIRA Issues (6 issues)**
Bamboo 2.0.2 Upgrade Guide

Upgrading from Bamboo 2.0.x to 2.0.2

Please follow the Bamboo Generic Upgrade Guide. No additional upgrade tasks are required to upgrade from Bamboo 2.0.x to 2.0.2, but please ensure that you have read the Bamboo 2.0.1 Upgrade Guide which contains information on minor database changes.

Upgrading from Bamboo 1.2.x or earlier

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.0.1 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!

29 April 2008
Atlassian Software Systems is proud to announce the release of Bamboo 2.0.1. This point release includes over 15 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.

Bamboo 2.0.1 is of course free to all customers with active maintenance.

Don’t have Bamboo 2.0 yet?
Take a look at all the new features in the Bamboo 2.0 Release Notes and see what you are missing out on!

Download Latest Version

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.0.1 Upgrade Guide.
## Bamboo 2.0.1 Upgrade Guide

### Upgrading from Bamboo 2.0 to 2.0.1

#### JIRA Issues (18 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2515</td>
<td>BAM-2515</td>
<td>Can't upgrade to Bamboo 2.0 or save plan configuration, due to Bamboo trying to write invalid SVN external definition to XML</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Apr 22, 2008</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-2514</td>
<td>BAM-2514</td>
<td>Upgrading 'allowed builds' to capability/requirements match doesn't work properly</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Apr 22, 2008</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-2488</td>
<td>BAM-2488</td>
<td>Bamboo should not run post-build actions and send out notifications, when it fails to save the buildresultsummary</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Apr 16, 2008</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-2487</td>
<td>BAM-2487</td>
<td>The commit_comment field in the COMMIT table is restricted to 4000 characters</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Apr 16, 2008</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-2428</td>
<td>BAM-2428</td>
<td>Allow configuration of SVNKit to spool / not spool</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Apr 01, 2008</td>
<td>N 2</td>
</tr>
<tr>
<td>BAM-2370</td>
<td>BAM-2370</td>
<td>Mechanism to control the agent heart beat interval.</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Jason Davis</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Mar 12, 2008</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-1869</td>
<td>BAM-1869</td>
<td>Graphs should not show data points for time periods with no successful builds</td>
<td>Unassigned</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Oct 25, 2007</td>
<td>N 2</td>
</tr>
<tr>
<td>BAM-1361</td>
<td>BAM-1361</td>
<td>Main clover tab get confused over weekends</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Robert Macaulay</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 19, 2007</td>
<td>A 2</td>
</tr>
<tr>
<td>BAM-945</td>
<td>BAM-945</td>
<td>Clover charts are inaccurate on days that builds did not occur</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Chris Beams</td>
<td>⚠️</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Mar 01, 2007</td>
<td>A 2</td>
</tr>
</tbody>
</table>

### Updates and Fixes in this Release

- **BAM-1361**: Main clover tab get confused over weekends
  - **Assignee**: Mark Chaimungkalanont
  - **Reporter**: Robert Macaulay
  - **Status**: Resolved
  - **Resolution**: Fixed
  - **Created**: Jun 19, 2007

- **BAM-2482**: Bamboo 2.0 does not support perforce versions BEFORE 2005.1
  - **Assignee**: Brydie McCoy
  - **Reporter**: Brydie McCoy
  - **Status**: Resolved
  - **Resolution**: Fixed
  - **Created**: Apr 15, 2008

- **BAM-2428**: Allow configuration of SVNKit to spool / not spool
  - **Assignee**: Mark Chaimungkalanont
  - **Reporter**: Mark Chaimungkalanont
  - **Status**: Resolved
  - **Resolution**: Fixed
  - **Created**: Apr 01, 2008

- **BAM-945**: Clover charts are inaccurate on days that builds did not occur
  - **Assignee**: Mark Chaimungkalanont
  - **Reporter**: Chris Beams
  - **Status**: Resolved
  - **Resolution**: Fixed
  - **Created**: Mar 01, 2007
Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 to 2.0.1.

Database Changes
Please note, we are replacing the commit_comment field in the user_commit table with a new commit_comment_clob field to allow for longer commit messages. This change will be made automatically and will not affect the user interface. However, please be aware of the field name change, if you are referencing this field externally (e.g. via a custom plugin).

Upgrading from Bamboo 1.2.x or earlier

In addition to the above, please read the Bamboo 2.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 2.0 Beta Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

This page contains a live summary of all Bamboo release notes for the Bamboo 2.0 Beta. Click through to each of the individual release notes to view the complete list of issues and fixes associated with each release.

Please read the release notes up to and including the version that you are interested in. You may also wish to read the Bamboo 2.0 Beta upgrade guides.

Bamboo 2.0 Beta Release Notes

- Bamboo 2.0 Beta 1
- Bamboo 2.0 Beta 2
- Bamboo 2.0 Beta 3
- Bamboo 2.0 Beta 4
- Bamboo 2.0 Beta 5
- Bamboo 2.0 Beta 6
- Bamboo 2.0 Beta 8
- Bamboo 2.0 Beta 9

Bamboo 2.0 Beta 1

Bamboo 2.0 will be launched in early 2008 and will introduce a number of new features, including the ability to run distributed builds, flexible build agent management and memory usage improvements.

Because Bamboo 2.0 will introduce major architectural changes, the Bamboo 2.0 Beta program is being provided to enable you to preview the upcoming features and perform preliminary testing.

Please note that this release is a beta and should not be used on production systems.

Upgrading to Bamboo 2.0 Beta

Bamboo 2.0 Beta can be downloaded from the Bamboo Download Centre. Before upgrading, please refer to the Bamboo 2.0 Beta Upgrade Guide. You must upgrade to Bamboo 1.2.x before upgrading to 2.0.

What’s New in Bamboo 2.0?

1. Distributed Builds — This release introduces the ability to run distributed builds. You will find this particularly useful if you need to run your builds in different geographic locations, or on different platforms. Simply install the new Bamboo Agent on your additional build servers, and your main Bamboo 2.0 server will be able to manage them. We have also provided a number of plugin points in case you need to control your distributed builds programmatically.

2. Flexible Build Agent Management — Bamboo 2.0 also gives you much more flexibility in managing your builds. Build queues are no longer required, with the introduction of requirements and capabilities. You can direct builds to be run on a particular agent, by specifying build
plan requirements to match the builder, JDK and custom capabilities that you have set up for the agent. Read more about agents and capabilities here.

3 **Memory Usage Improvements** — The underlying engine for Bamboo has been revamped to decrease memory usage. You will notice a distinct improvement in the performance of your builds, especially if you have very large logs.

Read more about Bamboo 2.0 Beta 1...

**Bamboo 2.0 Beta 2**

This point release includes more than 10 minor fixes and improvements. Bamboo 2.0 Beta 2 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 2 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Read more about Bamboo 2.0 Beta 2...

**Bamboo 2.0 Beta 3**

Bamboo 2.0 Beta 3 introduces a number of new features, including Perforce support and the ability to connect to Oracle and MS SQL as external databases. A number of significant fixes have been included as well, improving the overall quality and experience of the beta.

<table>
<thead>
<tr>
<th>Upgrading to Bamboo 2.0 Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo 2.0 Beta can be downloaded from the Bamboo Download Centre. Before upgrading, please read the Bamboo 2.0 Beta 3 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.</td>
</tr>
</tbody>
</table>

**What's New in Bamboo 2.0 Beta 3?**

1 **Perforce Support** — Bamboo brings back out of the box support for Perforce source repositories. A number of changes have been made to Perforce configuration to ensure that it works correctly with distributed builds. Read more about configuring Perforce.

2 **Oracle and MS SQL Server Support** — By popular request, Bamboo's supported databases now include Oracle and MS SQL Server. Read more about connecting Bamboo to Oracle and MS SQL Server.

3 **Major Bug Fixes** — Bamboo now works with PostgreSQL and MySQL correctly. Please see the relevant Postgresql and MySQL JIRA issues for details about the fixes.

Read more about Bamboo 2.0 Beta 3...

**Bamboo 2.0 Beta 4**

This point release includes more than 15 minor fixes and improvements. Bamboo 2.0 Beta 4 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 4 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

**Major Bug Fixes**

- Previously, Bamboo determined which agents could build a plan when the plan was queued, but would then incorrectly ignore any subsequent changes (including disabling the agent). Bamboo will now correctly update where plans can be built, even if changes are made after the plan has been queued.

**For the Developers**
The new 'RepositoryEventAware' interface allows you to implement custom actions before and/or after retrieving source code from your repository. Read more about extending the standard repository functionality.

Read more about Bamboo 2.0 Beta 4...

**Bamboo 2.0 Beta 5**

This point release includes more than 10 minor fixes and improvements. Bamboo 2.0 Beta 5 can be downloaded [here](#).

* Before upgrading, please read the Bamboo 2.0 Beta 5 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

**JDK and Builders page now included in Bamboo 2.0 Beta**

- The JDK and Builder pages which were temporarily removed from Bamboo for the 2.0 Beta have now been restored.

Read more about Bamboo 2.0 Beta 5...

**Bamboo 2.0 Beta 6**

This point release includes more than 5 minor fixes and improvements. Bamboo 2.0 Beta 6 can be downloaded [here](#).

* Before upgrading, please read the Bamboo 2.0 Beta 6 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

**Secured Remote Agents are now Supported**

- Remote agents can now be secured with the appropriate SSL configuration. Read more about [Securing your Remote Agents](#).

- We strongly recommend that you do not enable remote agent installation on any Bamboo instance accessible from a public or untrusted network without securing your remote agents. If you choose to enable your remote agents without securing them, please read [this security advisory](#) to understand the security implications.

**Changes to Client Workspace Configuration for Perforce**

- If you use Perforce, you can now choose whether your want Bamboo to manage your client workspace (i.e. set the client root) or manage it yourself. Read more about [Perforce configuration](#).

**'Shared Local Capabilities' are now called 'Local Server Capabilities'**

- All references to 'Shared Local Capabilities' (or equivalent terminology) have been changed to 'Local Server Capabilities' in the Bamboo user interface.

Read more about Bamboo 2.0 Beta 6...

**Bamboo 2.0 Beta 8**

This point release includes more than 20 minor fixes and improvements. Bamboo 2.0 Beta 8 can be downloaded [here](#).

* Before upgrading, please read the Bamboo 2.0 Beta 8 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

**Quiet Period Functionality Supported for Subversion & Perforce**

By popular request, Quiet Period parameters can now be specified for Subversion and Perforce when configuring a source repository for a build plan. You can choose to set how long Bamboo should wait after a commit before triggering a build, and the number of times it retries before initiating a build. Read more about configuring Subversion and Perforce source repositories.
'Force Clean Builds' Supported

Also by popular request, you can now force Bamboo to run 'Clean Builds' in a build plan. That is, the source directory is removed and then checked out from the repository prior to each build. Read more about this function in Specifying a Plan's Source Repository.

Read more about Bamboo 2.0 Beta 8...

Bamboo 2.0 Beta 9

This point release includes more than 10 minor fixes and improvements. Bamboo 2.0 Beta 9 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 9 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Edit and Rename Capabilities

You can now edit capabilities in Bamboo, as listed below:

- If you are editing a Builder capability, you can modify the 'Path' of the builder.
- If you are editing a JDK capability, you can modify the 'Java Home' of the JDK.
- If you are editing a Custom capability, you can modify the 'Value' of the capability.
- If you are editing a Perforce capability, you can modify the 'Perforce Executable' path.

You can also rename a capability. This is reflected in any plans that the capability is specified as a requirement for.

Read more about editing capabilities and renaming capabilities.

View Agents and Plans related to a Capability

A 'View Capability' screen is now available in Bamboo 2.0. This screen lists the agents that have/inherit a particular capability, as well as which plans have the capability specified as a requirement.

Read more about viewing capabilities.

Read more about Bamboo 2.0 Beta 9...

Bamboo 2.0 Beta Upgrade Guide

This page contains a live aggregate of all version-specific upgrade guides for the Bamboo 2.0 Beta.

How to read the Bamboo 2.0 Beta Upgrade Guides

- Read all Bamboo 2.0 Beta Upgrade Guides up to the version that you are upgrading to.
- Read the relevant release notes.
- Read the Bamboo Generic Upgrade Guide for general upgrade instructions.

Bamboo 2.0 Beta Upgrade Guides

- Upgrading to Bamboo 2.0 Beta 1
- Upgrading to Bamboo 2.0 Beta 2
- Upgrading to Bamboo 2.0 Beta 3
- Upgrading to Bamboo 2.0 Beta 4
- Upgrading to Bamboo 2.0 Beta 5
- Upgrading to Bamboo 2.0 Beta 6
- Upgrading to Bamboo 2.0 Beta 8
- Upgrading to Bamboo 2.0 Beta 9

Upgrading to Bamboo 2.0 Beta 1
If you are using a version of Bamboo prior to version 1.2, you will need to upgrade Bamboo to version 1.2 before you can upgrade to the 2.0 Beta. Note that the upgrade process from version 1.0.x is different from the upgrade process from version 1.1.x. Please follow the appropriate instructions below:

**Upgrading from Bamboo 1.1.x**

You will need to:

1. Upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
2. Then upgrade to the desired version of the Bamboo 2.0 Beta, as per the instructions below.

**Upgrading from Bamboo 1.0.x**

You will need to:

1. Upgrade to 1.1.2 first — please see the Bamboo 1.1.2 Upgrade Guide. (This step is necessary as there is an issue with the upgrade process from the 1.0.x series that we're currently looking into.)
2. Then upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
3. Then upgrade to the desired version of the Bamboo 2.0 Beta, as per the instructions below.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide. Additionally, please note the following:

1. **Adding a Broker URL property.**

Bamboo uses a messaging broker to communicate with its remote build agents. To ensure this works properly, a URL must be specified. This URL is where Bamboo will set up its embedded broker. Remote agents will also be provided with this URL on startup.

To specify the broker URL, please add a `bamboo.jms.broker.url` property in your `bamboo.cfg.xml` file, located inside the Bamboo home directory. For example:

```
<property name="bamboo.jms.broker.uri">tcp://HOSTNAME:54663</property>
```

where HOSTNAME is the canonical name of your Bamboo server.

Please note, as remote agents use this URL to communicate to the server, you should take care not to specify localhost as the host name in the broker URL.

If no broker URL is found in `bamboo.cfg.xml`, Bamboo will default the broker URL to `tcp://HOSTNAME:54663` in the `bamboo.cfg.xml` file, as seen in the example above. Bamboo will also append the parameter `wireFormat.maxInactivityDuration=0` by default to any broker URL coming from `bamboo.cfg.xml`.

2. **Changes to Server Configuration**

   **JDK support**

Bamboo 2.0 requires JDK 1.5 (i.e. JDK 1.4 is no longer supported). Please note that this does not affect the actual builds: it is only the Bamboo server itself that must be running JDK 1.5.

   **Database changes**

The release of 2.0 will include some changes to column names in the database as follows:

- In the BUILD_DEFINITION table, the column XML_DATA will be changed to XML_DEFINITION_DATA
- In the BUILDRESULTSUMMARY_CUSTOMDATA table, the column CUSTOM_INFO_DATA will be changed to CUSTOM_INFO_VALUE

These fields have also had types changed to CLOB to increase their maximum lengths.

**Plugins**

If you are using external or custom plugins, please make sure that your plugins compile against Bamboo 2.0 before upgrading.
3. Changes to Build Queues and Build Plans

Bamboo 2.0 introduces the concepts of agents and capabilities. To preserve the functionality of your existing plans, JDKs, Builders and Build Queues, the following will automatically happen during the upgrade:

**Conversion of Build Queues to Agents**

Prior to Bamboo 2.0, you could have multiple build queues. In Bamboo 2.0, there is now only one build queue, but multiple agents (see diagram). As part of the upgrade process,

- Each of your build queues will be converted to a local agent.
- If, prior to the upgrade, the build queue accepted builds from all plans, the agent will be given the following capability (and every plan will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: ALLOW_ANY_BUILDS
- Or if, prior to the upgrade, the build queue only accepted builds from specific plans, the agent will be given the following capability (and the relevant plans will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: <name of old queue>

If you wish to change this after the upgrade, please see Configuring Agents and Capabilities and Specifying a Plan's Capability Requirements.

**Conversion of Builders to Capabilities**

Prior to Bamboo 2.0, your builders (e.g. Maven) were defined globally. In Bamboo 2.0, builders are now defined as agent capabilities and specified as plan requirements. As part of the upgrade process,

- Each of your builders will be converted to a shared local capability (that is, it will apply to every local agent).
- Every plan will continue to have the same builder that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

**Conversion of JDKs to Capabilities**

Prior to Bamboo 2.0, your JDKs (e.g. JDK 1.5) were defined globally. In Bamboo 2.0, JDKs are now defined as agent capabilities and specified as plan requirements. As part of the upgrade process,

- Each of your JDKs will be converted to shared local capabilities (that is, it will apply to every local agent).
- Upon conversion, the labels of each of your JDKs will upgraded to the Bamboo 2.0 JDK label format, (i.e. 'JDK 9.9.9_99').
- Upon conversion, two more generic versions of the labels will be created for each JDK, (i.e. 'JDK 9.9' and 'JDK').
- Every plan will have its requirements upgraded, to keep the association with the same JDK that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

**Upgrading to Bamboo 2.0 Beta 2**

⚠️ It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 1 to 2.0 Beta 2.

**Upgrading to Bamboo 2.0 Beta 3**

⚠️ It is strongly recommended that you **back up your xml-data directory** before proceeding. You are also strongly recommended to **back up your database** due to schema changes in this release. For full instructions please follow the Bamboo Generic Upgrade Guide. Additionally, please note the following:

1. **Changes to Repositories**

Bamboo 2.0 introduces the concepts of agents and capabilities. To preserve the functionality of your existing Repositories, the following will...
automatically happen during the upgrade:

**Conversion of Perforce P4 Client Application Location to a Capability**

With the introduction of remote agents in Bamboo 2.0, the location of the Perforce P4 client application now needs to be specified as a capability. To create build plans using Perforce as repository, a shared local capability must be created for the P4 client application location. In addition, agent-specific remote capabilities must be created for each remote agent using Perforce.

As part of the upgrade process,

- A shared local Perforce capability will be created for the Perforce P4 client application location. The upgrade task reads this information from the system's environment variables. If the Perforce P4 client application location has not been specified as an environment variable, the shared local capability will need to be set up manually.

The upgrade task will not create agent-specific Perforce capabilities for any remote agents. These capabilities will need to be set up manually.

Please see Configuring a new Perforce Capability for further details on creating Perforce capabilities.

**Upgrading to Bamboo 2.0 Beta 4**

It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

**No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 3 to 2.0 Beta 4.**

**Upgrading to Bamboo 2.0 Beta 5**

It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

**No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 4 to 2.0 Beta 5.**

**Upgrading to Bamboo 2.0 Beta 6**

It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

1. **Changes to Perforce Workspace Management**

If you use Perforce, you can now choose whether you want Bamboo to manage your workspace or whether you want to manage it yourself. Prior to this release, Bamboo would automatically manage your workspace (i.e. changed the client root). Hence, if you want to manage your workspace in this release, you will need to reset your client roots.

**Upgrading to Bamboo 2.0 Beta 8**

It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

**No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 6 to 2.0 Beta 8.**

**Upgrading to Bamboo 2.0 Beta 9**

It is strongly recommended that you **back up your xml-data directory** before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

**No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 8 to 2.0 Beta 9.**

**Bamboo 2.0 Beta 9 Release Notes**

Bamboo 2.0 Beta 9 can be downloaded [here](#).

---

3 April 2008

Atlassian is proud to announce the release of Bamboo 2.0 Beta 9. This point release includes more than 10 minor fixes and improvements. Bamboo 2.0 Beta 9 can be downloaded [here](#).
Before upgrading, please read the Bamboo 2.0 Beta 9 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Edit and Rename Capabilities

You can now edit capabilities in Bamboo, as listed below:

- If you are editing a Builder capability, you can modify the ‘Path’ of the builder.
- If you are editing a JDK capability, you can modify the ‘Java Home’ of the JDK.
- If you are editing a Custom capability, you can modify the ‘Value’ of the capability.
- If you are editing a Perforce capability, you can modify the ‘Perforce Executable’ path.

You can also rename a capability. This is reflected in any plans that the capability is specified as a requirement for.

Read more about editing capabilities and renaming capabilities.

View Agents and Plans related to a Capability

A ‘View Capability’ screen is now available in Bamboo 2.0. This screen lists the agents that have/inherit a particular capability, as well as which plans have the capability specified as a requirement.

Read more about viewing capabilities.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 9 release. Please refer to each of the previous beta release notes to review the complete list of issues and fixes for each beta release.

- Hibernate Errors in logs - this is a known issue, caused due to our pre-hibernate upgrade tasks to prepare Bamboo database for Oracle and MS SQL Server compatibility - For further details, see this Knowledge base article.

Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

<table>
<thead>
<tr>
<th>JIRA Issues (15 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-2407</td>
</tr>
<tr>
<td>BAM-2359</td>
</tr>
</tbody>
</table>
Bamboo 2.0 Beta 9 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 9

These instructions outline how to upgrade Bamboo from version 2.0 Beta 8 to 2.0 Beta 9. If you are upgrading from a version prior to 2.0 Beta 8, please also refer to the aggregated upgrade guides for details on the previous beta releases.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 8 to 2.0 Beta 9.

Bamboo 2.0 Beta 8 Release Notes

27 March 2008
Atlassian is proud to announce the release of Bamboo 2.0 Beta 8. This point release includes more than 20 minor fixes and improvements. Bamboo 2.0 Beta 8 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 8 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Quiet Period Functionality Supported for Subversion & Perforce

By popular request, Quiet Period parameters can now be specified for Subversion and Perforce when configuring a source repository for a build plan. You can choose to set how long Bamboo should wait after a commit before triggering a build, and the number of times it retries before initiating a build. Read more about configuring Subversion and Perforce source repositories.
'Force Clean Builds' Supported

Also by popular request, you can now force Bamboo to run 'Clean Builds' in a build plan. That is, the source directory is removed and then checked out from the repository prior to each build. Read more about this function in Specifying a Plan's Source Repository.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 8 release. Please refer to each of the previous beta release notes to review the complete list of issues and fixes for each beta release.

- Hibernate Errors in logs - this is a known issue, caused due to our pre-hibernate upgrade tasks to prepare Bamboo database for Oracle and MS SQL Server compatibility - For further details, see this Knowledge base article.

Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

JIRA Issues (21 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2391</td>
<td>SVN externals now prohibitively slow</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2384</td>
<td>Test setting up database with JNDI</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2382</td>
<td>Perforce Web RepositoryUrls are not implemented in 2.0</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2375</td>
<td>Artifacts copy problems with local/remote agents</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2372</td>
<td>Repository web URL still create links even though none is specified</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2362</td>
<td>Can't edit existing perforce repository plan to a new P4 workspace</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Scott Marshall</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2358</td>
<td>Can't Find JUnit Results</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Sam Berlin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2309</td>
<td>NullPointerExecution in AgentHeartBeat on startup</td>
<td>Edwin Wong [Atlassian]</td>
<td>David O'Flynn [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2283</td>
<td>Upgrade task to change &quot;Queue&quot; to &quot;Agent&quot; in the 1.2.4 queue names</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2270</td>
<td>Link incorrectly escaped in Builder JDK error Message</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Edwin Wong [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2257</td>
<td>Variable substitution doesn't work in Bamboo 2.0.</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2202</td>
<td>Agent shutdown if the server is shutdown</td>
<td>Edwin Wong [Atlassian]</td>
<td>Benjamin LERMAN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2152</td>
<td>Indexer is not thread safe</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Edwin Wong [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2147</td>
<td>Reshuffle the admin menu to have Plan &amp; Build Resources subheading</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2095</td>
<td>Detected JDKs may not be persisted</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Adrian Hempel [Atlassian]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2007</td>
<td>Bamboo doesn't update available builders after each restart.</td>
<td>Mark Chaimungkalanont</td>
<td>Ajay Sridhar</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 2.0 Beta 8 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 8

These instructions outline how to upgrade Bamboo from version 2.0 Beta 6 to 2.0 Beta 8 (Please note that there is no 2.0 Beta 7). If you are upgrading from a version prior to 2.0 Beta 6, please also refer to the aggregated upgrade guides for details on the previous beta releases.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 6 to 2.0 Beta 8.

Bamboo 2.0 Beta 6 Release Notes

Bamboo 4.0 has been released. Read the full Bamboo 4.0 Release Notes and Upgrade Guide.

Don't have Bamboo 4.0? Take a look at the features of Bamboo's latest major version and try it out!

18 March 2008
Atlassian is proud to announce the release of Bamboo 2.0 Beta 6. This point release includes more than 5 minor fixes and improvements. Bamboo 2.0 Beta 6 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 6 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Secured Remote Agents are now Supported

• Remote agents can now be secured with the appropriate SSL configuration. Read more about Securing your Remote Agents.

We strongly recommend that you do not enable remote agent installation on any Bamboo instance accessible from a public or untrusted network without securing your remote agents. If you choose to enable your remote agents without securing them, please read this security advisory to understand the security implications.

Changes to Client Workspace Configuration for Perforce

• If you use Perforce, you can now choose whether you want Bamboo to manage your client workspace (i.e. set the client root) or manage it yourself. Read more about Perforce configuration.

'Shared Local Capabilities' are now called 'Local Server Capabilities'

• All references to 'Shared Local Capabilities' (or equivalent terminology) have been changed to 'Local Server Capabilities' in the Bamboo user interface.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 6 release. Please refer to each of the previous beta release notes to review the complete list of issues and fixes for each beta release.

• Secured remote agents are not supported in the Bamboo 3.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see...
Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

<table>
<thead>
<tr>
<th>JIRA Issues (11 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>BAM-2366</td>
</tr>
</tbody>
</table>

Bamboo 2.0 Beta 6 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 6

These instructions outline how to upgrade Bamboo from version 2.0 Beta 5 to 2.0 Beta 6. If you are upgrading from a version prior to 2.0 Beta 5, please also refer to the aggregated upgrade guides for details on the previous beta releases.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

1. Changes to Perforce Workspace Management
If you use Perforce, you can now choose whether you want Bamboo to manage your workspace or whether you want to manage it yourself. Prior to this release, Bamboo would automatically manage your workspace (i.e. changed the client root). Hence, if you want to manage your workspace in this release, you will need to reset your client roots.

Bamboo 2.0 Beta 5 Release Notes

Bamboo 2.0 Beta 5 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 5 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

JDK and Builders page now included in Bamboo 2.0 Beta

- The JDK and Builder pages which were temporarily removed from Bamboo for the 2.0 Beta have now been restored.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 5 release. Please refer to each of the previous beta release notes to review the complete list of issues and fixes for each beta release.

- JDK and Builders pages have not been included in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).

  FIXED!
- Hibernate Errors in logs - this is a known issue, caused due to our pre-hibernate upgrade tasks to prepare Bamboo database for Oracle and MS SQL Server compatibility - For further details, see this Knowledge base article.
- Secured remote agents are not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details.

Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

JIRA Issues (15 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-2346</td>
<td>Failure to clear working directory on repository change if the agent has never built that plan before.</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>✅</td>
<td>☑️</td>
<td>Fixed</td>
<td>Mar 06, 2008</td>
<td>Apr 01 2008</td>
</tr>
<tr>
<td></td>
<td>BAM-2326</td>
<td>Figure out why functional tests stop logging</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>✅</td>
<td>☑️</td>
<td>Fixed</td>
<td>Mar 04, 2008</td>
<td>Apr 01 2008</td>
</tr>
<tr>
<td></td>
<td>BAM-2325</td>
<td>ScheduleBackupConfiguration exception if backup is configured</td>
<td>Mark Chaimungkalanon [Atlassian]</td>
<td>Aristedes Maniatis</td>
<td>✅</td>
<td>☑️</td>
<td>Fixed</td>
<td>Mar 04, 2008</td>
<td>Apr 01 2008</td>
</tr>
<tr>
<td></td>
<td>BAM-2322</td>
<td>New Perforce Library puts password into command line which is not all that secure</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>✅</td>
<td>☑️</td>
<td>Fixed</td>
<td>Mar 03, 2008</td>
<td>Apr 01 2008</td>
</tr>
<tr>
<td></td>
<td>BAM-2312</td>
<td>Cannot Setup Perforce Repository: &quot;This user is not available on this port (Perforce server)&quot;</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Steven Salter</td>
<td>✅</td>
<td>☑️</td>
<td>Fixed</td>
<td>Feb 28, 2008</td>
<td>Apr 01 2008</td>
</tr>
</tbody>
</table>

Build status doesn't update | Mark |
Bamboo 2.0 Beta 5 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 5

These instructions outline how to upgrade Bamboo from version 2.0 Beta 4 to 2.0 Beta 5. If you are upgrading from a version prior to 2.0 Beta 4, please also refer to the aggregated upgrade guides for details on the previous beta releases.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 4 to 2.0 Beta 5.

Bamboo 2.0 Beta 4 Release Notes

Bamboo 2.0 Beta 4 has been released. Read the full Bamboo 2.0 Beta 4 Release Notes and Upgrade Guide. Don't have Bamboo 2.0? Take a look at the features of Bamboo’s latest major version and try it out!

4 March 2008
Atlassian is proud to announce the release of Bamboo 2.0 Beta 4. This point release includes more than 15 minor fixes and improvements. Bamboo 2.0 Beta 4 can be downloaded here.

Before upgrading, please read the Bamboo 2.0 Beta 4 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

Major Bug Fixes

• Previously, Bamboo determined which agents could build a plan when the plan was queued, but would then incorrectly ignore any subsequent changes (including disabling the agent). Bamboo will now correctly update where plans can be built, even if changes are made after the plan has been queued.

For the Developers

• The new ‘RepositoryEventAware’ interface allows you to implement custom actions before and/or after retrieving source code from your repository. Read more about extending the standard repository functionality.

Known Issues
The following issues are applicable at the time of the Bamboo 2.0 Beta 4 release. Please refer to each of the previous beta release notes to review the complete list of issues and fixes for each beta release.

- Please note that builds are currently allocated to agents during queuing time, not execution time. This may occasionally mean that a build is executed by an agent that you have disabled, which is slightly different from the functionality described in Monitoring Agent Status.
- FIXED!
- Hibernate Errors in logs - this is a known issue, caused due to our pre-hibernate upgrade tasks to prepare Bamboo database for Oracle and MS SQL Server compatibility - For further details, see this Knowledge base article.
- Secured remote agents are not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details.
- JDK and Builders pages have not been included in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).

Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

<table>
<thead>
<tr>
<th>JIRA Issues (16 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-2299</td>
</tr>
<tr>
<td>BAM-2294</td>
</tr>
<tr>
<td>BAM-2248</td>
</tr>
<tr>
<td>BAM-2193</td>
</tr>
</tbody>
</table>

Queued builds with no eligible builder are not built even if Edwin Wong | Adrian Hempel |  |  | Jan 16, 2008 | Apr 01, 2008 |
Bamboo 2.0 Beta 4 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 4

These instructions outline how to upgrade Bamboo from version 2.0 Beta 3 to 2.0 Beta 4. If you are upgrading from a version prior to 2.0 Beta 3, please also refer to the aggregated upgrade guides for details on the previous beta releases.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 3 to 2.0 Beta 4.

Bamboo 2.0 Beta 3 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

27 February, 2008

Atlassian Software Systems presents Bamboo 2.0 Beta 3

Bamboo 2.0 Beta 3 introduces a number of new features, including Perforce support and the ability to connect to Oracle and MS SQL as external databases. A number of significant fixes have been included as well, improving the overall quality and experience of the beta.

<table>
<thead>
<tr>
<th>Upgrading to Bamboo 2.0 Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo 2.0 Beta can be downloaded from the Bamboo Download Centre. Before upgrading, please read the Bamboo 2.0 Beta 3 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta 2, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.</td>
</tr>
</tbody>
</table>

What's New in Bamboo 2.0 Beta 3?

1. **Perforce Support** — Bamboo brings back out of the box support for Perforce source repositories. A number of changes have been made to Perforce configuration to ensure that it works correctly with distributed builds. Read more about configuring Perforce.

2. **Oracle and MS SQL Server Support** — By popular request, Bamboo's supported databases now include Oracle and MS SQL Server. Read more about connecting Bamboo to Oracle and MS SQL Server.

3. **Major Bug Fixes** — Bamboo now works with PostgreSQL and MySQL correctly. Please see the relevant Postgresql and MySQL JIRA issues for details about the fixes.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 3 release. Please refer to each of the previous beta release notes to
review the complete list of issues and fixes for each beta release.

- Perforce is not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release); FIXED!
- Bamboo currently does not work with MySQL. See BAM-2260 for further details. FIXED!
- Plans currently cannot be edited, if Bamboo is integrated with a Postgresql database. See BAM-2208 for further details. FIXED!
- Hibernate Errors in logs - this is a known issue, caused due to our pre-Hibernate upgrade tasks to prepare Bamboo database for Oracle and MS SQL Server compatibility. For further details, see this Knowledge base article.
- Secured remote agents are not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details.
- JDK and Builders pages have not been included in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).
- Please note that builds are currently allocated to agents during queuing time, not execution time. This may occasionally mean that a build is executed by an agent that you have disabled, which is slightly different from the functionality described in Monitoring Agent Status. For example:
  1. Agent A is enabled and is currently executing a build for Plan X.
  2. Plan Y submits a build to the queue, and the queue assigns the build to Agent A.
  3. You disable Agent A.
  4. Agent A completes Plan X's build.
  5. Although agent A is disabled, it will still run Plan Y's build, because it was determined as executable when plan Y was queued.

Updates and Fixes in this Release

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

<table>
<thead>
<tr>
<th>JIRA Issues (18 issues)</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2290</td>
<td>Incorrect instructions for running remote agent JAR</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2284</td>
<td>Upgrading from 1.2.4 will update the JDK keys to the wrong value</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2282</td>
<td>Ability to run automatic detection of environment variables as JDKs &amp; Builders</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2276</td>
<td>Better logging when remote agents disabled</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2264</td>
<td>Extend Bamboo REST API to provide access to details of test cases</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2262</td>
<td>Improve Bamboo REST API error response to provide information that can be presented to an end user</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2260</td>
<td>Mysql integratio with Bamboo 2.0 Beta release 1 doesn't work</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2245</td>
<td>Forgotten password reports wrong error when user doesn't exist</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2229</td>
<td>More agent meta data</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2112</td>
<td>Crowd with delegated LDAP auth - update documentation for Bamboo-Crowd integration</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1807</td>
<td>subversion/source control queue</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1504</td>
<td>Ability to import data during setup without restarting</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1466</td>
<td>Bamboo should support MS SQL Database</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1465</td>
<td>Bamboo should support Oracle Database</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1365</td>
<td>Assigning of build to queues is non-optimal</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1110</td>
<td>Ability to add plan to build queue as you are creating said plan</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1082</td>
<td>Perforce source code directory (client root) is currently cached</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-924</td>
<td>Perforce client error forces full checkout/build</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Bamboo 2.0 Beta 3 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 3

These instructions outline how to upgrade Bamboo from version 2.0 Beta 2 to 2.0 Beta 3. If you are upgrading from a version prior to 2.0 Beta 2, please refer to the aggregated upgrade guides for details on the previous beta releases.
It is strongly recommended that you **back up your xml-data directory** before proceeding. You are also strongly recommended to **back up your database** due to schema changes in this release. For full instructions please follow the **Bamboo Generic Upgrade Guide**. Additionally, please note the following:

1. **Changes to Repositories**

Bamboo 2.0 introduces the concepts of **agents and capabilities**. To preserve the functionality of your existing Repositories, the following will automatically happen during the upgrade:

**Conversion of Perforce P4 Client Application Location to a Capability**

With the introduction of remote agents in Bamboo 2.0, the location of the Perforce P4 client application now needs to be specified as a capability. To create build plans using Perforce as repository, a shared local capability must be created for the P4 client application location. In addition, agent-specific remote capabilities must be created for each remote agent using Perforce.

As part of the upgrade process,

- A shared local Perforce capability will be created for the Perforce P4 client application location. The upgrade task reads this information from the system's environment variables. If the Perforce P4 client application location has not been specified as an environment variable, the shared local capability will need to be set up manually.

The upgrade task will not create agent-specific Perforce capabilities for any remote agents. These capabilities will need to be set up manually.

Please see **Configuring a new Perforce Capability** for further details on creating Perforce capabilities.

### Bamboo 2.0 Beta 2 Release Notes

**Bamboo 4.4 has been released. Read the full Bamboo 4.4 Release Notes and Upgrade Guide. Don’t have Bamboo 4.4? Take a look at the features of Bamboo's latest major version and try it out!**

19 February 2008

Atlassian is proud to announce the release of Bamboo 2.0 Beta 2. This point release includes more than 10 minor fixes and improvements. Bamboo 2.0 Beta 2 can be downloaded [here](#). Before upgrading, please read the Bamboo 2.0 Beta 2 Upgrade Guide. If you are upgrading from a version of Bamboo prior to the Bamboo 2.0 Beta, please read all of the Bamboo 2.0 Beta Release Notes and Bamboo 2.0 Beta Upgrade Guides before upgrading.

#### Known Issues

The following issues are applicable at the time of the **Bamboo 2.0 Beta 2** release. Please refer to the aggregated **Bamboo 2.0 Beta Release Notes** to review the complete list of issues and fixes for each beta release.

- Perforce is not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).
- Bamboo currently does not work with MySQL. See **BAM-2260** for further details.
- Plans currently cannot be edited, if Bamboo is integrated with a PostgreSQL database. See **BAM-2280** for further details.
- Secured remote agents are not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details.
- JDK and Builders pages have not been included in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).
- Please note that builds are currently allocated to agents during queuing time, not execution time. This may occasionally mean that a build is executed by an agent that you have disabled, which is slightly different from the functionality described in **Monitoring Agent Status**. For example:
  1. Agent A is enabled and is currently executing a build for Plan X.
  2. Plan Y submits a build to the queue, and the queue assigns the build to Agent A.
  3. You disable Agent A.
  4. Agent A completes Plan X's build.
  5. Although agent A is disabled, it will still run Plan Y's build, because it was determined as executable when plan Y was queued.

#### Updates and issues fixed

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at [jira.atlassian.com](https://jira.atlassian.com).

### JIRA Issues (13 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2236</td>
<td>Ensure that java.io.tmp exists when server / agent starts up</td>
<td>Unassigned</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 12, 2008</td>
<td>Sep 01, 2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2231</td>
<td>Agents matrix now display which requirements are missing</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 12, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2222</td>
<td>Clicking on the &quot;stop build&quot; icon (red square) next to a build in the &quot;Current Activity&quot; tab throws exception</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Dave Loeng [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 11, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2201</td>
<td>Bamboo use a unexistant SQL function withpostgresql</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Benjamin LERMAN</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 08, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bamboo 2.0 Beta 2 Upgrade Guide**

**Upgrading to Bamboo 2.0 Beta 2**

These instructions outline how to upgrade Bamboo from version 2.0 Beta 1 to 2.0 Beta 2. If you are upgrading from a version prior to 2.0 Beta 1, please also refer to the aggregated upgrade guides for details on the previous beta releases.

⚠️ It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0 Beta 1 to 2.0 Beta 2.

**Bamboo 2.0 Beta 1 Release Notes**

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!
Atlassian Software Systems presents Bamboo 2.0 Beta

Bamboo 2.0 will be launched in early 2008 and will introduce a number of new features, including the ability to run distributed builds, flexible build agent management and memory usage improvements.

Because Bamboo 2.0 will introduce major architectural changes, the Bamboo 2.0 Beta program is being provided to enable you to preview the upcoming features and perform preliminary testing.

Please note that this release is a beta and should not be used on production systems.

<table>
<thead>
<tr>
<th>Bamboo 2.0 Beta can be downloaded from the Bamboo Download Centre. Before upgrading, please refer to the Bamboo 2.0 Beta Upgrade Guide. You must upgrade to Bamboo 1.2.x before upgrading to 2.0.</th>
</tr>
</thead>
</table>

What's New in Bamboo 2.0?

1. **Distributed Builds** — This release introduces the ability to run distributed builds. You will find this particularly useful if you need to run your builds in different geographic locations, or on different platforms. Simply install the new Bamboo Agent on your additional build servers, and your main Bamboo 2.0 server will be able to manage them. We have also provided a number of plugin points in case you need to control your distributed builds programmatically.

2. **Flexible Build Agent Management** — Bamboo 2.0 also gives you much more flexibility in managing your builds. Build queues are no longer required, with the introduction of requirements and capabilities. You can direct builds to be run on a particular agent, by specifying build plan requirements to match the builder, JDK and custom capabilities that you have set up for the agent. Read more about agents and capabilities here.

3. **Memory Usage Improvements** — The underlying engine for Bamboo has been revamped to decrease memory usage. You will notice a distinct improvement in the performance of your builds, especially if you have very large logs.

Security Considerations

⚠ Important security information for the Bamboo 2.0 Beta has been published. Please refer to the security advisory for details.

Known Issues

The following issues are applicable at the time of the Bamboo 2.0 Beta 1 release. Please refer to the aggregated Bamboo 2.0 Beta Release Notes to review the complete list of issues and fixes for each beta release.

- Perforce is not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).
- Bamboo currently does not work with MySQL. See BAM-2260 for further details.
- Plans currently cannot be edited, if Bamboo is integrated with a PostgreSQL database. See BAM-2208 for further details.
- Secured remote agents are not supported in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details.
- JDK and Builders pages have not been included in the Bamboo 2.0 Beta (but will be supported in the official Bamboo 2.0 release).
- Please note that builds are currently allocated to agents during queuing time, not execution time. This may occasionally mean that a build is executed by an agent that you have disabled, which is slightly different from the functionality described in Monitoring Agent Status. For example:
  1. Agent A is enabled and is currently executing a build for Plan X.
  2. Plan Y submits a build to the queue, and the queue assigns the build to Agent A.
  3. You disable Agent A.
  4. Agent A completes Plan X's build.
  5. Although agent A is disabled, it will still run Plan Y's build, because it was determined as executable when plan Y was queued.

Updates and Fixes in this Release

Please help us with the final 2.0 release by reporting any bugs and issues you find, in the Bamboo project at jira.atlassian.com.

<p>| JIRA Issues (189 issues) |
| --- | --- | --- |
| Key | Summary | Priority | Status |
| | | | |</p>
<table>
<thead>
<tr>
<th>JIRA Key</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-3010</td>
<td>Remove &quot;build auto-execution&quot; feature</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2609</td>
<td>Show who canceled a build in the log</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2462</td>
<td>Add link to the Status Summary Screen</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2460</td>
<td>Caching error may cause wrong build result to be cached</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2448</td>
<td>File Added In Attic Changelog Shown For CVS Plans on MAIN</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2447</td>
<td>CVS Plans Don't Work For Branches</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2435</td>
<td>When deleting a requirement within th error screen, creates null pointer</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2431</td>
<td>Unable to save buildresultsummary if a code commit doesn't have comment.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2430</td>
<td>Dashboard quirk while removing plan from favourites list on the My Bamboo tab of dashboard.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2429</td>
<td>500 on /admin/configurePipeline/default.action</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2425</td>
<td>Errors in IE7 while viewing the Edit Build configuration page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2422</td>
<td>Error while importing Bamboo buildresults from bdac due to missing authors from commits</td>
<td>Closed</td>
</tr>
<tr>
<td>BAM-2407</td>
<td>Remote Agent running on Sun JDK 1.6 exits after build is queued for it</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2405</td>
<td>Add extension builder interfaces</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2404</td>
<td>Clicking on the previous button of the Plan creation wizard doesn't work on the Requirements page.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2401</td>
<td>Remote agent restart can leave builds in a building state while the agent reports it is idle</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2391</td>
<td>SVN exteranls now prohibitively slow</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2390</td>
<td>Allow capabilities to be used in variable replacements</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2388</td>
<td>Javascript messagebox shown when ajax connection to bamboo fails is very annoying</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2383</td>
<td>NPE trying to access <a href="https://bamboo.developer.atlassian.com/browse/FISH-STABLE/test">https://bamboo.developer.atlassian.com/browse/FISH-STABLE/test</a></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2382</td>
<td>Perforce Web Repository Urls are not implemented in 2.0</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2381</td>
<td>Failing since build for a test reports the wrong result</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2372</td>
<td>Repository web URL still create links even though none is specified</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2368</td>
<td>Queue Reconstruction intermittently fails when queue size is large</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2366</td>
<td>Perforce Validation does not work if there is no global permissions set</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2360</td>
<td>Manual Build No Longer Shows Changes</td>
<td>Resolved</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>BAM-2359</td>
<td>Plan Not Building</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2358</td>
<td>Can't Find JUnit Results</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2355</td>
<td>Bamboo 2.0 Upgrade Task 608 Fails</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2353</td>
<td>CVS Repository last update time should be from the files updated</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2348</td>
<td>Build result navigator misaligned</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2346</td>
<td>Failure to clear working directory on repository change if the agent has never built that plan before.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2330</td>
<td>Can't add new notification rules for plans.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2329</td>
<td>Bamboo sends IM messages with an empty &quot;to&quot; field.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2327</td>
<td>Dashboard Refresh problems</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2325</td>
<td>ScheduleBackupConfiguration exception if backup is configured</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2322</td>
<td>New Perforce Library puts password into command line which is not all that secure</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2321</td>
<td>Perforce failing to connect on agent</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2317</td>
<td>Quirks with logging on Bamboo agent.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2314</td>
<td>Allow further hooks for the repository</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2312</td>
<td>Cannot Setup Perforce Repository: &quot;This user is not available on this port (Perforce server)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2309</td>
<td>NullPointerException in AgentHeartBeat on startup</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2308</td>
<td>Improve memory footprint of Bamboo</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2299</td>
<td>Provide the ability to add a comment to a build result via Bamboo's REST API</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2294</td>
<td>Setup wizard fails at first step if IP address cannot be determined</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2293</td>
<td>On repository change clearing build directory fails</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2291</td>
<td>Can't add custom build requirement</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2290</td>
<td>Incorrect instructions for running remote agent JAR</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2288</td>
<td>Make details of the commits contained in a build visible via the Bamboo REST API</td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Resolution</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>BAM-2284</td>
<td>Upgrading from 1.2.4 will update the JDK keys to the wrong value</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2283</td>
<td>Upgrade task to change &quot;Queue&quot; to &quot;Agent&quot; in the 1.2.4 queue names</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2282</td>
<td>Ability to run automatic detection of environment variables as JDKs &amp; Builders</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2279</td>
<td>Improve link to download WAR edition of Bamboo</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2278</td>
<td>Build status doesn't update correctly on dashboard until a refresh is done</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2276</td>
<td>Better logging when remote agents disabled</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2270</td>
<td>Link incorrectly escaped in Builder JDK error Message</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2267</td>
<td>Editing capabilities and requirements</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2264</td>
<td>Extend Bamboo REST API to provide access to details of test cases</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2262</td>
<td>Improve Bamboo REST API error response to provide information that can be presented to an end user</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2260</td>
<td>Mysql integratio with Bamboo 2.0 Beta release 1 doesn't work</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2250</td>
<td>Agent bootstrap doesn't handle paths without trailing /</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2248</td>
<td>Unable to delete a build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2246</td>
<td>Improve logging of why a build passes or fails</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2245</td>
<td>Forgotten password reports wrong error when user doesn't exist</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2241</td>
<td>Build Configuration page for script builder, doesn't show the values in the Arguments field in the UI.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2240</td>
<td>Number Format Exception when adding a label to a build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2238</td>
<td>pre-authenticated RSS feed</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2236</td>
<td>Ensure that java.io.tmp exists when server / agent starts up</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2235</td>
<td>Live logs should show full username where possible</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2231</td>
<td>Agents matrix now display which requirements are missing</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2230</td>
<td>Autonomising of agents only goes up to (2)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2229</td>
<td>More agent meta data</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2226</td>
<td>Build expiry fails when excluding labels</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2225</td>
<td>Bamboo sent empty notification e-mail</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2224</td>
<td>Regression: Failing tests no longer displayed in build failure summary</td>
<td>Resolved</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>BAM-2223</td>
<td>Build requirements are duplicated</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2222</td>
<td>Clicking on the &quot;stop build&quot; icon (red square) next to a build in the &quot;Current Activity&quot; tab throws exception</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2208</td>
<td>Cannot edit maven 2 plan</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2207</td>
<td>Unable to start Bamboo successfully if JRE cannot determine IP address</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2206</td>
<td>Alt text appearing instead of icons on All Plans tab</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2205</td>
<td>Cannot add comment to failed build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2202</td>
<td>Agent shutdown if the server is shutdown</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2201</td>
<td>Bamboo use a unexistant SQL function with postgreSQL</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2199</td>
<td>BEAC Upgrade issues</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2197</td>
<td>UpgradeTask610BuilderPlanRequirements fails if we can't parse JDK</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2196</td>
<td>UpgradeTask608BuildLogStorageUpdate will fail a plan doesn't have a results folder</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2195</td>
<td>Clicking on favourite link navigates to malformed page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2194</td>
<td>Clicking edit for an agent, then cancel, should return you to the page you were just on</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2193</td>
<td>Bamboo has problems if you change the SVN Source</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2176</td>
<td>Put in a null check to UpgradeTask608BuildLogStorageUpdate, to ensure the buildresultxml file being parsed is not null!</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2166</td>
<td>Importing Bambood data between different locales produces java.text.ParseException: Unparseable date: due to different date/time formats.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2152</td>
<td>Indexer is not thread safe</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2150</td>
<td>Clicking &quot;Clear error from log&quot; on a plan's page navigates to the Dashboard</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2147</td>
<td>Reshuffle the admin menu to have Plan &amp; Build Resources subheading</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2146</td>
<td>Readd Builder and JDK page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2140</td>
<td>A capability screen that allows bulk editing of capabilities and requirements</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2132</td>
<td>Two attempts to start a manual build of a plan in quick session locks the plan, and does not result in a build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2129</td>
<td>getLastBuildResultSummaries is returning first, rather than last, BuildResultSummaries</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2122</td>
<td>Some exceptions during a build don't cause build to abort</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2118</td>
<td>CustomBuildInfo is not transferred from remote agent to server</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2117</td>
<td>VCSVersionReader returns an illegal null on Remote Agent</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2116</td>
<td>If Remote Agent dies, subsequent attempt to start it fails</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2113</td>
<td>Clover plug-in may use out-of-date Clover data</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2104</td>
<td>Confusing to upgrade an expired Bamboo 1.2 license, to Bamboo 2.0!</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2103</td>
<td>Bamboo 1.2.4 import into Bamboo 2.0, throws exception due to UpgradeTask 612</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2102</td>
<td>Queued builds with no eligible builder are not built even if requirements are edited to make a builder eligible</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2100</td>
<td>Bamboo should indicate queued builds that cannot be built on any active agent</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2096</td>
<td>JDKs autodetected on Bamboo 2.0 agents don't match builds requiring autodetected JDKs imported from Bamboo 1.2.4</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2095</td>
<td>Detected JDKs may not be persisted</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2077</td>
<td>Build Result Not Saved</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2073</td>
<td>Nullpointer exception when trying to save build configuration</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2063</td>
<td>Reimplement include / exclude files</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2057</td>
<td>Installers should configure JMS broker URI</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2032</td>
<td>Perforce configuration throws nullpointer exception</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2029</td>
<td>Bamboo upgrade task fails if the build Results XML file is corrupted</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2017</td>
<td>BuildResultsImpl.cloneAsBuildResults() does not copy the data included in the customBuildData variable</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2013</td>
<td>Implement Perforce ticket authentication in Bamboo.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2007</td>
<td>Bamboo doesn't update available builders after each restart.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1996</td>
<td>Anonymous user is able to download artifacts even if Anonymous mode is disabled both at global level and plan level</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1976</td>
<td>Build artifacts should display the size of the file (if single file)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1968</td>
<td>Make the install4j path a property in the standalone distribution project</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1967</td>
<td>the main distribution POM doesn't use the correct parent POM</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1966</td>
<td>The title of the Configure Groups page is incorrect</td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue ID</td>
<td>Description</td>
<td>Resolution</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>BAM-1964</td>
<td>Use the enforcer plugin to fail fast if a developer uses a jdk below 1.5</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1944</td>
<td>Users should not be able to view plans if Global Anonymous access is disabled.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1932</td>
<td>Fix up files tab</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1897</td>
<td>Change log entries that contain URLs appear fine on screen but the actual HREF is mangled</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1882</td>
<td>Code coverage setup says optional, but is required</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1860</td>
<td>Set a time limit on number of times Bamboo tries to checkout.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1847</td>
<td>Bamboo does not export ServerID</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1821</td>
<td>Bamboo should not die with freemarker error due to undefined values.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1813</td>
<td>The project level RSS feeds are ordered wrong.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1807</td>
<td>subversion/source control queue</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1797</td>
<td>Bamboo runs out of Permgen space, when saving build result - due to massive build log output</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1752</td>
<td>Bamboo does not handle the failure to delete source code very well</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1740</td>
<td>Add file sizes to Artifacts page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1719</td>
<td>create a Confluence Bamboo Plugin (similar to the JIRA Bamboo Plugin)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1709</td>
<td>FileNotFoundException when accessing 'Index Operations' screen in a new empty instance</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1696</td>
<td>Increase the VARCHAR(4000) in 'CUSTOM_INFO_DATA' column of 'BUILDRESULTSUMMARY_CUSTOMDATA' table from 4000</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1688</td>
<td>When updating group membership in crowd changes don't get reflected in Bamboo for a while</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1682</td>
<td>Prevent OutOfMemory errors while viewing the logs page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1658</td>
<td>BuildNumberStamper should be sidegraded to a CustomPreBuildAction</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1652</td>
<td>ChangeLogEntry has invalid myDate field</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1628</td>
<td>NullPointerException when adding many artifacts</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1618</td>
<td>Feedback as an admin message upon catastrophic IO exceptions</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1604</td>
<td>SVN Externals, do not work when pointed to a specific revision number</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1586</td>
<td>Bamboo doesn't free up SVN connections</td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Status</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>BAM-1585</td>
<td>Successful test results are apparently unordered</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1572</td>
<td>Add also the user who triggered the manual build to the Build Results page and Recently Completed Builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1518</td>
<td>Build logs should be separated from the activity logs</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1504</td>
<td>Ability to import data during setup without restarting</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1466</td>
<td>Bamboo should support MS SQL Database</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1465</td>
<td>Bamboo should support Oracle Database</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1416</td>
<td>Allow parallel VCS checkouts</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1401</td>
<td>Allow multiple build plans to be updating from SCM at the same time</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1365</td>
<td>Assigning of build to queues is non-optimal</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1356</td>
<td>Collect build changes for full checkout/clean builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1314</td>
<td>An option to delete local working copy of the repository before building</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1303</td>
<td>Ability to see who executed a manual build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1297</td>
<td>Keep track of what/who triggered a build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1295</td>
<td>Show a list of all successful tests run for a build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1294</td>
<td>Ability to find a specific test on the successful tests screen</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1288</td>
<td>CVS changelogs are not picked up when the repository is on different timezone</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1257</td>
<td>Pause in between builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1251</td>
<td>Ability to pick force synch flag in Perforce</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1186</td>
<td>Bamboo shouldn't loop through all the plans, when building an specific plan</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1178</td>
<td>Commit quiet period</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1110</td>
<td>Ability to add plan to build queue as you are creating said plan</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1082</td>
<td>Perforce source code directory (client root) is currently cached</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1069</td>
<td>Bamboo keeps building and won't stop</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1041</td>
<td>Build Expiry Settings - mark some build as non expirable</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1034</td>
<td>Allow a full checkout to be performed for each change</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1012</td>
<td>Output Shows Directories With Undisplayable Character</td>
<td>Resolved</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>BAM-991</td>
<td>Need a BuildRequired queue</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-990</td>
<td>BuildChangeDetector should only run on a Quartz schedule</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-981</td>
<td>Checkins during a build don't seem to trigger another build after the current one finished</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-926</td>
<td>An option to delay building after checkout detected for SVN</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-924</td>
<td>Perforce client error forces full checkout/build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-894</td>
<td>Multiple remote cross platform builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-760</td>
<td>Errors on dashboard should show which build they were raised in</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-705</td>
<td>Log the user who started the build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-662</td>
<td>Having independant build queues underutilises resources</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-498</td>
<td>Building hangs if one build does not get response from source repository.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-479</td>
<td>Starting builds on multiple servers</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-462</td>
<td>Some errors in XML results parsing not reported to the UI</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-293</td>
<td>Ability to do remote/distributed builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-229</td>
<td>Link dependent builds to the build which launched them</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-195</td>
<td>Create new BuildReason object which can pass along more information for dependency builds.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-21</td>
<td>Investigate making change detector to run asynchronously or in a queue</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

**Bamboo 2.0 Beta 1 Upgrade Guide**

Upgrading to Bamboo 2.0 Beta 1
If you are using a version of Bamboo prior to version 1.2, you will need to upgrade Bamboo to version 1.2 before you can upgrade to the 2.0 Beta. Note that the upgrade process from version 1.0.x is different from the upgrade process from version 1.1.x. Please follow the appropriate instructions below:

**Upgrading from Bamboo 1.1.x**

You will need to:

1. Upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
2. Then upgrade to the desired version of the Bamboo 2.0 Beta, as per the instructions below.

**Upgrading from Bamboo 1.0.x**

You will need to:

1. Upgrade to 1.1.2 first — please see the Bamboo 1.1.2 Upgrade Guide. (This step is necessary as there is an issue with the upgrade process from the 1.0.x series that we're currently looking into.)
2. Then upgrade to Bamboo 1.2 — please see the Bamboo 1.2 Upgrade Guide.
3. Then upgrade to the desired version of the Bamboo 2.0 Beta, as per the instructions below.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide. Additionally, please note the following:

1. **Adding a Broker URL property.**

Bamboo uses a messaging broker to communicate with its remote build agents. To ensure this works properly, a URL must be specified. This URL is where Bamboo will set up its embedded broker. Remote agents will also be provided with this URL on startup.

To specify the broker URL, please add a `bamboo.jms.broker.url` property in your `bamboo.cfg.xml` file, located inside the Bamboo home directory. For example:

```
<property name="bamboo.jms.broker.uri">tcp://HOSTNAME:54663</property>
```

where HOSTNAME is the canonical name of your Bamboo server.

Please note, as remote agents use this URL to communicate to the server, you should take care not to specify localhost as the host name in the broker URL.

If no broker URL is found in `bamboo.cfg.xml`, Bamboo will default the broker URL to `tcp://HOSTNAME:54663` in the `bamboo.cfg.xml` file, as seen in the example above. Bamboo will also append the parameter `wireFormat.maxInactivityDuration=0` by default to any broker URL coming from `bamboo.cfg.xml`.

2. **Changes to Server Configuration**

   **JDK support**

Bamboo 2.0 requires JDK 1.5 (i.e. JDK 1.4 is no longer supported). Please note that this does not affect the actual builds: it is only the Bamboo server itself that must be running JDK 1.5.

   **Database changes**

The release of 2.0 will include some changes to column names in the database as follows:

- In the `BUILD_DEFINITION` table, the column `XML_DATA` will be changed to `XML_DEFINITION_DATA`
- In the `BUILDRESULTSUMMARY_CUSTOMDATA` table, the column `CUSTOM_INFO_DATA` will be changed to `CUSTOM_INFO_VALUE`

These fields have also had types changed to CLOB to increase their maximum lengths.

   **Plugins**

If you are using external or custom plugins, please make sure that your plugins compile against Bamboo 2.0 before upgrading.

```
We've made significant changes to the internals of the application for Bamboo 2.0. If you've installed an external plugin for 1.2.4, it's likely that it will be broken. Please take care when upgrading.
```
3. Changes to Build Queues and Build Plans

Bamboo 2.0 introduces the concepts of agents and capabilities. To preserve the functionality of your existing plans, JDKs, Builders and Build Queues, the following will automatically happen during the upgrade:

Conversion of Build Queues to Agents

Prior to Bamboo 2.0, you could have multiple build queues. In Bamboo 2.0, there is now only one build queue, but multiple agents (see diagram).

As part of the upgrade process,

- Each of your build queues will be converted to a local agent.
- If, prior to the upgrade, the build queue accepted builds from all plans, the agent will be given the following capability (and every plan will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: ALLOW_ANY_BUILDS
- Or if, prior to the upgrade, the build queue only accepted builds from specific plans, the agent will be given the following capability (and the relevant plans will be given an equivalent requirement):
  - Key: bamboo.1.2.queue
  - Value: <name of old queue>

If you wish to change this after the upgrade, please see Configuring Agents and Capabilities and Specifying a Plan's Capability Requirements.

Conversion of Builders to Capabilities

Prior to Bamboo 2.0, your builders (e.g. Maven) were defined globally. In Bamboo 2.0, builders are now defined as agent capabilities and specified as plan requirements.

As part of the upgrade process,

- Each of your builders will be converted to a shared local capability (that is, it will apply to every local agent).
- Every plan will continue to have the same builder that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

Conversion of JDKs to Capabilities

Prior to Bamboo 2.0, your JDKs (e.g. JDK 1.5) were defined globally. In Bamboo 2.0, JDKs are now defined as agent capabilities and specified as plan requirements.

As part of the upgrade process,

- Each of your JDKs will be converted to shared local capabilities (that is, it will apply to every local agent).
- Upon conversion, the labels of each of your JDKs will upgraded to the Bamboo 2.0 JDK label format, (i.e. 'JDK 9.9.9_99').
- Upon conversion, two more generic versions of the labels will be created for each JDK, (i.e. 'JDK 9.9' and 'JDK').
- Every plan will have its requirements upgraded, to keep the association with the same JDK that it had before the upgrade.

If you wish to change this after the upgrade, please see Configuring Capabilities and Specifying a Plan's Capability Requirements.

Bamboo 1.2 Release Notes

Bamboo 1.2 can be downloaded here, and is of course free to all customers who purchased their Bamboo licence or maintenance after July 9, 2006.

When upgrading, please refer to the Bamboo 1.2 Upgrade Guide.
Permissions (global and plan-based)

Different organisations, and different projects, have different security requirements. Some information can be made public, while sensitive information may need to be confined to a particular group of people.

Bamboo 1.2 gives you the ability to set security on individual build plans, as well as on your entire Bamboo system:

- Plan permissions allow your chosen users to perform a particular operation in relation to a particular build plan (e.g. view its build results).
- Global permissions allow your chosen users to perform a particular operation in relation to Bamboo as a whole.

External database support

Bamboo ships with a built-in HSQL database, which is well suited to evaluation purposes. When deploying Bamboo in production, however, you will probably prefer to connect Bamboo to an enterprise database of your choice.

Bamboo 1.2 now includes support for MySQL and Postgres. If you need to use a different database, generic instructions for connecting Bamboo to an unsupported database are also provided.

Perforce triggering support

We are pleased to announce that Bamboo builds can now be triggered by Perforce repositories (previously only Subversion and CVS repositories were supported).

Scheduled backups

You can now schedule your Bamboo data exports to occur automatically at a convenient time:
New Bundled NAnt plugin

Want to build your .Net projects on Bamboo? Now you can, with the NAnt plugin, which comes bundled by default with Bamboo 1.2.

On the topic of plugins, have you checked out Bamboo Extensions space, home to a whole host of cool Bamboo plugins?

Other updates and bug fixes

On top of these features, Bamboo 1.2 also includes a host of minor new features, improvements and bug fixes:

<table>
<thead>
<tr>
<th>JIRA Issues (50 issues)</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1683</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1426</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1422</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1417</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1415</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1406</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1399</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1395</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1384</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1382</td>
<td>Build Labeller plugin fails to validate regex pattern</td>
<td>Resolved</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>BAM-1380</td>
<td>Exporting from an instance with large ZIP artifacts may fail</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1374</td>
<td>Delete recovery mechanism in Bamboo</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1373</td>
<td>Bamboo throws nullpointer exceptions while deleting build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1366</td>
<td>NullPointerException after clicking “1. Plan Details” Tab</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1348</td>
<td>Bamboo shows &quot;null build&quot; in RSS header (for project builds).</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1346</td>
<td>Single quotes in logs are prefixed with a back slash \</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1345</td>
<td>Subscribing to a RSS feed from a plan page leads to a 404 page not found exception</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1340</td>
<td>Deadlock issue while view currently running build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1337</td>
<td>REST API invalid user id error not handled correctly</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1336</td>
<td>REST API documentation not accurate</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1328</td>
<td>Look at Clover XML Even If Build Fails</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1322</td>
<td>Bamboo error when browsing authors</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1309</td>
<td>Edit build notification screen has no title</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1302</td>
<td>Perforce depot access will fail if the depot contains no workspace mapping</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1298</td>
<td>Bamboo doesn’t URL encode &quot;&quot;</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1282</td>
<td>Bamboo goes through password reminder even if no mail server configured</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1259</td>
<td>deleting a build plan caused NPE</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1255</td>
<td>Duplicate Email Notifications</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1253</td>
<td>Allow for expiry of just the artifacts</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1250</td>
<td>Null Pointer error in GetReturnURL</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1184</td>
<td>Option to expire artifacts from previous builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1152</td>
<td>Bamboo should accept repository triggers for Perforce builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1130</td>
<td>access control to specific plan (based on groups)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1077</td>
<td>Change terminology/function of Perforce Source Repository page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1068</td>
<td>New mime types for bamboo artifact downloads</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1042</td>
<td>Modify bamboo.sh to start the JVM in ‘server’ mode</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1030</td>
<td>Ability to download particular build log file</td>
<td>Closed</td>
</tr>
<tr>
<td>BAM-957</td>
<td>Ability to configure a database</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-906</td>
<td>Automate backups with a task scheduler</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-859</td>
<td>Downloadable logs</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-840</td>
<td>JUnit XML Improperly Parsed</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-798</td>
<td>Sub Menu Tabs move from side to side in IE</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-752</td>
<td>Perforce Configuration should allow passwords</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-701</td>
<td>Native support for NAnt builder</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-525</td>
<td>Restrict access to statistics</td>
<td>Resolved</td>
</tr>
</tbody>
</table>
Bamboo 1.2 Upgrade Guide

Upgrading from Bamboo 1.1.2 to 1.2

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

- If you are using plugins, please make sure that your plugins are compile against 1.2 before upgrading.
- Crowd on Bamboo 1.2
  If you are using Bamboo with Crowd, please make sure that you upgrade to Crowd 1.1.2 before upgrading Bamboo.
- Bamboo on Tomcat 5
  If you are running Bamboo on Tomcat 5, please follow the instructions on this page.

Please note that the upgrade process may take a while to complete.

Upgrading from Bamboo 1.1.1 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.

- If you're upgrading from Bamboo 1.0.x to Bamboo 1.2, please upgrade to 1.1.2 first. There is an issue with the upgrade process from the 1.0.x series that we're currently looking into.

Bamboo 1.2 Plugin Interface Changes

Below are details of plugin interface changes with Bamboo 1.2

Notification Condition

The method getTextEmail has changed from

```java
public void getTextEmail(Event event, Email email);
```

to

```java
public Email getTextEmail(Event event, Email email);
```

It now requires you to return the email object with the content populated (body, subject mimeType etc)

Bamboo 1.2.4 Release Notes
Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

17 October 2007

Atlassian is proud to announce the release of Bamboo 1.2.4. This point release includes more than 20 minor fixes and improvements. Bamboo 1.2.4 can be downloaded here. When upgrading, please refer to the Bamboo 1.2.4 Upgrade Guide.

Updates and issues fixed

<table>
<thead>
<tr>
<th>JIRA Issues (28 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>BAM-2048</td>
</tr>
<tr>
<td>BAM-2041</td>
</tr>
<tr>
<td>BAM-1821</td>
</tr>
<tr>
<td>BAM-1812</td>
</tr>
<tr>
<td>BAM-1808</td>
</tr>
<tr>
<td>BAM-1799</td>
</tr>
<tr>
<td>BAM-1794</td>
</tr>
<tr>
<td>BAM-1788</td>
</tr>
<tr>
<td>BAM-1742</td>
</tr>
<tr>
<td>BAM-1726</td>
</tr>
<tr>
<td>BAM-1711</td>
</tr>
<tr>
<td>BAM-1700</td>
</tr>
<tr>
<td>BAM-1698</td>
</tr>
<tr>
<td>BAM-1697</td>
</tr>
<tr>
<td>BAM-1695</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>BAM-1693</td>
</tr>
<tr>
<td>BAM-1687</td>
</tr>
<tr>
<td>BAM-1684</td>
</tr>
<tr>
<td>BAM-1666</td>
</tr>
<tr>
<td>BAM-1591</td>
</tr>
<tr>
<td>BAM-1583</td>
</tr>
<tr>
<td>BAM-1515</td>
</tr>
<tr>
<td>BAM-1481</td>
</tr>
<tr>
<td>BAM-1222</td>
</tr>
<tr>
<td>BAM-1019</td>
</tr>
<tr>
<td>BAM-411</td>
</tr>
</tbody>
</table>

**Bamboo 1.2.4 Upgrade Guide**

*Upgrading from Bamboo 1.2.x to 1.2.4*

Please follow the Bamboo Generic Upgrade Guide, plus:

![Bamboo on Tomcat 5](image)

If you are running Bamboo on Tomcat 5, please follow the instructions on this page.

*Upgrading from Bamboo 1.1.x or earlier*

In addition to the above, please read the Bamboo 1.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 1.2.3 Release Notes**

![Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!](image)

*12 September 2007*

Atlassian is proud to announce the release of Bamboo 1.2.3. This point release includes more than 20 minor fixes and improvements. Most notably, for greater flexibility when configuring a build plan, `variables` can now be used in a number of different places. Bamboo 1.2.3 can be downloaded here. When upgrading, please refer to the Bamboo 1.2.3 Upgrade Guide.
<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
<th>Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1680</td>
<td>BAM-1680</td>
<td>SSO with Crowd causes NoSuchMethodError</td>
<td>Unassigned</td>
<td>Arjan Schaaf</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Resolved</td>
<td>Sep 07, 2007</td>
<td>Sep 10, 2007</td>
</tr>
<tr>
<td>BAM-1668</td>
<td>BAM-1668</td>
<td>Bamboo cant parse test results, where method name is &quot;Test&quot;</td>
<td>Adrian Hempel</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Sep 03, 2007</td>
<td>Sep 03, 2007</td>
</tr>
<tr>
<td>BAM-1660</td>
<td>BAM-1660</td>
<td>Duplicate Path variable(s) added by Bamboo when executing Build on windows</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 31, 2007</td>
<td>Sep 11, 2007</td>
</tr>
<tr>
<td>BAM-1654</td>
<td>BAM-1654</td>
<td>Upgrade Serpah dependency from 0.7.17 to 0.7.23</td>
<td>Edwin Wong</td>
<td>Justin Koke</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 29, 2007</td>
<td>Sep 10, 2007</td>
</tr>
<tr>
<td>BAM-1646</td>
<td>BAM-1646</td>
<td>Implement rules to determine whether an artifact is to be downloaded or viewed in the browser</td>
<td>Brydie McCoy</td>
<td>Adrian Hempel</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 28, 2007</td>
<td>Oct 21, 2007</td>
</tr>
<tr>
<td>BAM-1638</td>
<td>BAM-1638</td>
<td>Bamboo fails to export when no plans are configured</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 27, 2007</td>
<td>Aug 28, 2007</td>
</tr>
<tr>
<td>BAM-1634</td>
<td>BAM-1634</td>
<td>Bamboo will not export under JDK 1.4 using JBoss 4.0.5</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 26, 2007</td>
<td>Sep 03, 2007</td>
</tr>
<tr>
<td>BAM-1625</td>
<td>BAM-1625</td>
<td>AccessLogFilter is configured but does not show &lt;user&gt; &lt;curl&gt; &lt;starting memory free in bamboo.log</td>
<td>Brydie McCoy</td>
<td>Levent Tutar</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 22, 2007</td>
<td>Aug 27, 2007</td>
</tr>
<tr>
<td>BAM-1596</td>
<td>BAM-1596</td>
<td>Bamboologs show a harmless LicenseException during Setup Wizard</td>
<td>Adrian Hempel</td>
<td>Adrian Hempel</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Resolved</td>
<td>Aug 14, 2007</td>
<td>Aug 14, 2007</td>
</tr>
<tr>
<td>BAM-1563</td>
<td>BAM-1563</td>
<td>Incorrect format for Build Time when configuring a plan</td>
<td>Brydie McCoy</td>
<td>James Odeen</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Resolved</td>
<td>Aug 06, 2007</td>
<td>Aug 14, 2007</td>
</tr>
<tr>
<td>BAM-1559</td>
<td>BAM-1559</td>
<td>Build results list sorted by build start date / time, displays completed time</td>
<td>Brydie McCoy</td>
<td>Mark Chaimungkalanont</td>
<td>Fixed</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 05, 2007</td>
<td>Aug 12, 2007</td>
</tr>
</tbody>
</table>
Bamboo 1.2.3 Upgrade Guide

Upgrading from Bamboo 1.2.x to 1.2.3

Please follow the Bamboo Generic Upgrade Guide, plus:

**Bamboo on Tomcat 5**
If you are running Bamboo on Tomcat 5, please follow the instructions on this page.

Upgrading from Bamboo 1.1.x or earlier

In addition to the above, please read the Bamboo 1.2 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Bamboo 1.2.2 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.2.2!

**Major features include:**
- Bulk editing of plan permissions.
- Administrators can now change users’ passwords.
- Improved caching on the dashboard, for better performance.

**Major fixes include:**
- Import and export when integrated with LDAP or Crowd.
- More import and export fixes.
- Users can now IM with Crowd integrated.

**Updates and Issues fixed**

<table>
<thead>
<tr>
<th>JIRA Issues (23 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Exports (Scheduled back-ups) do not work</td>
</tr>
<tr>
<td>Issue Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>BAM-1550</td>
</tr>
<tr>
<td>BAM-1547</td>
</tr>
<tr>
<td>BAM-1436</td>
</tr>
<tr>
<td>BAM-1285</td>
</tr>
<tr>
<td>BAM-1252</td>
</tr>
<tr>
<td>BAM-1188</td>
</tr>
<tr>
<td>BAM-1086</td>
</tr>
</tbody>
</table>
Bamboo 1.2.2 Upgrade Guide

Upgrading from Bamboo 1.2 (or 1.2.1) to 1.2.2

Please follow the Bamboo Generic Upgrade Guide, plus:

⚠️ Bamboo on Tomcat 5
If you are running Bamboo on Tomcat 5, please follow the instructions on this page.

Upgrading from Bamboo 1.1 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.

Bamboo 1.2.1 Release Notes

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.2.1! Bamboo 1.2.1 is mainly a bug fix release.

It is strongly recommended that you upgrade to Bamboo 1.2.1! It contains a fix to a critical security exploit in the system.

Major fixes include:

- Security exploit in Webwork 2.2.
- JDK 1.4 support
- Import & Export of build plan dependencies
- Upgrading from 1.0.x to 1.2.

Updates and Issues fixed

<table>
<thead>
<tr>
<th>JIRA Issues (12 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>BAM-1477</td>
</tr>
<tr>
<td>BAM-1475</td>
</tr>
<tr>
<td>BAM-1471</td>
</tr>
<tr>
<td>BAM-1470</td>
</tr>
<tr>
<td>BAM-1460</td>
</tr>
<tr>
<td>BAM-1455</td>
</tr>
<tr>
<td>BAM-1453</td>
</tr>
<tr>
<td>BAM-1452</td>
</tr>
<tr>
<td>BAM-1437</td>
</tr>
<tr>
<td>BAM-1359</td>
</tr>
<tr>
<td>BAM-978</td>
</tr>
<tr>
<td>BAM-977</td>
</tr>
</tbody>
</table>
Bamboo 1.2.1 Upgrade Guide

Upgrading from Bamboo 1.2 to 1.2.1

Please follow the Bamboo Generic Upgrade Guide

Upgrading from Bamboo 1.1 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.

Bamboo 1.1 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

The Atlassian Bamboo team is proud to announce the release of Bamboo 1.1! This release contains a whole host of new features targetted to make your build plans even more powerful and flexible.

Want to see Bamboo 1.1 in action? Check out our live opensource instance.

Advanced Notifications

In this release, we have extended Bamboo notifications framework to provide more flexibility, allowing you to select the how, who and when of notifications.

Notification Rules

Rather than having static fields for emails/IM recipients, Bamboo now allows you to define your own notifications for your build plans as a set of rules, giving you greater granularity in controlling exactly which recipient gets notified and when.

<table>
<thead>
<tr>
<th>Notification Trigger</th>
<th>Notification Recipients</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed Builds And First Successful</td>
<td>Users:</td>
<td>Editing and removing notifications, and adding new ones.</td>
</tr>
<tr>
<td>Notify After X Failed Builds</td>
<td>Roles:</td>
<td>Editing and removing notifications, and adding new ones.</td>
</tr>
<tr>
<td>All Completed Builds</td>
<td>Roles:</td>
<td>Editing and removing notifications, and adding new ones.</td>
</tr>
</tbody>
</table>

Notification Triggers

In release 1.1, we introduce notification triggers, defining exactly when you would like a notification to be sent by Bamboo. By default, you can select a notification to be sent on "all builds completion", "after X failed builds" or "failed builds and first successful build". Want more customised triggers? You can now write your own as a notification condition plugin.

Add Build Notification

Notification Trigger: All Completed Builds

Roles: Watcher - Users who have marked this build as failed

Notification Preferences

Different users prefer to get notified in different ways. Bamboo now lets you control that, via the new user notification preferences.
Dynamic recipients

Only want to receive a notification when you have committed against the build? Want to opt-in to receive notifications on the build plan that you are keeping an eye on? Bamboo 1.1 introduces two new dynamic recipient roles: committers (those users who have committed to the plan triggering the particular build to execute) and watchers (those users who have marked the build plan as their favourite), which allow you to do just that!

Roles:
- Committer - Users who have committed to the build
- Watcher - Users who have marked this build as favourite

Build Metadata

Every build process is different, and each build will have its own information that you may want to keep track of and use on top of the information that Bamboo stores about your build. This is particularly the case if you run custom plugins in your build process.

Pass them to your build

One way to use your build metadata is to pass it along to your builder as a property or target. To do this, you simply specify your variables in your target (or goal) field in your builder configuration. During build execution, the variables will be substituted with the actual values from your build metadata.

Global Variables

Bamboo 1.1 also allows you the option to specify variables globally. When a build begins, the global variables will be populated to the build's metadata. This is a handy option for you to control many plans in one go.

Global Variables

You can use this page to view, add and delete global variables. Global variables are available on every build run in Bamboo.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>bambooVersion</td>
<td>1.1</td>
<td>Delete</td>
</tr>
</tbody>
</table>

View your metadata

Use the "Metadata" tab to keep track of all of your build's metadata.
In this release, we also introduce the file trigger inclusion/exclusion filter. Instead of listening and picking up all changes from a repository, you can now use regex patterns to define those files which you do (or don't) want to trigger builds.

**More pluggability**

In release 1.1, we have added more plugin points to make Bamboo even more extensible than before. On top of the notification condition plugin point, we have also added pre-build action plugins, as well as repository plugins.

- **Repository Plugins** Not using SVN, CVS, or Perforce? You can now write a plugin to integrate with your very own source control.
- **Pre-build Plugins** Similar to the post-build action plugin, the pre-build action plugin will allow you to perform any custom task you may wish. The only difference is, of course, that it occurs before the build execution begins.

**Improved Maven 2 error log parsing**

Bamboo now intelligently parses the Maven 2 error log for possible errors in the build errors log, giving you a better view of what really went wrong in your build summary.

**LDAP and external user management support**

In release 1.1, we have improved our user management capability to support externally sourced users and groups, including LDAP, and Crowd.

**Performance of Dashboard**

With this release, we have also made significant performance improvements to the dashboard, which should see its load times reduce dramatically.

**Other updates and bug fixes**

On top of these features, we have also made a whole host of bug fixes, with over x bugs fixed since release 1.0.5.
<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1191</td>
<td>IM notifications should allow group to be un-selected</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1189</td>
<td>Add global variables to be used in plans</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1183</td>
<td>Tool tip in the dashboard for the plans</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1173</td>
<td>Add an option to use a non Ajax diashboard &quot;hideDashboard&quot;</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1163</td>
<td>Dashboard very slow</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1155</td>
<td>Error cancelling a user preferences dialog</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1134</td>
<td>&quot;operation not permitted&quot; clicking Completed Builds tab</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1124</td>
<td>Crowd 1.0.6 + Bamboo 1.0.4 NoClassDefFoundError:</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1115</td>
<td>Artifact URL fails to escape invalid characters</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1090</td>
<td>Allow pluggable pre-build actions</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1088</td>
<td>Hide User Info From Non-Admins</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1081</td>
<td>Triggering a build only on certain commits</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1080</td>
<td>Lock obtain timed out error</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1073</td>
<td>No way to pass proxy information down to Ant</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1059</td>
<td>Implementation of new style of Notifications</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1058</td>
<td>Error viewing Build: Expression failing SinceBuild.buildResultsSummary is undefined</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1053</td>
<td>ANT_HOME is used for ant</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1032</td>
<td>NullPointerException accessing RSS feed</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1029</td>
<td>LDAP integration</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1020</td>
<td>Inconsistent Test Results between builds</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1016</td>
<td>Allow include/exclude patterns for triggering build</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1011</td>
<td>bamboo.home is not a valid environment variable</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1010</td>
<td>Error summary should parse logs in Maven 2 builds</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>BAM-996</td>
<td>FreeMarker template error in Completed Build Results screen</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-952</td>
<td>Bamboo uses older m2 version even when maven 2.0.5 builder configured for the plan</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-950</td>
<td>Out of memory error while processing Clover 2.0a4 results</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-942</td>
<td>Upgrade jfreechart version due to concurrency bug</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-940</td>
<td>Make Repository pluggable</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-938</td>
<td>Editing LDAP users in Bamboo</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-934</td>
<td>OutOfMemory while checking a build result</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-915</td>
<td>Time values in Report data table should be more readable</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-856</td>
<td>Add a simple batch / shell script to run the standalone</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-847</td>
<td>Remote API for manual checkout and build.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-819</td>
<td>'Important Files and Directories'</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>BAM-792</td>
<td>Pass in bamboo parameters such as project and plan key to builders</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-787</td>
<td>add screenshots (when Clover data is available)</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-767</td>
<td>We able to add custom project specific Build Telemetry data</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-729</td>
<td>make terminolody consistent (&quot;build&quot; vs. &quot;plan&quot;)</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-681</td>
<td>Checkboxes for build queues</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-680</td>
<td>Picker for e-mail notifications</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-641</td>
<td>Ability to ignore file patterns for updates</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-633</td>
<td>Add Recent Activity tab for the dashboard</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-604</td>
<td>API access to the SCM repository's build identifier for a given build</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-574</td>
<td>The interface used to provide a view of artifacts has issues dealing with subdirectories</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-561</td>
<td>Configure dependencies in the opposite direction</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-517</td>
<td>Email Notifications</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-493</td>
<td>Better handling of large number of unit tests</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-309</td>
<td>Would like to see a feature list along the lines of <a href="http://docs.codehaus.org/display/DAMAGECONTROL/Continuous+Integration+Server+Feature+Matrix">http://docs.codehaus.org/display/DAMAGECONTROL/Continuous+Integration+Server+Feature+Matrix</a></td>
<td>Resolved</td>
<td></td>
</tr>
</tbody>
</table>
BAM-146 | Build time graph should use minutes in scale |Resolved
---|---|---
BAM-57 | Improve configuration of notifications |Resolved

If you want to check out a live Bamboo instance, take a look at our opensource instance.

**Bamboo 1.1 Upgrade Guide**

**Upgrading from Bamboo 1.0.5 to 1.1**

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the [Bamboo Generic Upgrade Guide](#).

Please note that the upgrade process may take a while to complete.

**Upgrading from Bamboo 1.0.5 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](#).

**Bamboo 1.1.2 Release Notes**

✅ Bamboo 2.4 has been released. Read the full [Bamboo 2.4 Release Notes and Upgrade Guide](#). *Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!*

Atlassian is proud to announce the release of Bamboo 1.1.2! Bamboo 1.1.2 is mainly a bug fix release.

Major fixes include:

- Export - Windows Export caused some problems, these are now fixed
- Subversion - We have ungraded to the latest SVNKit to incorporate many of their bug fixes
- Fisheye Integration - The Fisheye links for perforce have been fixed
- Character Encoding - Bamboo now lets you use all Unicode characters
- LDAP - More LDAP fixes!

**Updates and Issues fixed**

<table>
<thead>
<tr>
<th>JIRA Issues (21 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>BAM-1296</td>
</tr>
<tr>
<td>BAM-1289</td>
</tr>
<tr>
<td>BAM-1279</td>
</tr>
<tr>
<td>BAM-1265</td>
</tr>
<tr>
<td>BAM-1262</td>
</tr>
<tr>
<td>BAM-1260</td>
</tr>
<tr>
<td>BAM-1256</td>
</tr>
<tr>
<td>BAM-1254</td>
</tr>
<tr>
<td>BAM-1238</td>
</tr>
<tr>
<td>BAM-1235</td>
</tr>
<tr>
<td>BAM-1232</td>
</tr>
<tr>
<td>BAM-1231</td>
</tr>
<tr>
<td>BAM-1229</td>
</tr>
</tbody>
</table>
BAM-1138  File Version Number problems with Perforce  Resolved
BAM-1094  start-bamboo wrapper uses false uname syntax  Resolved
BAM-1065  Bamboo reports twice as many unit test  Resolved
BAM-1046  Valid cron expression not really valid  Resolved
BAM-973   Users with non standard characters in their names show up corrupted.  Resolved
BAM-971   JIRA issue parser should look for numbers in project key as well.  Resolved
BAM-745   Encoding (of project name)  Resolved
BAM-518   Unable to copy build artifacts  Resolved

Bamboo 1.1.2 Upgrade Guide

Upgrading from Bamboo 1.1.1 to 1.1.2

Please follow the Bamboo Generic Upgrade Guide

Upgrading from Bamboo 1.1 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.

Bamboo 1.1.1 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.1.1! Bamboo 1.1.1 is mainly a bug fix release.

Major fixes include:
- LDAP - Many problems with LDAP integration have been overcome
- IMPORT/EXPORT - Several import fixes were implemented
- CVS - CVS change detection has been improved

Updates and Issues fixed

<table>
<thead>
<tr>
<th>JIRA Issues (21 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>BAM-1857</td>
</tr>
<tr>
<td>BAM-1244</td>
</tr>
<tr>
<td>BAM-1234</td>
</tr>
<tr>
<td>BAM-1233</td>
</tr>
<tr>
<td>BAM-1227</td>
</tr>
<tr>
<td>BAM-1226</td>
</tr>
<tr>
<td>BAM-1224</td>
</tr>
<tr>
<td>BAM-1216</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>BAM-1215</td>
</tr>
<tr>
<td>BAM-1206</td>
</tr>
<tr>
<td>BAM-1200</td>
</tr>
<tr>
<td>BAM-1195</td>
</tr>
<tr>
<td>BAM-1192</td>
</tr>
<tr>
<td>BAM-1165</td>
</tr>
<tr>
<td>BAM-1154</td>
</tr>
<tr>
<td>BAM-1151</td>
</tr>
<tr>
<td>BAM-1120</td>
</tr>
<tr>
<td>BAM-1117</td>
</tr>
<tr>
<td>BAM-1066</td>
</tr>
<tr>
<td>BAM-825</td>
</tr>
<tr>
<td>BAM-505</td>
</tr>
</tbody>
</table>

**Bamboo 1.1.1 Upgrade Guide**

*Upgrading from Bamboo 1.1 to 1.1.1*

Please follow the Bamboo Generic Upgrade Guide

*Upgrading from Bamboo 1.0.5 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.

**Bamboo 1.0 Release Notes**

[Check out Bamboo's features and see what it can do for you, please visit our Feature Tour.](#)

[Upgrading from a pre-release version? Please see the Bamboo 1.0 Upgrade Guide.](#)

[Doing an upgrade? Make sure you re-index Bamboo by going to the Administration section and hitting 'Re-index'.](#)
Changes since RC2

The final steps to 1.0 since RC2 has been focused on resolving issues. Release 1.0 includes over 30 issues resolved.

In addition, the 1.0 release also sports another revised "All Plans" tab in the dashboard.

**Atlassian Bamboo**

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-879</td>
<td>Where are my nice build result commit message tool tips?</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-878</td>
<td>NumberFormatException for Test</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-876</td>
<td>Move the Clover plugin to opensource as an example of a Bamboo plugin</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-875</td>
<td>User page no longer show tabs with author information on them</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-873</td>
<td>All Projects table shows the 'little hand' icon over rows that can't be expanded</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-867</td>
<td>test mail should contain clickable base url</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-860</td>
<td>New more condensed dashboard</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-857</td>
<td>Document our external Javascript widgets</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-839</td>
<td>Tests Page Has URL Escapes</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-837</td>
<td>Allow Properties to be Passed to Ant</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-835</td>
<td>Build completed time on summary page is actually build start time</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-821</td>
<td>Unable to export build configuration - no info on how to repair</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-807</td>
<td>Bamboo passes bad parameter diff ViewVC</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-803</td>
<td>Use minified version of js libs in 1.0 final</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-796</td>
<td>Number Format Exception</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-788</td>
<td>IM bot should reconnect before sending message if it was disconnected</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Other updates and bug fixes.
<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-783</td>
<td>Build test result tab taking to long to load.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-779</td>
<td>Report graphs are not displaying data: build duration and # of tests</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-778</td>
<td>Clicking the previous build button while viewing changes - got stacktrace</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-775</td>
<td>Disabled plans should have visual cue on the Summary page.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-770</td>
<td>Bamboo User ID should also be a repository alias.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-723</td>
<td>Plugin Guide</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-722</td>
<td>Administrator's Guide</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-721</td>
<td>Bamboo User Guide</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-720</td>
<td>Upgrade Guide (generic)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-719</td>
<td>Release Notes &amp; Upgrade Guides: reformat as per JIRA's/Confluence's</td>
<td>Closed</td>
</tr>
<tr>
<td>BAM-718</td>
<td>Installation Guide (WAR)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-717</td>
<td>Installation Guide (Standalone)</td>
<td>Closed</td>
</tr>
<tr>
<td>BAM-706</td>
<td>Added two builds to the queue, canceled the 2nd one, got a hibernate exception</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-690</td>
<td>Improve validation for CVS :ext</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-683</td>
<td>Allow multiple tabs on dashboard page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-628</td>
<td>Test reponsibility for a build summary</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-610</td>
<td>HTML in test output log is doubly-escaped</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-499</td>
<td>Use 307 response code instead of meta-refresh when hitting path of bamboo</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-433</td>
<td>exe installer home setting issue</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-16</td>
<td>Ability to externally embed full build status</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

**Bamboo 1.0 Upgrade Guide**

**Upgrading from Bamboo 1.0-RC2 to 1.0**

Please follow the Bamboo Generic Upgrade Guide

⚠️ You will need to reindex your data after the upgrade is complete and Bamboo has started. To do this, go to the indexing page under the Administration section in Bamboo.

**Upgrading from Bamboo 1.0-RC1 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.

**Bamboo 1.0.5 Release Notes**

✅ Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.0.5! Bamboo 1.0.5 is mainly a bug fix release related to subversion connectivity issues.

**Updates and Issues fixed**

<table>
<thead>
<tr>
<th>JIRA Issues (11 issues)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Summary</td>
</tr>
</tbody>
</table>

Bamboo 1.0.5 Upgrade Guide

Upgrading from Bamboo 1.0.4 to 1.0.5

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

Upgrading from Bamboo 1.0.4 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

Bamboo 1.0.4 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.0.4! Bamboo 1.0.4 is mainly a bug fix release with over 10 issues resolved.

In this release, the focus has been on resolving connectivity issues with Subversion and Perforce

Perforce Improvements

There have been a few changes in Bamboo’s Perforce integration

- Bamboo will now cache the client root rather than polling the repository continuously to obtain it
  This reduces the load on the Perforce server considerably. However, if you change the root in the client definition on Perforce, Bamboo will require a restart to pick up the change
- Bamboo now uses changelist numbers to detect source code changes rather than a timestamp
  This will avoid all sorts of problems that occur when the Bamboo server clock and Perforce server clock are out of sync
- Bamboo now picks up multi line change descriptions from Perforce
- Bamboo can now generate web urls for perforce files when using Fisheye

Updates and Issues fixed

<table>
<thead>
<tr>
<th>JIRA Issues (14 issues)</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1113</td>
<td></td>
<td>Perforce modifications not causing build</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1096</td>
<td></td>
<td>Change the way bamboo detects changes in perforce</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1085</td>
<td></td>
<td>Subversion code refresh failing to pick up new revisions</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1078</td>
<td></td>
<td>BuildChangeDetector continuously polling Perforce repository</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
BAM-1056 Failed to get the build source code: svn: report aborted
BAM-1028 Bamboo throws exception when it polls Subversion repository
BAM-979 Different time zone on Perforce server does not work
BAM-974 Bamboo penetrated Perforce server with repeated requests on plan creation
BAM-890 SVN triggered update failing.
BAM-871 Manual builds still poll the Perforce server
BAM-829 Only first line of change description is displayed for Perforce changes
BAM-823 Web Repository URL is not persisted for Perforce repositories
BAM-750 Perforce changes are not displayed when a manual build is executed
BAM-747 Perforce repository polling build plan not building

Bamboo 1.0.4 Upgrade Guide

Upgrading from Bamboo 1.0.3 to 1.0.4

In this version, an upgrade task has been added to update Perforce plans to use the change list number rather than the timestamp when detecting changes. Please ensure that you have connectivity to the Perforce server before you upgrade.

If Bamboo encounters any errors during the upgrade task it will set the Perforce plan's last change list number to 0. This means that the next time you build that plan there may be some unusual results (e.g. picking up every single change list). Once this build is complete normal behaviour will resume.

It is strongly recommended that you back up your xml-data directory before proceeding. For full instructions please follow the Bamboo Generic Upgrade Guide.

Upgrading from Bamboo 1.0.2 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

Bamboo 1.0.3 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don’t have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.0.3! Bamboo 1.0.3 is mainly a bug fix release with over 10 issues resolved.

In this release, the focus has been on improving SVN integration (detection of SVN Externals) and CVS integration (detection of ampersand modules).

Updates and Issues fixed

<table>
<thead>
<tr>
<th>JIRA Issues (14 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-1063</td>
</tr>
<tr>
<td>BAM-1017</td>
</tr>
<tr>
<td>BAM-1008</td>
</tr>
<tr>
<td>BAM-1005</td>
</tr>
<tr>
<td>BAM-1000</td>
</tr>
<tr>
<td>Key</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>BAM-993</td>
</tr>
<tr>
<td>BAM-986</td>
</tr>
<tr>
<td>BAM-976</td>
</tr>
<tr>
<td>BAM-964</td>
</tr>
<tr>
<td>BAM-884</td>
</tr>
<tr>
<td>BAM-882</td>
</tr>
<tr>
<td>BAM-862</td>
</tr>
<tr>
<td>BAM-844</td>
</tr>
<tr>
<td>BAM-710</td>
</tr>
</tbody>
</table>

**Bamboo 1.0.3 Upgrade Guide**

**Upgrading from Bamboo 1.0.2 to 1.0.3**

In this version, an upgrade task has been added to upgrade your CVS commit files data to a correct path (which includes module name). This may take a while to run, and it is strongly recommended that you back up your xml-data directory before proceeding. For fuller instructions please follow the Bamboo Generic Upgrade Guide.

**Upgrading from Bamboo 1.0.1 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.

**Bamboo 1.0.2 Release Notes**

Atlassian is proud to announce the release of Bamboo 1.0.2! Bamboo 1.0.2 is mainly a bug fix release with over 10 issues resolved.

In addition, Bamboo 1.0.2 also sees added support for ssh private key authentication for both Subversion and CVS repositories.

**Updates and Issues fixed**

<table>
<thead>
<tr>
<th>JIRA Issues (19 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>BAM-1098</td>
</tr>
<tr>
<td>BAM-1031</td>
</tr>
<tr>
<td>BAM-1026</td>
</tr>
<tr>
<td>BAM-1023</td>
</tr>
<tr>
<td>BAM-1006</td>
</tr>
<tr>
<td>BAM-1002</td>
</tr>
<tr>
<td>BAM-994</td>
</tr>
</tbody>
</table>
### Bamboo 1.0.2 Upgrade Guide

**Upgrading from Bamboo 1.0.1 to 1.0.2**

Please follow the Bamboo Generic Upgrade Guide

**Upgrading from Bamboo 1.0.1 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.

### Bamboo 1.0.1 Release Notes

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide.

Don't have Bamboo 2.4? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the release of Bamboo 1.0.1! Bamboo 1.0.1 is largely a bug fix build with over 20 issues resolved, including:

- Support for SVN cached default authentication.
- IE7 Javascript issues.
- Startup Script issues.

**New startup procedures for Mac OS X and Linux distributions**

The Bamboo startup procedure for Mac OS X and Linux distributions have now changed. Instead of using the Java Service Wrapper by invoking `run-bamboo` (in Mac OS X) or `start-bamboo` in Linux, the default startup script has been replaced by a generic `bamboo.sh` script in the root Bamboo installation folder. Using this script bypasses the Java Service Wrapper.

**Usages for bamboo.sh**

- `start` - starts Bamboo
- `stop` - stops Bamboo
- `console` - runs Bamboo in the console
- `status` - checks the status of Bamboo.

The Java Service Wrapper is still available, and you can startup Bamboo with it if you so choose. To do this, simply run your startup command in the `/wrapper` folder rather than the installation root folder.

### Updates and Issues fixed.

<table>
<thead>
<tr>
<th>JIRA Issues (28 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>BAM-989</td>
</tr>
<tr>
<td>BAM-988</td>
</tr>
<tr>
<td>BAM-987</td>
</tr>
<tr>
<td>BAM-983</td>
</tr>
<tr>
<td>BAM-982</td>
</tr>
<tr>
<td>BAM-980</td>
</tr>
<tr>
<td>BAM-962</td>
</tr>
<tr>
<td>BAM-960</td>
</tr>
<tr>
<td>BAM-885</td>
</tr>
<tr>
<td>BAM-806</td>
</tr>
<tr>
<td>BAM-791</td>
</tr>
<tr>
<td>BAM-691</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>BAM-975</td>
</tr>
<tr>
<td>BAM-965</td>
</tr>
<tr>
<td>BAM-959</td>
</tr>
<tr>
<td>BAM-958</td>
</tr>
<tr>
<td>BAM-943</td>
</tr>
<tr>
<td>BAM-939</td>
</tr>
<tr>
<td>BAM-937</td>
</tr>
<tr>
<td>BAM-936</td>
</tr>
<tr>
<td>BAM-935</td>
</tr>
<tr>
<td>BAM-933</td>
</tr>
<tr>
<td>BAM-925</td>
</tr>
<tr>
<td>BAM-918</td>
</tr>
<tr>
<td>BAM-917</td>
</tr>
<tr>
<td>BAM-916</td>
</tr>
<tr>
<td>BAM-908</td>
</tr>
<tr>
<td>BAM-899</td>
</tr>
<tr>
<td>BAM-897</td>
</tr>
<tr>
<td>BAM-889</td>
</tr>
<tr>
<td>BAM-869</td>
</tr>
<tr>
<td>BAM-851</td>
</tr>
<tr>
<td>BAM-848</td>
</tr>
<tr>
<td>BAM-841</td>
</tr>
<tr>
<td>BAM-790</td>
</tr>
<tr>
<td>BAM-746</td>
</tr>
<tr>
<td>BAM-692</td>
</tr>
<tr>
<td>BAM-634</td>
</tr>
<tr>
<td>BAM-297</td>
</tr>
<tr>
<td>BAM-90</td>
</tr>
</tbody>
</table>

**Bamboo 1.0.1 Upgrade Guide**

**Upgrading from Bamboo 1.0 to 1.0.1**

Please follow the Bamboo Generic Upgrade Guide

> You will need to reindex your data after the upgrade is complete and Bamboo has started. To do this, go to the indexing page under the Administration section in Bamboo.

**Upgrading from Bamboo 1.0 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](#).
Bamboo 1.0-Beta Release Notes

The Atlassian Bamboo team is proud to announce the release of Bamboo 1.0 beta. This release includes over 40 bug fixes and improvements.

⚠️ Upgrading? Please see the Bamboo 1.0-Beta Upgrade Guide.

New in Release 1.0 - Beta

Anonymous access and sign on control.

In this release, you can now control whether your Bamboo is a public or private instance via the anonymous access and sign on options. Anonymous access allows users not signed in to view read only sections of Bamboo. Sign on allows users to create their own account for login. Disable these options to fully protect your Bamboo instance.

Auto favourite feature

Bamboo gets smarter with an auto-favourite marking feature. It'll mark those builds you commit against as your favourites.

Longest time to fix tests

Get a view of which tests in your builds are taking the longest the fix.

Other updates and bug fixes

<table>
<thead>
<tr>
<th>JIRA Issues (44 issues)</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Summary</td>
<td></td>
</tr>
<tr>
<td>BAM-714</td>
<td>Support for SVN File Protocol</td>
<td></td>
</tr>
<tr>
<td>BAM-697</td>
<td>CVS connection fails if password has @ in it</td>
<td></td>
</tr>
<tr>
<td>BAM-693</td>
<td>If project only has one plan, the project summary should redirect to the Plan Summary Page</td>
<td></td>
</tr>
<tr>
<td>BAM-685</td>
<td>Commit comments lose line breaks</td>
<td></td>
</tr>
<tr>
<td>BAM-684</td>
<td>Can't add builder of type Ant</td>
<td></td>
</tr>
<tr>
<td>BAM-676</td>
<td>Unable to re-index due to locked file</td>
<td></td>
</tr>
<tr>
<td>BAM-674</td>
<td>Build fails to start</td>
<td></td>
</tr>
<tr>
<td>BAM-668</td>
<td>JIRA tabs shows up regardless</td>
<td></td>
</tr>
<tr>
<td>BAM-661</td>
<td>Security and Login Improvements</td>
<td></td>
</tr>
<tr>
<td>BAM-659</td>
<td>Edit configuration needs formatting fixes</td>
<td></td>
</tr>
<tr>
<td>BAM-658</td>
<td>Project names not ordered in dropdown to create project</td>
<td></td>
</tr>
<tr>
<td>BAM-657</td>
<td>Typo and user interface improvement</td>
<td></td>
</tr>
<tr>
<td>BAM-656</td>
<td>No way to ‘complete’ setting up a project as ‘save’ hidden by javascript</td>
<td></td>
</tr>
<tr>
<td>BAM-652</td>
<td>Checkboxes dont work properly when removing dependant builds</td>
<td></td>
</tr>
<tr>
<td>BAM-647</td>
<td>Split webapp WAR module into a JAR and a WAR module</td>
<td></td>
</tr>
<tr>
<td>BAM-644</td>
<td>Script builder fails for windows</td>
<td></td>
</tr>
<tr>
<td>BAM-637</td>
<td>Reports build table not sorted</td>
<td></td>
</tr>
<tr>
<td>BAM-635</td>
<td>Auto favourite functionality</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Status</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>BAM-632</td>
<td>Breadcrumbs should have build numbers</td>
<td></td>
</tr>
<tr>
<td>BAM-631</td>
<td>Reports on top ten longest time to fix for tests</td>
<td></td>
</tr>
<tr>
<td>BAM-630</td>
<td>Test summary page still using the old style</td>
<td></td>
</tr>
<tr>
<td>BAM-625</td>
<td>Redirect after a plan is created</td>
<td></td>
</tr>
<tr>
<td>BAM-624</td>
<td>IE caches ajax response for comments and labels</td>
<td></td>
</tr>
<tr>
<td>BAM-622</td>
<td>Last screen of create build broken on Safari</td>
<td></td>
</tr>
<tr>
<td>BAM-616</td>
<td>Capture code changes for dependent and scheduled builds</td>
<td></td>
</tr>
<tr>
<td>BAM-612</td>
<td>Label grammar</td>
<td></td>
</tr>
<tr>
<td>BAM-611</td>
<td>Allow two character plan keys</td>
<td></td>
</tr>
<tr>
<td>BAM-609</td>
<td>error displaying build queue admin page</td>
<td></td>
</tr>
<tr>
<td>BAM-608</td>
<td>too many files open error</td>
<td></td>
</tr>
<tr>
<td>BAM-606</td>
<td>Invalid path to clover throws error</td>
<td></td>
</tr>
<tr>
<td>BAM-603</td>
<td>Accessing /api/index.action throws a freemaker error</td>
<td></td>
</tr>
<tr>
<td>BAM-602</td>
<td>Minor issues with the build status wizard section on the editBuildConfiguration.action page</td>
<td></td>
</tr>
<tr>
<td>BAM-595</td>
<td>Adding a comment from the Summary page doesn't work in IE</td>
<td></td>
</tr>
<tr>
<td>BAM-579</td>
<td>Make top right options clearer</td>
<td></td>
</tr>
<tr>
<td>BAM-572</td>
<td>Need option to disable signups</td>
<td></td>
</tr>
<tr>
<td>BAM-570</td>
<td>Testing for mail and IM servers should be more visible</td>
<td></td>
</tr>
<tr>
<td>BAM-569</td>
<td>Auto report grouping for Tests doesn't seem to work</td>
<td></td>
</tr>
<tr>
<td>BAM-566</td>
<td>The back button on create plan wizard clears previously selected values</td>
<td></td>
</tr>
<tr>
<td>BAM-563</td>
<td>Validation for report not being selected</td>
<td></td>
</tr>
<tr>
<td>BAM-543</td>
<td>Minor tweaks of the Admin pages</td>
<td></td>
</tr>
<tr>
<td>BAM-542</td>
<td>Plugin Points for Web Fragments</td>
<td></td>
</tr>
<tr>
<td>BAM-490</td>
<td>Ability to run a Bamboo in 'private mode'</td>
<td></td>
</tr>
<tr>
<td>BAM-162</td>
<td>Passwords in plaintext</td>
<td></td>
</tr>
<tr>
<td>BAM-91</td>
<td>Static files are not cached, increases size of downloads</td>
<td></td>
</tr>
</tbody>
</table>

**Bamboo 1.0-Beta Upgrade Guide**

**Upgrading from Bamboo 0.9 to 1.0-Beta**

Please follow the Bamboo Generic Upgrade Guide.

**Upgrading from Bamboo 0.8 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.

**Bamboo Release Summary**

This page shows the highlights of the major Bamboo releases.
Current Release

For information about the latest release, please go to the Release Notes.

Bamboo 2.4 — 6 October 2009

- Bamboo Gadgets in JIRA
- Clover Enhancements
- REST Improvements
- Runtime Log4j Configuration
- More in release notes

Bamboo 2.3 — 6 August 2009

- Dependency Blocking Strategies
- New Build Notifications and Queue Reordering
- Bulk Actions
- Multiple Elastic Images
- Elastic Instance Scheduling
- PHPUnit Builder
- Bamboo REST APIs
- Plugins Changes
- More in release notes

Bamboo 2.2 — 9 March 2009

- Elastic Bamboo
- Customisable Email Templates
- Build Comment Notification Event
- Hanging Build Detection Event
- Faster Artifact Transfer
- Dependent Builds
- Agent Improvements
- More in release notes

Bamboo 2.1 — 5 August 2008

- Link Issues and Builds
- Specify the Issues that are Fixed by a Build
- Track the Builds for your Projects and Versions
- View Issues under Development
- Post Change Detection Plugin Point
- More in release notes

Bamboo 2.0 — 14 April 2008

- Distributed builds
- Capability matching
- Memory usage improvements
- Parallel VCS updates and checkouts
- Ability to force a ‘clean build’
- Quiet Period functionality supported for Subversion & Perforce
- Bamboo Plugin for Confluence
- Support for Oracle and MS SQL Server databases
- Status Summary screens
- More in release notes

Bamboo 1.2 — 09 July 2007

- Permissions (global and plan-based)
- External database support
- Perforce triggering support
- Scheduled backups
- New Bundled NAnt plugin
- More in release notes

Bamboo 1.1 — 07 May 2007

- Advanced notifications - rules, triggers, preferences, dynamic recipients
- Build Metadata - pass them to your build, global variables, view them
- File Trigger Inclusions/Exclusions
- More pluggability
- Improved Maven 2 error log parsing
- LDAP and external user management support
- Dashboard loading has been improved
- More in release notes

Bamboo 1.0 — 20 February 2007
- All Plans tab
- More in release notes

Bamboo Upgrade Guides

You should read the general Bamboo Generic Upgrade Guide, if you are planning on upgrading your installation of Bamboo, as well as the upgrade guide for the version of Bamboo you are upgrading to. The upgrade guides for each of the Bamboo releases, can be found below:

- Previous Production Releases
- Previous Beta Releases

Please also read the relevant release notes for the version you are upgrading to.

Previous Production Releases

- Bamboo 2.4 Upgrade Guide
- Bamboo 2.3.1 Upgrade Guide
- Bamboo 2.3 Upgrade Guide
- Bamboo 2.2.4 Upgrade Guide
- Bamboo 2.2.3 Upgrade Guide
- Bamboo 2.2.2 Upgrade Guide
- Bamboo 2.2.1 Upgrade Guide
- Bamboo 2.1.5 Upgrade Guide
- Bamboo 2.1.4 Upgrade Guide
- Bamboo 2.2 Upgrade Guide
- Bamboo 2.1.3 Upgrade Guide
- Bamboo 2.1.2 Upgrade Guide
- Bamboo 2.1.1 Upgrade Guide
- Bamboo 2.1 Upgrade Guide
- Bamboo 2.0.6 Upgrade Guide
- Bamboo 2.0.5 Upgrade Guide
- Bamboo 2.0.4 Upgrade Guide
- Bamboo 2.0.3 Upgrade Guide
- Bamboo 2.0.2 Upgrade Guide
- Bamboo 2.0.1 Upgrade Guide
- Bamboo 2.0 Upgrade Guide
- Bamboo 1.2.4 Upgrade Guide
Before you begin
This upgrade guide contains generic upgrade tasks that are common across Bamboo versions. Please ensure that you also read the Upgrade Guide for the version you are upgrading to.

Bamboo 2.4 has been released. Read the full Bamboo 2.4 Release Notes and Upgrade Guide. Don’t have Bamboo 2.4? Take a look at the features of Bamboo’s latest major version and try it out!

Step 1. Identify your Bamboo directories

Go to the 'System Info' page in the 'Administration' menu of your Bamboo instance and note the location of the Bamboo Home, Build Data Path and Configuration Path directories:
Step 2. Shut down Bamboo

You need to shut down Bamboo before backing up or performing the upgrade.

Step 3. Back up Bamboo

Back up the three directories you identified in Step 1 — your Bamboo Home, Build Data Path and Configuration Path directories (for more information about these directories please see Important Directories and Files).

Note that it is not necessary to back up your Working Directory (located inside your Bamboo Home directory).

Step 4. Re-install Bamboo

Please note:

- When specifying the {BAMBOO_HOME} directory, use the same Bamboo Home directory as in your old installation. That is, specify the same directory and path as the Bamboo Home directory shown in ‘Step 1’ (above).
- Make sure that your {BAMBOO_INSTALL} directory is either a new directory, or else delete your old {BAMBOO_INSTALL} directory before you begin, as legacy files may cause problems.
- The {BAMBOO_HOME} directory must be different from the {BAMBOO_INSTALL} directory. This will ensure that your data is not lost when upgrading or re-installing Bamboo.

If you are using external database

Bamboo doesn’t ship with JDBC drivers for external database. If you are using an external database, remember to copy the JDBC driver from your previous Bamboo installation to your new Bamboo install.

Follow steps 1 and 2 of the installation instructions for your operating system:

- Bamboo Standalone Installation Guide — Windows
- Bamboo Standalone Installation Guide — Linux
- Bamboo Standalone Installation Guide — Mac
- Bamboo EAR-WAR Installation Guide

Step 5. Update plugins

If you are using any plugins other than the ones that ship with Bamboo, check that each one is compatible with the new version of Bamboo. Upgrade any plugins that are out-of-date, and disable any plugins that are incompatible with your new version of Bamboo.

Step 6. Re-configure external user repositories (if applicable)

- LDAP integration — If you had previously integrated Bamboo with LDAP/AD, copy your old ...
  .../<Bamboo-install>/webapps/WEB-INF/classes/atlassian-user.xml file to the new Bamboo installation.
- Crowd integration — If you had previously integrated Bamboo with Crowd, you will need to re-enable Crowd integration. For details please see integrating Crowd with Bamboo.

Step 7. Start Bamboo

Once you have installed Bamboo and set the ‘bamboo.home’ property (as described in the Installation Guides), start Bamboo. The upgrade process will be performed when Bamboo starts up. You will not see the Setup Wizard.

Monitor the atlassian-bamboo.log to ensure that the upgrade process has completed successfully.
Step 8. Re-index Bamboo (if indicated in release notes)

Bamboo maintains an index of its build results. This allows Bamboo to display aggregate build results information across builds. You may need to perform a re-index of Bamboo if the upgrade process requires it. This step may or may not be required (depending on the upgrade versions). Also note that you only need to do this if you have existing data in Bamboo.

To re-index, go to 'Administration', then 'Indexing', and click the 'Reindex' button.

Depending on the number of builds and tests you may have, the indexing process may take a significant amount of time. During this period, Bamboo will not be available. Also, it is advisable to ensure that all you have disabled all build queues (or all agents, if you are upgrading from Bamboo 2.0 or later), and that no builds are in progress when you start the re-indexing process. If you have a large instance, it is recommended that you reindex overnight.

Troubleshooting

If you have any problems during upgrade, please raise a support request at https://support.atlassian.com/ and attach your atlassian-bamboo.log so we can help you find out what's gone wrong.

Bamboo Security Advisories

As a distributed application, Bamboo's application-level security is important. This document contains links to version-specific security advisories and related documents for the Bamboo application.

This document is intended to provide information to system administrators about the security of the Bamboo application. It does not address Bamboo'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 Bamboo 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

Open an issue on http://jira.atlassian.com in the Bamboo 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 Bamboo Security Advisories

When a security issue in Bamboo 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 bamboo-users and bamboo-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:

- Bamboo Security Advisory 2009-03-09
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 Bamboo version as soon as possible
- issue a patch to the current stable version of Bamboo
- issue patches for older versions of Bamboo if feasible

Patches will generally be attached to the relevant JIRA issue.

Security Advisories
Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta)

In this advisory:

- Bamboo 2.0 Beta Security Considerations
- Risk Assessment
- Vulnerability
- Fix

Bamboo 2.0 Beta Security Considerations

Risk Assessment

The Bamboo 2.0 Beta does not include the security features that will be present in the final released product. Please note the following security implications when enabling Bamboo's remote agent functionality:

- No encryption of data passed between server and agent — this includes data such as:
  - login credentials for version control repositories
  - build logs
  - build artifacts

- No authentication of the agent or server — this could result in unauthorised actions being taken on your system, such as:
  - Unauthorised parties installing new remote agents — version control repository login credentials could be stolen.
  - Unauthorised parties masquerading as a Bamboo server — the unauthorised server could pass malicious code to the agent to run.

We strongly recommend that you do not enable remote agent installation on any Bamboo instance accessible from a public or untrusted network. Creating remote agents is disabled by default. These are limitations of the beta release only and will be addressed before the final released product.

Vulnerability

An unauthorised party could steal sensitive data passing between the Bamboo server and agents or run malicious code on your agents, as described in the 'Risk Assessment' section.

Fix

These are limitations of the beta release only and will be addressed before the final released product.

Securing your Remote Agents

We strongly recommend that you do not enable remote agent installation without securing them on any Bamboo instance accessible from a public or untrusted network. Creating remote agents is disabled by default. If you choose to enable your remote agents without securing them, please read this Security Advisory to understand the security implications.

You can secure your remote agents by configuring them to use SSL (Secure Sockets Layer). This protocol provides a secure mechanism for communication between your Bamboo server and remote agents. The information below describes how to configure your remote agents to use SSL.

Please note that you cannot set up client certificates in Bamboo due to limitations with ActiveMQ. Thus, while encryption works both ways between the server and client in this recommended configuration, authentication is only one-way. That is, the clients (i.e. agents) can authenticate the server, but the server will not be able to authenticate the clients (i.e. agents).

On this page:

- Step 1. Create keys, stores and certificates
- Step 2. Tell your Bamboo server and agents where to find the stores
- Step 3. Configure your Bamboo server to use SSL

Step 1. Create keys, stores and certificates

The first step in configuring your remote agents to use SSL is to create the required keys, stores and certificates. These artefacts are created using a keytool, as described below:
SSL relies on keys being set up on your server and clients (i.e. agents). To securely store these keys, keystores (databases of keys) need to be created. A certificate is then created by the server (and optionally on the clients, but not for this configuration) to allow publication of the server's key. To establish that the client "trusts" the server, this server certificate is then imported into a truststore (key database file that contains the public keys for a specific server) created on the client.

To create the required keys, stores and certificates for your server and agents,

1. Using a keytool, create a certificate for your server by entering the following command:

   ```bash
   keytool -genkey -alias server -keyalg RSA -keystore server.ks
   ```

2. The server's certificate will be created. Export the certificate, so it can be shared with clients, by entering the following command:

   ```bash
   keytool -export -alias server -keystore server.ks -file server_cert
   ```

3. Each client should now be able to access the server's certificate. Create a keystore for each client, by entering the following command:

   ```bash
   keytool -genkey -alias client -keyalg RSA -keystore client.ks
   ```

4. Create a truststore for each client and import the server's certificate, by entering the command below. This establishes that the client "trusts" the server:

   ```bash
   keytool -import -alias server -keystore client.ts -file server_cert
   ```

Step 2. Tell your Bamboo server and agents where to find the stores

The second step in configuring your agents to use SSL is to instruct your Bamboo server and agents to use the keystores and truststores that you have just created.

To tell your server where to find the keystore,
1. **Add the system properties** `javax.net.ssl.keyStore=/path/to/server.ks` and `javax.net.ssl.keyStorePassword=password` to your VM, by carrying out any of the following three steps:
   
   - *(Standalone or Non-Standalone instances of Bamboo)* Set the `SSL_OPTS` environment variable to hold the `javax.net.ssl.keyStore=/path/to/server.ks` and `javax.net.ssl.keyStorePassword=password` properties.
     
     **E.g.**
     ```
     export SSL_OPTS = -Djavax.net.ssl.keyStore=/path/to/server.ks
     -Djavax.net.ssl.keyStorePassword=password
     ```
     
     Or,
     
   - *(Standalone instances of Bamboo only)* Add `javax.net.ssl.keyStore=/path/to/server.ks` and `javax.net.ssl.keyStorePassword=password` as additional properties to the `wrapper.conf` file.
     
     **E.g.**
     ```
     wrapper.java.additional.4 = -Djavax.net.ssl.keyStore=/path/to/server.ks
     wrapper.java.additional.5 = -Djavax.net.ssl.keyStorePassword=password
     ```
     
     Or,
     
   - *(Non-Standalone instances of Bamboo only)* Make the `javax.net.ssl.keyStore=/path/to/server.ks` and `javax.net.ssl.keyStorePassword=password` properties visible to the VM, as per the instructions for your webserver.

To tell your agents where to find the keystore and truststore,

For each agent,

1. **Tell your agent where to find the keystore and the trust store**, by executing the following command to run the agent,

   ```
   java -jar bamboo-agent-2.0-SNAPSHOT.jar <agentserverURL>
   ```

   including the following command line parameters,

   ```
   -Djavax.net.ssl.keyStore=/path/to/client.ks
   -Djavax.net.ssl.keyStorePassword=password
   -Djavax.net.ssl.trustStore=/path/to/client.ts
   ```

   where `<agentserverURL>` is the URL of the agent's server, e.g.

   ```
   http://192.168.3.235:8085/agentServer/
   ```

   For example,

   ```
   java -Djavax.net.ssl.keyStore=/path/to/client.ks -Djavax.net.ssl.keyStorePassword=password
   ```

**Step 3. Configure your Bamboo server to use SSL**

Once the server and agents know where to find the keystores and truststores, the final step is to instruct your Bamboo server to start using SSL so that agents will be able to authenticate the server.

**To configure your Bamboo server to use SSL,**

...
If you are setting up Bamboo for the first time,

1. Launch the **Bamboo Setup Wizard** and change the protocol of the ‘**Broker URL**’ to ‘SSL’.
   i.e. `ssl://host:port/`

Or, if you are configuring an existing installation of Bamboo,

1. Shut down your Bamboo server and agents.
2. Change the protocol of your ‘**Broker URL**’ in the **bamboo.cfg.xml** file to ‘SSL’. Note, do not change the address of this URL.
   e.g. `<property name="bamboo.jms.broker.uri">ssl://myhost:myport?wireFormat.maxInactivityDuration=0</property>`
3. Start up the Bamboo server.
4. Start up the Bamboo agents. If your agents do not start up, please check that you have set up your certificates correctly.

---

**Bamboo Security Advisory 2009-03-09**

In this advisory:

- **Security vulnerabilities**
  - XSS vulnerabilities on the User Profile page
  - XSS vulnerabilities when adding Requirements for a Build
  - XSS vulnerabilities in the user's full name
  - XSS vulnerabilities in build logs

---

**Security vulnerabilities**

**XSS vulnerabilities on the User Profile page**

**Severity**

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the **Bamboo 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 Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in Bamboo’s ‘**User Profile**’ page. This potentially allows a malicious user (hacker) to hack the URL of controls on the page (e.g. User Profile link) to insert special JavaScript. A hacker could present the hacked URL to users (e.g. disguised in an email). If any users clicked 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 the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to **Bamboo 2.2** to fix the vulnerabilities described below.

You can read more about XSS attacks at [cngisecurity](http://example.com), [CERT](http://example.com) 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 Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

**Vulnerability**

The User Profile page in Bamboo is affected. The URLs of links on this page are not HTML-escaped.

**Fix**

The fix is to HTML-encode the URLs of all links on the User Profile page, so that it cannot be used to run special scripts.
This issue has been fixed in Bamboo 2.2 only. There are no patches available for previous versions of Bamboo, for this fix.

---

**XSS vulnerabilities when adding Requirements for a Build**

**Severity**

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the Bamboo 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 Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability when adding requirements for a build. This potentially allows a malicious user (hacker) to insert special JavaScript in the key of a requirement when adding it to a build. If any users clicked the requirement, 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 the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to Bamboo 2.2 to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](http://example.com), [CERT](http://example.com) 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 Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

**Vulnerability**

The requirements for a build are affected. The key is not HTML-escaped. This affects all versions from 2.0 onwards.

**Fix**

The fix is to HTML-encode the keys of requirements for builds, so that they cannot be used to run special scripts.

This issue has been fixed in Bamboo 2.2 only. There are no patches available for previous versions of Bamboo, for this fix.

---

**XSS vulnerabilities in the user's full name**

**Severity**

Atlassian rates this vulnerability as **HIGH**, according to the scale published in the Bamboo 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 Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in the user's full name. This potentially allows a malicious user (hacker) to create a new user and hack the user's full name to insert special JavaScript. The user's full name is presented in a number of places, including author statistics page, build result comments, build changes and commit notifications. If any users clicked the user name, 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 the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to Bamboo 2.2 to fix the vulnerabilities described below.

You can read more about XSS attacks at [cgisecurity](http://example.com), [CERT](http://example.com) 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 Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The author statistics page, build result comments, build changes and commit notifications are affected. The user name is not HTML-escaped.

Fix

The fix is to HTML-encode the user's full name on these pages/notifications, so that it cannot be used to run special scripts.

This issue has been fixed in Bamboo 2.2 only. There are no patches available for previous versions of Bamboo, for this fix.

XSS vulnerabilities in build logs

Severity

Atlassian rates this vulnerability as HIGH, according to the scale published in the Bamboo 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 Bamboo instances in a public environment. This flaw is an XSS (cross-site scripting) vulnerability in the Bamboo build logs. This potentially allows a malicious user (hacker) to insert special JavaScript into a build log. If a user opened the hacked build log, 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 the User Profile page. This is potentially damaging to your company's reputation.

Atlassian recommends that you upgrade to Bamboo 2.2 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 Bamboo system until you have applied the necessary patch or upgrade. For even tighter control, you could restrict Bamboo access to trusted groups only.

Vulnerability

The Bamboo build logs are affected. The log lines are not HTML-escaped.

Fix

The fix is to HTML-encode the log entries for the build logs, so that they cannot be used to run special scripts.

This issue has been fixed in Bamboo 2.2 only. There are no patches available for previous versions of Bamboo, for this fix.

Please let us know what you think of the format of this security advisory and the information we have provided.

Bamboo User's Guide
About

Bamboo is a **continuous integration** (CI) server. Bamboo assists software development teams by providing:

- automated building and testing of software source-code status.
- updates on successful/failed builds.
- reporting tools for statistical analysis.

The **Bamboo User's Guide** provides information about using Bamboo. If you need information about installing Bamboo or configuring builds, please visit Bamboo Documentation Home.

If you are using Bamboo for the first time, the **Bamboo 101 guide** has some helpful tips.

If you have a question about using Bamboo that hasn't been answered here, please [let us know](#).

Download

You can [download the Bamboo documentation](#) in PDF, HTML or XML formats.

Search the User's Guide

Getting Help

Support | Feature Request | Forums | Knowledge Base

---

**Manage your Bamboo builds inside your IDE**

Use the **Atlassian Connector for Eclipse** or the **Atlassian Connector for IntelliJ IDEA** to work with your Bamboo builds right there in your development environment. Do you use [JIRA](#), [Crucible](#) or [FishEye](#) too? With the connector you can manage your issues 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.

---

**Table of Contents**

**Getting Started**

- Using the Bamboo Dashboard
- Viewing Bamboo's Current Activity
- Viewing your Latest Build Results
- Working with Favourites
  - Adding a Plan to your Favourites
  - Removing a Plan from your Favourites
- Displaying a Build Monitor
- Viewing Bamboo's Agents

**Working with Projects and Plans**

- Viewing a Plan's Activity Log
- About Projects and Plans
- Viewing a Plan's Details
- Viewing the JIRA Issues linked to the Builds in a Build Plan
- Viewing the Clover Coverage Summary for a Build Plan

**Working with Build Results**

- About Builds and Build Results
- Viewing a Build Result
• Viewing the Code Changes that triggered a Build
• Viewing a Build's Artifacts
• Viewing a Build Log
• Viewing the Metadata for a Build Result
• Viewing the Clover Code-Coverage for a Build Result
• Viewing the JIRA Issues for a Build Result
• Linking JIRA Issues to a Build

Working with Tests
• Viewing Test Results for a Build
• Viewing a Test's History
• Viewing Test Statistics for a Plan

Reporting on Plan Trends
• Viewing Build Statistics for a Plan
• Generating Reports across multiple Plans
  • 'Build Activity' Report
  • 'Build Duration' Report
  • 'Build Queue Duration' Report
  • 'Clover Code Coverage' Report
  • 'Clover Lines of Code' Report
  • 'Number of Build Failures' Report
  • 'Number of Tests' Report
  • 'Percentage of Successful Builds' Report
  • 'Time to Fix' Report

Reporting on Author Trends
• Viewing Build Statistics for all Authors
• Viewing Build Results for an Author
• Generating Reports on selected Authors
  • 'Build Activity per Author' Report
  • 'Number of Build Failures per Author' Report
  • 'Number of Builds Broken per Author' Report
  • 'Number of Builds Fixed per Author' Report
  • 'Percentage of Successful Builds per Author' Report

Working with Comments
• About Comments
• Commenting about a Build Result
• Viewing Comments about a Build Result
• Viewing Code Check-in Comments

Working with Labels
• About Labels
• Labelling a Build Result
• Removing a Label from a Build Result
• Viewing Labelled Build Results
• Viewing Popular Labels

Subscribing to RSS Feeds
• Subscribing to an RSS Feed for All Build Results for All Plans
• Subscribing to an RSS Feed for All Build Results for a Particular Plan
• Subscribing to an RSS Feed for Failed Builds for All Plans
• Subscribing to an RSS Feed for Failed Builds for a Particular Plan
• Subscribing to an RSS Feed for Labelled Build Results

Working with Instant Messenger (IM) Notifications
• About Instant Messenger (IM) Notifications
• Labelling a Build Result via IM
• Commenting about a Build Result via IM

Managing your User Profile
• Changing your Personal Details
• Viewing your Notifications
• Changing your Password
• Changing your Notification Preferences
• Associating your Author Name with your User Profile

Getting Started

Getting Started

• Displaying a Build Monitor
• Using the Bamboo Dashboard
• Viewing Bamboo’s Agents
• Viewing Bamboo’s Current Activity
• Viewing your Latest Build Results
• Working with Favourites
  • Adding a Plan to your Favourites
  • Removing a Plan from your Favourites

Using the Bamboo Dashboard

The Dashboard is your Bamboo ‘home’ page. The Dashboard contains three tabs:

• ‘All Plans’ — a list of build plans and each plan’s latest build result.
• ‘Current Activity’ — Bamboo’s agents and build queue, showing which plans Bamboo is currently building and which plans are waiting to be built.
• ‘My Bamboo’¹ — a convenient summary of information that is relevant to you:
  • plans which you have nominated as your favourites.
  • your latest build results (i.e. builds that were triggered by your latest code changes).
  • a summary of your build statistics².

Screenshot: Bamboo Dashboard—‘All Plans’ tab

Each blue name (e.g. ‘Crowd’) represents a project. The name(s) to the right of each project name are the plan(s) (e.g. ‘Crowd 1.1 - Build Plugin’) belonging to that project, and the build number (e.g. ‘CWD-BUILDPLUGIN-11’) represents the latest build result for the plan, while the icon indicates the plan’s current status:

- This plan’s latest build was successful.
- This plan’s latest build failed.
Bamboo is currently checking-out the source-code for this plan, in preparation for starting a build.

Bamboo is currently executing a build for this plan.

This plan has been disabled.

You can:

- click the **plan name** (e.g. 'Crowd 1.1 - Build Plugin') to view the plan details.
- click the **build number** (e.g. 'CWD-BUILDPLUGIN-11') to view the build result.
- click the **author’s name** to view the author’s details (the author is the person who **triggered** the build by checking-in code).

---

**Handy Hint**

You can return to the Dashboard from anywhere in Bamboo by clicking the **Home** link in the top navigation bar.

---

¹ only if you have logged in to Bamboo.

² only if your Bamboo User Profile has been associated with your Author Name.

### Viewing Bamboo’s Current Activity

Sometimes you may want to see which plans are currently being built, and which plans (if any) are waiting in the build queue. The Bamboo **build queue** controls the sequence of **builds**. When a plan submits a build to the build queue, the build will wait in the build queue until a suitable **agent** is available to run the build.

**To view Bamboo’s current activity,**

1. Click the **Home** link in the top navigation bar. This will display the **Dashboard**.
2. Click the **Current Activity** tab. This will display Bamboo’s **Agents** and **Build Queue**, as well as a list of **Recently Completed Builds**.

---

**Screenshot: Bamboo Dashboard ‘Current Activity’ tab**

The above screenshot shows a Bamboo system that has three **Agents**: ‘Local Agent 1’, ‘Local Agent 2’ and ‘Remote Agent on

---
sapporo.sydney.atlassian.com’. 'Local Agent 1' is currently building a plan called ‘Studio Functional Tests - Studio Functional Tests’. 'Local Agent 2' is currently building a plan called ‘Bamboo - Acceptance Tests’. 'Remote Agent on sapporo.sydney.atlassian.com' is currently not building any plans.

You can:

- click a plan name (e.g. ‘Studio Functional Tests’) to view the plan details.
- click the ⚙ icon to view the plan's continuous scrolling activity log.

Additionally, in the ‘Recently Completed Builds’ section, you can:

- click a build number (e.g. ‘SUT-SNF-40) to view the build result.
- click a ‘Reason’ (e.g. 'Updated by...') to view the code changes that triggered the build.

Viewing your Latest Build Results

To view your latest build results,

1. Click the 'Home' link in the top navigation bar. This will display the Dashboard.
2. Click the 'My Bamboo' tab.
3. Your 10 latest build results (that is, builds that were triggered when you checked-in code) are listed in the 'My Latest Changes' section.

Handy Hint
Click any build number (e.g. 'BAM-MAIN-1846') to view the build result.

Screenshot: 'My Bamboo--My Latest Changes'
If your Bamboo User Profile has not yet been associated with your Author Name, there will be no 'My Latest Changes' section.

Working with Favourites

Whereas the 'All Plans' tab on the Bamboo Dashboard lists every plan that exists in your Bamboo system, the 'My Bamboo' tab lists just your chosen favourites — that is, the plans you work with the most. You can easily add and remove plans from your favourites.

When you add a plan to your favourites, you become a 'watcher' of the plan. This means that you may receive notifications about the build results for some or all of your favourite plans, depending on how your administrator has configured each plan's notifications. You can choose whether you would like to receive your notifications by email and/or Instant Messaging (IM).

To view your favourite plans,

1. Click the 'Home' link in the top navigation bar. This will display the Dashboard.
2. Click the 'My Bamboo' tab.
3. Your favourite plans are listed in the 'My Favourite Plans' section. A yellow star is shown next to each.
Adding a Plan to your Favourites

To add a plan to your favourites,

1. Click the 'Home' link in the top navigation bar. This will display the Dashboard.
2. Click the 'All Plans' tab.
3. This will display a list of all plans in your Bamboo system. (Note: Plans that have already been added to your favourites are indicated by a yellow star icon. Plans that have not been added to your favourites are indicated by a grey star icon.)
4. Locate the plan and click the grey star icon.
5. Click the 'My Bamboo' tab.
6. Verify that the plan is now listed in the 'My Favourite Plans' section.

You are now a 'watcher' of the plan. This means that you may receive notifications about the build results for this plan, depending on how your administrator has configured the plan's notifications. You can choose whether you would like to receive your notifications by email and/or Instant Messaging (IM).

Handy Hint
If your administrator has enabled 'Auto-Favourites', each plan will be automatically added to your favourites the first time you check-in code for that plan.

Removing a Plan from your Favourites

To remove a plan from your favourites,
1. Click the 'Home' link in the top navigation bar. This will display the Dashboard.
2. Click the 'All Plans' tab.
3. This will display a list of all plans in your Bamboo system. (Note: Plans that have been added to your favourites are indicated by a yellow star icon. Plans that have not been added to your favourites are indicated by a grey star icon.)
4. Locate the plan and click the yellow star icon: ⭐
5. Click the 'My Bamboo' tab.
6. Verify that the plan is not listed in the 'My Favourite Plans' section.

---

**Displaying a Build Monitor**

Sometimes a development team can benefit from setting up a monitor to display Bamboo's latest build results, e.g.:

You can choose to display Bamboo's latest results for your **favourite plans** only, or for all plans that you have permission to see.

**To display Bamboo's latest build results, for all plans,**

1. Log into Bamboo. (Note: if your Bamboo administrator has allowed **anonymous access**, this step is optional.)
2. Type the following URL into your browser, but replace 'bambooserver' with the real name of your Bamboo server:

   ![Build Monitor Example](image)

   `http://bambooserver:8080/bamboo/telemetry.action`

**To display Bamboo's latest build results, for your favourite plans only,**

1. Log into Bamboo. (Note: only logged-in users can have **favourites**.)
2. Type the following URL into your browser, but replace 'bambooserver' with the real name of your Bamboo server:


---

**Hint**

If you are going to display the build monitor permanently, you may want to ask your Bamboo administrator to **create a user** who has only a limited set of **permissions**.
Viewing Bamboo's Agents

An agent is a service that runs Bamboo builds. There are two types of agents:

- **local agents** run on the Bamboo server.
- **remote agents** run on computers other than the Bamboo server. An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2).

(Note: Local agents run in the server’s process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.)

To view only agents which are currently active, see Viewing Bamboo’s Current Activity

To view all of Bamboo’s agents,

1. Click the ‘Home’ link in the top navigation bar. This will display the Dashboard.
2. Navigate to the ‘Current Activity’ tab.
3. Click the ‘Agents’ link in the ‘Agents’ panel.

A list of all agents in your Bamboo system will display (see screenshot below).

To view a specific agent,

1. Click the ‘Home’ link in the top navigation bar. This will display the Dashboard.
2. Click the name of the agent you wish to view, in the 'Agents' panel of the 'Current Activity' tab.

Agents

Local Agent 1

Crowd - Crowd Trunk - Main
Build

Building for 1 minute.
Approximately 3 minutes remaining

3. The details of the selected agent will display (see screenshot below).

Agents > Local Agent 1 (local)

Information

Current status: Idle

Recent Builds

Executable Plans

4. Click on the 'Executable Plans' tab to view the plans that this agent is capable of building (see screenshot below).

Executable Plans

This agent meets the requirements to execute builds for the following plans:

1. Bamboo - Acceptance Tests
2. Bamboo - Distribution
3. Bamboo - HEAD
4. Bamboo - HEAD Coverage
5. Bamboo - HEAD nightly
6. Bamboo - UCAP Integration Tests
7. Bamboo - Remote Agent Functional Tests
8. Bamboo - Stabilization Tests
9. Bamboo - Stable Branch
10. Bamboo 2.7 on Win64 - Branch
5. Click on the ‘Capabilities’ tab to view the capabilities of this agent.

**Screenshot: View Agent - capabilities**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bamboo functionalTest</td>
<td>true</td>
</tr>
<tr>
<td>bamboo pricey</td>
<td>true</td>
</tr>
<tr>
<td>cloud functionalTest</td>
<td>true</td>
</tr>
<tr>
<td>local</td>
<td>true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (Ant)</td>
<td>/opt/java/tools/ant</td>
</tr>
<tr>
<td>Bash (Command)</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>MAVEN_HOME (Maven)</td>
<td>/opt/java/tools/maven</td>
</tr>
<tr>
<td>Maven 1 (Maven)</td>
<td>/opt/java/tools/maven</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.0)</td>
<td>/opt/java/tools/maven2</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.1)</td>
<td>/opt/java/tools/maven2</td>
</tr>
<tr>
<td>Script (Script)</td>
<td>/opt/java/tools/script</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Java Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.0_07</td>
<td>usr/java/jdk1.5.0_07</td>
</tr>
<tr>
<td>JDL</td>
<td>usr/java/jdk1.5.0_07</td>
</tr>
<tr>
<td>JDL 1</td>
<td>usr/java/jdk1.5.0_07</td>
</tr>
</tbody>
</table>

**Working with Projects and Plans**

- About Projects and Plans
- Viewing a Plan's Activity Log
- Viewing a Plan's Details
- Viewing the Clover Coverage Summary for a Build Plan
- Viewing the JIRA Issues linked to the Builds in a Build Plan

---

**Viewing a Plan’s Activity Log**

Every plan has an activity log. An activity log is a temporary display of the latest output from the plan's most recent build log. To view a plan's activity log,
1. From the Dashboard, locate and click a plan name from the list; OR: From within a build result, click the plan name at the top left of the screen.
2. Click the 'Activity Log' tab.

The plan's current activity will be displayed. For example, the following screenshot shows a plan for which a build is currently in progress:

**Screenshot: Plan Activity**

**Project:** Confluence  **Plan:** Unit Tests

**Confluence - Unit Tests: Live Activity Logs**

<table>
<thead>
<tr>
<th>Summary</th>
<th>Activity</th>
<th>Completed Builds</th>
<th>Tests</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Number:</td>
<td></td>
<td>1075 (currently building)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Started:</td>
<td>02/09/2007 00:07 PM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completed:</td>
<td>52%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time remaining:</td>
<td>Approximately 4 minutes remaining</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Changes**

This build was the result of the following change:

- Joe Smith

**Build Activity per Plan**

*Build activity* is the number of builds that occur in a given period of time.

**About Projects and Plans**

A Bamboo plan (or build plan) is the "recipe" for a build.
A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project.

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans.

Each project has a Name (e.g. "CRM System") and a Key (e.g. "CRM"). The Project Key is prefixed to the relevant Plan Keys, e.g. the "CRM" project could have plans "CRM-TRUNK" and "CRM-BRANCH".

Every Bamboo plan is listed on the Dashboard, from where you can:

- Click on a Plan Name to view the plan details
- Click on a Build Number to view the plan’s latest build result

Projects and plans can only be configured by Bamboo administrators. Please see the Bamboo Administrator’s Guide for details.

Viewing a Plan’s Details

To view a plan’s details,

- From the Dashboard, locate and click a Plan Name from the list;
- OR:
- From within a build result, click the Plan Name at the top left of the screen.

The Plan Summary will be displayed as follows:

Screenshot: ‘Plan Summary’
In the above screenshot:

- The latest build (BUCKET MAIN 74) was successful.
- It was completed 2 days ago and took 30 seconds.
- There is a chart showing build timing and number of failures per build.
- A graph displays the percentage of successful builds and average duration per time period.
- A note indicates that charts are drawn from the results of the last 20 builds.

Recent Failures:

- Average time to fix a failure: 62 minutes
- Average number of builds between fixes: 1 build
- The longest time to fix a failure is 62 minutes, from failure starting in build 74.
- The greatest number of builds taken to fix a failure is 1, from failure starting in build 73.

<table>
<thead>
<tr>
<th>Failed Build</th>
<th>Fixed In</th>
<th>Time Taken To Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>72 (Updated by testers)</td>
<td>Jan 21, 2007 04:20 PM - 2 days ago</td>
<td>42 minutes</td>
</tr>
<tr>
<td>74 (Updated by testers)</td>
<td>Jan 21, 2007 10:33 AM - 2 days ago</td>
<td>42 minutes</td>
</tr>
</tbody>
</table>

Overall, the current status of the project is successful with 96% of builds completed successfully on average. The dropdown on the right allows filtering by time period.
The **green box** indicates that this plan's latest build was successful. Note that a **red box** in this position would indicate that the plan's latest build failed, while a **blue box** would indicate that a build is currently in progress.

- Click the build number (i.e. 'BUCKET-MAIN-71') to view the build result.
- Click the 'Updated by' link to view the code changes that triggered the latest build result.
- **Build duration** is the total time taken to execute a [build plan](About Projects and Plans) --- that is, the time taken to compile the code and run all of the plan's tests.

The large **%' box indicates the success rate of this plan's recent builds. This percentage is calculated on the last 25 builds, or as per your selection via the blue down-arrow: **

- **Shoving Last 25 builds**

Click the **blue down-arrow** to choose how you would like the percentage and graphs on this screen to be calculated. Choose from the following:

- this plan's last 25 builds.
- this plan's builds in the last 7 days.
- this plan's builds in the last 30 days.
- this plan's builds in the last 90 days.
- all of this plan's builds. The percentage and graphs on this screen will all be recalculated automatically when you choose a different option.
The ‘Summary’ tab provides a quick snapshot of the current status of the plan. For more details:

- Click the 'Activity' tab to view the plan's current activity.
- Click the 'Completed Builds' tab to view a list of build results for this plan's recent builds (i.e. the last 25 builds, or as per your selection via the blue down-arrow).
- Click the 'Tests' tab to view a summary of the test results for this plan's recent builds (i.e. the last 25 builds, or as per your selection via the blue down-arrow).
- Click the 'Files' tab to view a list of all the files currently contained in this plan's source-code repository. (Note: The 'Files' tab will only be visible if the plan's latest build ran on a local agent, since the files will not be available if the build ran on a remote agent.)

Plans can only be configured by an administrator. For details please see the Bamboo Administrator's Guide.

Viewing the JIRA Issues linked to the Builds in a Build Plan

If your organisation uses the Atlassian's JIRA and your administrator has integrated Bamboo with JIRA, you will be able to view the JIRA issues that have been linked to the builds in your build plan. This provides an easy way to jump to relevant issue(s) to see details about what the code is intended to achieve.

You may also wish to read about these related topics:

- Viewing the JIRA Issues for a Build Result
- Linking JIRA Issues to a Build

**Viewing the JIRA Issues linked to Builds in a Build Plan**

To view the JIRA issues linked to builds in a build plan:

1. Go to the plan in Bamboo.
2. Click the 'Issues' tab. A list of all of the issues linked to builds (automatically and manually) related to the build plan will display, sorted by build date. You can constrain the list by the build filter (e.g. 'Showing last 25 builds') next to the tabs.

   - Click the issue key to view the issue in JIRA.
   - Click the 'N related builds' link (where N is the number of builds related to the issue) to view the builds related to the issue in JIRA.

Screenshot: JIRA Issues linked to Builds in a Build Plan
Viewing the Clover Coverage Summary for a Build Plan

If your organisation uses the Atlassian's Clover and your build's plan specifies a Clover directory (for details please refer to the Enabling the Clover Plugin), you will be able to view the Clover coverage summary for the build plan.

You may also wish to read about these related topics:

- Enabling the Clover Plugin
- Viewing the Clover Code-Coverage for a Build Result

Viewing the Clover Coverage Summary for a Build Plan

To view the JIRA issues linked to builds in a build plan:

1. Go to the plan in Bamboo.
2. Click the 'Clover' tab. Graphs of the coverage history and lines of code history will display.

Screenshot: ver Coverage Summary for a Build Plan
About Builds and Build Results

A build is one execution of a plan.

Every build has a Build Number, which is appended to the relevant Plan Key to form the Build Key. For example, if a plan with the key "CRM-BRANCH" is executed for the seventeenth time, the build key will be "CRM-BRANCH-17".

Every completed build has a build result:
• 'Successful' — the code compiled, with or without errors, and all tests completed successfully.
• 'Failed' — either the code did not compile, or at least one test failed.

Additionally,

• if the build result is 'Failed', and the previous build result was 'Successful', the build is said to be 'Broken'.
• if the build result is 'Successful', and the previous build result was 'Failed', the build is said to be 'Fixed'.

The latest build result for every plan is listed on the Dashboard. Bamboo can also send notifications and generate RSS feeds about build results.

**Viewing a Build Result**

**To view a plan's most recent build result:**

1. Go to the Dashboard.
2. Locate the plan in the list, then click the Build Number.

**To view all build results for a plan:**

1. Go to the relevant plan.
2. Click the 'Completed Builds' tab to see a summary list of build results.
3. To view the details for a particular build result, click the Build Number in the list.

A build result looks like this:

**Screenshot: 'Build Result Summary’**

- **Build BONNIE-MAIN-21**
  - Summary
  - Tests
  - Changes
  - Artifacts
  - Logs
  - Comments
  - Metadata

- Code Changes
  - This build was the result of the following change:

- Tests
  - 75 tests in total

- Build Errors
  - The build generated some errors. See the full build log for details.

In the above screenshot:

- You can click the plan name ('Main Build' in this example) to see the plan details for this build:

  **Project: Bonnie**  **Plan: Main Build**  **Build: 21**

- You can use the build results navigator to scroll through other build results for this plan:
• The 'Summary' tab shows a snapshot of the build result. To see more detail:
  • Click the 'Tests' tab to view the build's test results.
  • Click the 'Changes' tab to view the code changes that triggered this build (if applicable).
  • Click the 'Artifacts' tab to view any artifacts relating to this build.
  • Click the 'Logs' tab to view a complete build log.
  • Click the 'Comments' tab to view a trail of comments regarding this build result. You can also click the following icon to add a comment:
  • Click the 'Metadata' tab to view any metadata that relates to this build result.

• Depending on how your Bamboo administrator has configured the system, the following additional tabs may be available:
  • Click the 'Clover' tab to view the Clover code-coverage that relate to this build result (if applicable).
  • Click the 'JIRA' tab to view any JIRA issues that relate to this build result (if applicable).

Viewing the Code Changes that triggered a Build

If a build was triggered by a code change, the code changes will be shown in the build result.

To view the code changes that triggered a particular build result,

1. Go to the build result.
2. Click the 'Changes' tab. A list of updated files will be shown.
   - If there are no code changes since last build (e.g. a manual build) there will not be any changes displayed on screen.
3. Click the filename to view the changes; or, click the version number to view the entire file; or, click the 'diffs' links to view the differences between the current and previous version of each file.
   - Links to individual source-code files will only be available if your Bamboo administrator has specified a 'Web Repository URL' in the build's plan. For details please see Specifying a Plan's Source Repository in the Bamboo Administrator's Guide.

Tip: Bamboo-FishEye Integration — If you are using Atlassian's FishEye with Bamboo, you will be able to view the source code, version history and 'diff' comparison for the file in FishEye via the relevant links described above. (Your Bamboo administrator needs to have specified the URL of your FishEye project when specifying the source repository for the plan).

Screenshot: 'Code Changes'

Build Result BUCKET-MAIN-69

Related Documentation

• About Build Triggering

Viewing a Build's Artifacts

An artifact is something created by a build. There are two types of artifacts:

• User-defined artifacts (e.g. JAR files) are specified in the build's plan by a Bamboo administrator.
• Auto-generated artifacts are created automatically by Bamboo.
To view a build's artifacts,

1. Go to the build result.
2. Click the 'Artifacts' tab.

If you would like to view the latest version of an artifact for a latest build, you can manually edit the URL to retrieve it, as follows:

To view the latest version of an artifact for a build,

1. Copy the URL for the build artifact.
2. Paste the URL for the build artifact in your browser and replace the build number in the URL with '/latest'
   
   For example, if the URL for your artifact is:
   
   http://server/bamboo/browse/MYBUILD-254/artifact/logs/sample-log.log
   
   You would replace '-254' with /latest:
   
   http://server/bamboo/browse/MYBUILD/latest/artifact/logs/sample-log.log
   
   If you need to login to view the artifacts, you can append os_username & os_password parameters to the URL to access the files.

Screenshot: 'Build Artifacts'

Build Result CONF-DIST-29

Summary | Tests | Changes | Artifacts | Logs | Comments | Metadata
--- | --- | --- | --- | --- | --- | ---

Build Artifacts

User-Defined Artifacts

The build has the following artifacts.

- standalone (25 MB)
- war (24 kB)
- source (71 kB)

Remove all artifacts

Auto-Generated Artifacts

The following artifacts have been generated by Bamboo.

- Build Log

Viewing a Build Log

Every build has a build log. A build log is a permanent record of all the output generated by compiling the plugin's source-code and executing the tests.

To view a build log,

1. Go to the build result.
2. Click the 'Logs' tab.

Screenshot: 'Build Log'
Viewing the Metadata for a Build Result

If your source-code repository provides metadata for your build results, Bamboo will display it.

To view the metadata for a build result:

1. Go to the plan.
2. Click the 'Completed Builds' tab, then click the Build Number in the list.
3. This will display the Build Result Summary. Click the 'Metadata' tab.
Viewing the Clover Code-Coverage for a Build Result

If your organisation uses the Atlassian Clover code-coverage tool, Bamboo can record code-coverage details (i.e. the percentage of code covered by tests) for each build result.

This is only available if the build's plan specifies a Clover directory (for details please refer to the Enabling the Clover Plugin). Also note that the Clover analysis will only be recorded for successful builds.

To view Clover code-coverage for a build result:

1. Go to the plan.
2. Click the 'Completed Builds' tab, then click the Build Number in the list.
3. This will display the Build Result Summary. Click the 'Clover' tab.
Bamboo also provides data on code-coverage trends for a plan over a period of time. For details see:


**Viewing the JIRA Issues for a Build Result**

If your organisation uses Atlassian's JIRA and your administrator has integrated Bamboo with JIRA, you will be able to view the JIRA issues that have been linked to the build. These issue links are either created automatically by Bamboo or can be manually added. This provides an easy way to jump to relevant issue(s) to see details about what the code is intended to achieve.

You may also wish to read about these related topics:

- Viewing the JIRA Issues linked to the Builds in a Build Plan
- Linking JIRA Issues to a Build

**Viewing the JIRA Issues for a Build Result**

The JIRA Issues linked to a build can be viewed on the Build Result pages. If you have specified an issue in your build comments, labels or commit messages (note, you must specify the issue key in upper-case), then they will be automatically linked to your build and displayed. In addition, if you have manually linked any issues to your build, then they will also be displayed.

The JIRA Issues linked to a build will be listed on the 'Summary' tab and the 'Issues' tab of the build result. The 'Summary' tab will only show up to two issues, so you will need to view the 'Issues' tab if you wish to see more than two issues.

**To view the JIRA issues for a build result:**
1. Go to the plan in Bamboo.
2. Click the 'Completed Builds' tab, then click the Build Number for the build that you wish to view.
3. This will display the Build Result Summary. The 'JIRA Issues' section will show up to two JIRA issues that are linked to the build. See 'Screenshot: JIRA Issues for a Build Result — Summary tab' below.
4. Click the 'Issues' tab. All of the JIRA issues linked to your build will display grouped by 'Fixed Issues' (i.e. issues fixed by the build) and 'Related Issues' (i.e. issues linked to the build but not fixed by it). Issues will be sorted by issue key within these two groups. See 'Screenshot: JIRA Issues for a Build Result — Issues tab' below.

### Screenshot: JIRA Issues for a Build Result — Summary tab

![JIRA Issues for a Build Result — Summary tab](image1)

### Screenshot: JIRA Issues for a Build Result — Issues tab

![JIRA Issues for a Build Result — Issues tab](image2)
Linking JIRA Issues to a Build

If your organisation uses the Atlassian's JIRA and your administrator has integrated Bamboo with JIRA, you will be able to view the JIRA issues that have been linked to the build. If you have specified an issue in your build comments, labels or commit messages (note, you must specify the issue key in upper-case), then these issue links are either created automatically by Bamboo. You can also manually add new issue links to your build, or edit or remove any issue links.

You may also wish to read about these related topics:

- Viewing the JIRA Issues for a Build Result
- Viewing the JIRA Issues linked to the Builds in a Build Plan

Editing Issues Links for your Build

Issue links that have been created automatically or manually, can be edited or removed from the build.

To edit an existing JIRA Issue link for a build result:

1. Go to the plan in Bamboo.
2. Click the 'Completed Builds' tab, then click the Build Number for the build that you wish to view.
3. This will display the Build Result Summary. Click the 'Issues' tab.
4. All of the JIRA issues linked to your build will display grouped by 'Fixed Issues' and 'Related Issues'. By default, all issue links that are automatically created by Bamboo are created as 'Related Issues'.
   - To change a 'Related Issue' to a 'Fixed Issue', click the up arrow icon.
   - To change a 'Fixed Issue' to a 'Related Issue', click the down arrow icon.
   - To remove an issue link from the build (the issue will not be removed from JIRA), click the rubbish bin icon.

Manually Adding New Issue Links to a Build

If an issue has not been linked automatically to your build, you can manually create a new link from that issue to your build.

To manually add a new JIRA Issue link to a build result:

1. Go to the plan in Bamboo.
2. Click the 'Completed Builds' tab, then click the Build Number for the build that you wish to view.
3. This will display the Build Result Summary. Click the 'Issues' tab.
4. All of the JIRA issues linked to your build will display. Click the 'Add linked issue' link.
5. The 'Add Linked JIRA Issue' screen will display (see screenshot below). Select the 'Type of Issue Link', which can be 'Fixed' or 'Related'.
   - 'Fixed' means that this issue is fixed by this build, e.g. a bug.
   - 'Related' means that this issue is related to this build, but not fixed by it, e.g. a documentation task related to changes from the build.
6. Enter the JIRA issue key of the issue you want to link to this build. Please note, you must specify the issue key in upper-case, e.g. 'JIRA-1234'.
7. Click 'Save' to link the issue to your build. It will now display on the 'Issues' tab.
Working with Tests

- Viewing a Test's History
- Viewing Test Results for a Build
- Viewing Test Statistics for a Plan

Viewing Test Results for a Build

Bamboo provides a convenient summary of all the tests that were run when a particular build was executed — as well as full details of any errors. This is useful when you are investigating what caused a build to fail.

To view the tests for a particular build:

1. Go to the build result.
2. Click the 'Tests' tab.
3. Click the 'Failed Tests' / 'Successful Tests' links (see screenshot below) to view a list of which tests failed or were successful.

To see a particular test's results for other builds, click the test name.

Screenshot: Test Results for a Build
For more meaningful display of test names within Bamboo, the word 'test' is stripped out of test case name names if it occurs at the beginning, and capitals and underscores are treated as word separators.

### Viewing a Test's History

A test's history shows you:

- The occasions when the test has failed. This can be useful when investigating what code changes were related to a failed test (see below).
- The test's average duration (running time), and whether the duration is increasing or decreasing across builds.

To view a test's history,
1. Go to a plan or a build result.
2. Click the 'Tests' tab.
3. Click the name of the test in which you are interested.
4. (Skip this step if you are looking at a plan.) The test's latest result will be displayed. Click the link 'View test case across builds'.
5. The 'Test History' will be displayed as shown below.
To view the code changes that relate to a failed test,

1. Under ‘Recent Failures’, click the relevant build result (‘47’ in the above screenshot).
2. This will display the build result. Click the ‘Changes’ tab to display the code changes.

Viewing Test Statistics for a Plan

Bamboo provides a summary of test results across all of a plan’s builds. This helps you to:

- Troubleshoot by identifying which tests fail most frequently, and which tests take longest to fix.
- Manage your build duration by identifying the plan’s slowest running tests.
- Ensure quality by monitoring the number of tests over time: are your test cases growing with your code base?

To view the test statistics for all of a plan’s builds:

1. Go to the plan.
2. Click the ‘Tests’ tab.
3. The plan’s ‘Top 10 Most Failing Tests’ sub-tab will be displayed. Click the other three sub-tabs to view the plan’s ‘Top 10 Longest to Fix Tests’, ‘Top 10 Longest Running Tests’, ‘Number of Tests’ (see screenshots below).

Screenshot 1: Top 10 Most Failing Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Times Failed</th>
<th>Most Recent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fa q</td>
<td>2</td>
<td>37, 30</td>
</tr>
<tr>
<td>Image</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Link 17</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Escape: escape16</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Ul render camel case off: ut1</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Ul render camel case on: ut1</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Link 16</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Ul render camel case off: ut1</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Tables: table14</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>Link 5</td>
<td>1</td>
<td>47</td>
</tr>
</tbody>
</table>

To view a test’s history, click the test name.
## Screenshot 2: Top 10 Longest to Fix Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Average Time to Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAQ</td>
<td>1 hour, 2 minutes</td>
</tr>
<tr>
<td>Image camel case on: image</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Link: link17</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Escopo: escopo15</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Url render camel case off: url0</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Url render camel case on: url0</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Link: link6</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Url render camel case off: url1</td>
<td>19 minutes</td>
</tr>
<tr>
<td>Link: link0</td>
<td>19 minutes</td>
</tr>
</tbody>
</table>

## Screenshot 3: Top 10 Longest Running Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Average Duration (seconds)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAQ</td>
<td>7.43 (1 second)</td>
</tr>
<tr>
<td>Jira release plan</td>
<td>1.62 (1 second)</td>
</tr>
<tr>
<td>Layout</td>
<td>0.60 (less than 1 second)</td>
</tr>
<tr>
<td>Image after macro</td>
<td>0.62 (less than 1 second)</td>
</tr>
<tr>
<td>Image camel case on: default</td>
<td>0.46 (less than 1 second)</td>
</tr>
<tr>
<td>Image camel case off: default</td>
<td>0.44 (less than 1 second)</td>
</tr>
<tr>
<td>Lists: list21</td>
<td>0.42 (less than 1 second)</td>
</tr>
<tr>
<td>New rv inserted link after title</td>
<td>0.43 (less than 1 second)</td>
</tr>
<tr>
<td>Emoticons next to styled text</td>
<td>0.41 (less than 1 second)</td>
</tr>
<tr>
<td>Insert renders as u</td>
<td>0.41 (less than 1 second)</td>
</tr>
</tbody>
</table>

* Results are based on the last 25 builds only.

## Screenshot 4: Number of Tests
To compare the number of tests over time across multiple plans, view the 'Number of Tests' Report.

**Reporting on Plan Trends**

**Reporting on Trends**

- Generating Reports across multiple Plans
  - 'Build Activity' Report
  - 'Build Duration' Report
  - 'Build Queue Duration' Report
  - 'Clover Code Coverage' Report
  - 'Clover Lines of Code' Report
  - 'Number of Build Failures' Report
  - 'Number of Tests' Report
  - 'Percentage of Successful Builds' Report
  - 'Time to Fix' Report
- Viewing Build Statistics for a Plan

**Viewing Build Statistics for a Plan**

To view a plan's build statistics,

- From the Dashboard, locate and click a Plan Name from the list;
- OR:
  - From within a build result, click the Plan Name at the top left of the screen.

The Plan Summary will be displayed as follows:

*Screenshot: 'Plan Summary'*
In the above screenshot:

- **Latest build BUCKET-MAIN-74 was successful**
  - Reason: Updated by developer
  - Completed 2 days ago
  - Duration: 30 seconds

- **96% Successful Builds: 24/25**
  - Average Duration: 59 seconds

- **Build Timing & Number of Failures per Build**
  - Chart showing build timing and number of failures per build.

- **% Successful Builds & Avg Duration per Time Period**
  - Chart showing percentage of successful builds and average duration per time period.

---

**Recent Failures**

- Average time to fix a failure: 62 minutes
- Average number of builds between first failure and fix: 1 builds
- The longest time taken to fix a failure is 62 minutes, from failure starting in build 72.

<table>
<thead>
<tr>
<th>Failed Build</th>
<th>Fixed In</th>
<th>Time Taken To Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>73(\text{Updated by developer})</td>
<td>Jan 23, 2007 9:40:09 AM - 2 days ago</td>
<td>1 builds</td>
</tr>
<tr>
<td>74(\text{Updated by developer})</td>
<td>Jan 23, 2007 10:33:05 AM - 2 days ago</td>
<td>42 minutes</td>
</tr>
</tbody>
</table>

---

**Note:** Charts are drawn from the results of the last 20 builds. Use the drop down in the right corner to change your filter size.

- **Feed for all builds or just the failed builds**

---

In the above screenshot:
The green box indicates that this plan's latest build was successful. Note that a red box in this position would indicate that the plan's latest build failed, while a blue box would indicate that a build is currently in progress.

- Click the build number (i.e. 'BUCKET-MAIN-71') to view the build result.
- Click the 'Updated by' link to view the code changes that triggered the latest build result.
- Build duration is the total time taken to execute a build plan — that is, the time taken to compile the code and run all of the plan's tests.

The large '%' box indicates the success rate of this plan's recent builds. This percentage is calculated on the last 25 builds, or as per your selection via the blue down-arrow:
- Shoving Last 25 builds

Click the blue down-arrow to choose how you would like the percentage and graphs on this screen to be calculated. Choose from the following:
- this plan's last 25 builds.
- this plan's builds in the last 7 days.
- this plan's builds in the last 30 days.
- this plan's builds in the last 90 days.
- all of this plan's builds. The percentage and graphs on this screen will all be recalculated automatically when you choose a different option.

Latest build BUCKET-MAIN-74 was successful
Reason: Updated by todavies  Completed: 2 days ago  Duration: 33 seconds

Latest build TEST-BETA-3 failed with 14 failing tests
Reason: Manual build  Completed: 6 days ago  Duration: 4 minutes

96% Successful
Successful Runs: 24 / 26
Average Duration: 30 seconds
The ‘Summary’ tab provides a quick snapshot of the current status of the plan. For more details:

- Click the ‘Activity’ tab to view the plan's current activity.
- Click the ‘Completed Builds’ tab to view a list of build results for this plan's recent builds (i.e. the last 25 builds, or as per your selection via the blue down-arrow).
- Click the ‘Tests’ tab to view a summary of the test results for this plan's recent builds (i.e. the last 25 builds, or as per your selection via the blue down-arrow).
- Click the ‘Files’ tab to view a list of all the files currently contained in this plan's source-code repository. (Note: The ‘Files’ tab will only be visible if the plan's latest build ran on a local agent, since the files will not be available if the build ran on a remote agent.)

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Summary of the plan's current status</td>
</tr>
<tr>
<td>Activity</td>
<td>Activity plan's current activity</td>
</tr>
<tr>
<td>Completed Builds</td>
<td>List of build results for this plan's recent builds (last 25 builds)</td>
</tr>
<tr>
<td>Tests</td>
<td>Summary of test results for this plan's recent builds (last 25 builds)</td>
</tr>
<tr>
<td>Files</td>
<td>List of all the files currently contained in this plan's source-code repository</td>
</tr>
</tbody>
</table>

---

### Generating Reports across multiple Plans

Bamboo provides a report generator that enables you to compare build statistics across one or more plans, using a variety of different metrics.

**To report on build statistics per plan,**

1. Click the ‘Reports’ link in the top navigation bar. This will display the ‘Report Parameters’ screen as shown below.
2. **Report** — choose from the available reports. Available reports include:
   - ‘Build Activity’ Report
   - ‘Build Duration’ Report
   - ‘Build Queue Duration’ Report
   - ‘Clover Code Coverage’ Report
   - ‘Clover Lines of Code’ Report
   - ‘Number of Build Failures’ Report
   - ‘Number of Tests’ Report
   - ‘Percentage of Successful Builds’ Report
   - ‘Time to Fix’ Report
3. **Build plans** — choose the plan(s) on which you want to report. You can use the <Ctrl> key to select multiple plans.
4. **Group By** — choose whether your report's horizontal axis should show days, months or weeks. You can also specify ‘Auto’, which varies by report, but will generally default to ‘week’.
5. **Date Filter** — choose from:
   - ‘All builds’
   - ‘Last 7 days’
   - ‘Last 30 days’
   - ‘Last 90 days’
   - ‘Select Range’ — choosing this option will display two boxes in which you will need to specify the ‘from’ and ‘to’ dates (dd/MM/yyyy).
6. Click the ‘Submit’ button to generate your report.

Additional information:

- Your Bamboo administrator may configure custom reports by using plugins. For details please see the Bamboo Administrator's Guide.

---

*Screenshot: ‘Report Parameters—Build Plans’*
'Build Activity' Report

*Build activity* is the number of builds that occur in a given period of time.

You can choose the plan(s) and time period on which you want to report.

*Sample Report: ‘Build Activity per Plan’*
'Build Duration' Report

*Build duration* is the total time taken to execute a build plan — that is, the time taken to compile the code and run all of the plan's tests.

You can choose the plan(s) and time period on which you want to report.

*Sample Report: 'Build Duration per Plan'*
The report shows how long your build takes over time. Is it getting slower or faster?

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Build Queue Duration'
The report shows how long your build is spending in the queue over time.

This report will only be available if your administrator has specified 'Clover output will be produced' in the plan's configuration. For details please see the Bamboo Administrator's Guide.

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Clover Code Coverage per Plan'
Clover Code Coverage

Comparing code coverage gives you an idea of how well the code base is tested. 100% coverage means that all code elements have been covered by your tests.

'Clover Lines of Code' Report

This report will only be available if your administrator has specified 'Clover output will be produced' in the plan's configuration. For details please see the Bamboo Administrator's Guide.

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Clover Lines of Code per Plan'
'Number of Build Failures' Report

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Number of Build Failures per Plan'
Number of Build Failures

How many builds are being broken? A high value indicates a relatively unstable plan that tends to be broken often.

'Sample Report: 'Number of Tests per Plan''

You can choose the plan(s) and time period on which you want to report.
'Percentage of Successful Builds' Report

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Percentage of Successful Builds per Plan'
'Time to Fix' Report

You can choose the plan(s) and time period on which you want to report.

Sample Report: 'Time to Fix per Plan'
Reporting on Author Trends

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user.

- Generating Reports on selected Authors
  - 'Build Activity per Author' Report
  - 'Number of Build Failures per Author' Report
  - 'Number of Builds Broken per Author' Report
  - 'Number of Builds Fixed per Author' Report
  - 'Percentage of Successful Builds per Author' Report
- Viewing Build Results for an Author
- Viewing Build Statistics for all Authors
An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user.

To view a summary of all authors’ statistics,

1. Click the ‘Authors’ link in the top navigation bar.
2. This will display the following screen, where you can click any column-header to sort in ascending order (or click twice to sort in descending order).

Screenshot: ‘Authors Summary-sorted by ‘Fixed’ (descending)’

<table>
<thead>
<tr>
<th>Name</th>
<th>Triggered</th>
<th>Failed</th>
<th>% Failed</th>
<th>Broken</th>
<th>Fixed</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe Bloggs</td>
<td>002</td>
<td>275</td>
<td>28%</td>
<td>124</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>Mary Smith</td>
<td>454</td>
<td>60</td>
<td>21%</td>
<td>47</td>
<td>44</td>
<td>-3</td>
</tr>
<tr>
<td>Tom Brown</td>
<td>401</td>
<td>60</td>
<td>22%</td>
<td>60</td>
<td>40</td>
<td>-10</td>
</tr>
<tr>
<td>Sally Jones</td>
<td>432</td>
<td>102</td>
<td>24%</td>
<td>41</td>
<td>32</td>
<td>-2</td>
</tr>
</tbody>
</table>

Handy Hint
You can click any author’s name to see their recent build results.

Viewing Build Results for an Author

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user.

To view an author’s build results,

1. Click the ‘Authors’ link in the top navigation bar.
2. This will display the ‘Authors Summary’ screen. Click the relevant author’s name.
3. This will display the author’s ‘User Details’ (email address, etc) — see screenshot below. Click through the following tabs to view recent build results:
   - ‘Builds Summary’ — a statistical summary of all the author’s builds.
   - ‘Last 10 Builds’ — a list of the last 10 builds that were triggered by this author.
   - ‘Last 10 Broken’ — a list of the last 10 builds that were triggered by this author, where the build failed and the previous build for the same plan was successful.
   - ‘Last 10 Fixed’ — a list of the last 10 builds that were triggered by this author, where the build was successful and the previous build for the same plan failed.

If your Bamboo User Profile has not yet been associated with your Author Name, there will be no ‘User Details’ tab.

Screenshot 1: ‘User Details’ tab
Screenshot 2: 'Builds Summary' tab

<table>
<thead>
<tr>
<th>Builds Triggered by Author</th>
<th>Broken/Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All builds triggered:</td>
<td></td>
</tr>
<tr>
<td>Failed Builds:</td>
<td></td>
</tr>
<tr>
<td>Successful Builds:</td>
<td></td>
</tr>
</tbody>
</table>

Builds triggered by an author are those builds which contain changes committed by the author.

All builds triggered: 993
Failed Builds: 278 (28%)
Successful Builds: 715 (72%)

Broken/Passed
Broken means the build had failed but the previous build was successful.
Passed means that the build was successful but the previous build had failed.
Broken By Author: 128 (13% of all builds triggered)
Passed By Author: 127 (13% of all builds triggered)
Difference of Fixed and Broken: 4

Screenshot 3: 'Last 10 Builds' tab

<table>
<thead>
<tr>
<th>Build</th>
<th>When</th>
<th>Comments</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-MAIN-1755</td>
<td>6 hours ago</td>
<td>Merged bamboo-hit properties sample to the right location (BAAM-1755)</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1756</td>
<td>1 day ago</td>
<td>Fixed up stupid pointer to the cancel link on editatted player (BAAM-1756)</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1757</td>
<td>2 days ago</td>
<td>Added some UMP stuff for dashboard</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1758</td>
<td>2 days ago</td>
<td>Fixed up wording of error succeeded, not succeeded</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1759</td>
<td>6 days ago</td>
<td>Added TUI demo wrapper... might be worth checking out?</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1760</td>
<td>6 days ago</td>
<td>Updated the default values for the TUI</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1761</td>
<td>6 days ago</td>
<td>Added ajax notify for latest build widget and the legs page</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1762</td>
<td>6 days ago</td>
<td>Revised</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1763</td>
<td>6 days ago</td>
<td>[mvn-realsec-plugin] prepare new development docation</td>
<td>[details]</td>
</tr>
<tr>
<td>BAM-MAIN-1764</td>
<td>6 days ago</td>
<td>[mvn-realsec-plugin] prepare release artifact_bamboo_1.0_beta</td>
<td>[details]</td>
</tr>
</tbody>
</table>

Screenshot 4: 'Last 10 Broken' tab
Generating Reports on selected Authors

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user.

To generate a report on selected authors,
1. Click the ‘Authors’ link in the top navigation bar.
2. Click the ‘Statistics’ tab. This will display the ‘Report Parameters’ screen as shown below.
3. ‘Report’ — choose from the available reports. Available reports include:
   • ‘Build Activity per Author’ Report
   • ‘Number of Build Failures per Author’ Report
   • ‘Number of Builds Broken per Author’ Report
   • ‘Number of Builds Fixed per Author’ Report
   • ‘Percentage of Successful Builds per Author’ Report

   Additionally, your Bamboo administrator may configure custom reports by using plugins. For details please see the Bamboo Administrator's Guide.

4. ‘Authors’ — choose the author(s) on whom you want to report. You can use the <Ctrl>-key to select multiple author.
5. ‘Group By’ — choose whether your report’s horizontal axis should show days, months or weeks. You can also specify ‘Auto’, which varies by report, but will generally default to ‘month’.
6. Click the ‘Submit’ button to generate your report.

Screenshot: ‘Report Parameters--Authors’

Report Parameters

<table>
<thead>
<tr>
<th>Report:</th>
<th>Select...</th>
</tr>
</thead>
</table>
| Authors: | amakovoi
   andeask
   epol
   are
   bkuo
   bmcouy
   bnguyen
   cmiller
   comauftord
   conwen
   dbrown
   defkin
   devuser
   dharaska
   dhust
   dloeng
   dmogahan
   dso
   dswilis
   edmawong |
| Group By: | Auto |

‘Build Activity per Author’ Report

Build activity is the number of builds that occur in a given period of time.

You can select the author(s) on whom you want to report.

Sample Report: ‘Build Activity per Author’
Build Activity

Você pode selecionar o(s) autor(es) a quem você quer relatar.

Sample Report: 'Number of Build Failures per Author'
'Number of Builds Broken per Author' Report

You can select the author(s) on whom you want to report.

Sample Report: 'Number of Builds Broken per Author'
Number of Builds Broken

You can select the author(s) on whom you want to report.

Sample Report: 'Number of Builds Fixed per Author'

Number of Builds Fixed
'Percentage of Successful Builds per Author' Report

You can select the author(s) on whom you want to report.

Sample Report: 'Percentage of Successful Builds per Author'

### Percentage of Successful Builds

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>msmith</td>
<td>bloggs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Working with Comments

#### Working with Comments

- **About Comments**
- **Commenting about a Build Result**
- **Viewing Code Check-in Comments**
- **Viewing Comments about a Build Result**

### About Comments

Comments are a useful way to record and share information about builds. There are two types of comments in Bamboo:

- Comments you make **when you commit code** — these comments are automatically copied into Bamboo from your source-code repository. See **Viewing Code Check-in Comments**.
- Comments you make **about a build result** — these are comments that you make ad-hoc about a particular build result. See **Commenting about a Build Result** and **Viewing Comments about a Build Result**.
Commenting about a Build Result

Bamboo allows you to record comments about a build result. This is a convenient way to record relevant information for future reference, and to collaborate with colleagues.

To comment on a build result,

1. From within the 'Build Result' screen, click the 'Comments' tab. A list of existing comments about this build result will be displayed.
2. Type your comment into the 'Add Comment' box, then click the 'Save' button.

Screenshot: Build Result - Comments

You must login to Bamboo before you can comment on a build result.

Viewing Comments about a Build Result

To view comments about a particular build result,
1. From within the ‘Build Result’ screen, click the ‘Comments’ tab. A list of all comments about this build result will be displayed, including author and timestamp:

**Build Result BUCKET-MAIN-69**

<table>
<thead>
<tr>
<th>Comments</th>
</tr>
</thead>
</table>
| Rosie Jameson (Jan 29, 2007 10:26:19 AM)  
This is another comment. |
| Matt Ryall (Dec 19, 2006 2:52:36 PM)  
This is a reply to Edwin’s comment. |
| Edwin Wong (Dec 19, 2006 4:42:28 PM)  
This is a comment. |

To view comments about all build results for a particular plan,

1. From the Dashboard, click the plan you are interested in.
2. Click the plan’s ‘Completed Builds’ tab.
3. This will display a list of the plan’s build results. The icon indicates that there are one or more comments about a particular build result. Hold your mouse over the icon to see the comment(s), e.g.:

**Atlassian Bucket - Main Build: Completed Build Results**

<table>
<thead>
<tr>
<th>Build Number</th>
<th>Reason</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUCKET-MAIN-74</td>
<td>Updated by tlavies</td>
<td>6 days ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-73</td>
<td>Updated by tlavies</td>
<td>6 days ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-72</td>
<td>Updated by tlavies</td>
<td>0 days ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-71</td>
<td>Updated by Bamboo Master</td>
<td>2 weeks ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-70</td>
<td>Updated by David Loeng</td>
<td>2 weeks ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-69</td>
<td>Updated by tlavies</td>
<td>2 months ago</td>
</tr>
<tr>
<td>BUCKET-MAIN-68</td>
<td>Updated by David Loeng</td>
<td>2 months ago</td>
</tr>
</tbody>
</table>
| BUCKET-MAIN-67 | Rosie Jameson (Jan 29, 2007 10:26:19 AM)  
This is another comment. | 3 months ago |
| BUCKET-MAIN-66 | Matt Ryall (Dec 29, 2006 2:52:36 PM)  
This is a reply to Edwin’s comment. | 3 months ago |
This is a comment. | 3 months ago |
| BUCKET-MAIN-64 | Updated by Charles Miller | 4 months ago |
| BUCKET-MAIN-63 | Updated by Charles Miller | 4 months ago |
Viewing Code Check-in Comments

If a build was triggered by a code change, the commit comment (or check-in comment) will be shown in the build result.

To view the code check-in comments for a particular build result,

1. Go to the build result.
2. The build's commit comment will be shown to the right of the screen, under the heading 'Code Changes'.

A note about build triggering

There are a number of ways in which a build can be triggered for a plan,

- **Build Strategy**:
  - Code updated — a build can be triggered whenever one or more authors checks-in code.
  - Scheduled build — a build can be scheduled to occur at regular intervals.
  - Manual build — a build can be triggered manually.
  - Initial clean build — a build will be triggered when a new plan is created.

- **Build Dependency**:
  - Dependency — a build can be triggered whenever a successful build occurs for another plan.

The way in which each build was triggered is listed in the 'Reason' column on the Dashboard.

Working with Labels

Working with Labels

- About Labels
- Labelling a Build Result
- Removing a Label from a Build Result
- Viewing Labelled Build Results
- Viewing Popular Labels

About Labels

A label is a convenient way to tag and group build results that are logically related to each other. Labels can also be used to define RSS feeds and to control build expiry.

Labels can be applied to build results automatically, by specifying the label(s) in a build plan (note that only Bamboo administrators can do this). Labels can also be applied ad hoc to build results by Bamboo users.

Labelling a Build Result

With Bamboo, you can label your build results in whatever way works best for your team. Labels are not restricted to a particular plan, so you can
apply the same label to build results from different plans.

For example, it might not be practical for your QA team to review every build, and you need to know which builds they have reviewed. By using labels such as "qa_passed" and "qa_failed", Bamboo allows them to simply indicate which builds have passed and failed QA.

To label a build result,

1. Go to the build result.
2. Locate the 'Labels' link at the top of the screen:
   (Hint: To view a list of existing labels, click the 'Labels' link.)
3. Click the 'Add' link. This will display the following:

   Labels:
Enter labels to add to this build:
   qa-passed
   Done  Cancel

   The 'Add' link will only be visible if you have logged in to Bamboo.
4. Type the relevant label (or multiple labels, separated by commas). Note that the label will be saved in lowercase characters.
5. Click the 'Done' button.

You can also label a build result via Instant Messaging (IM).

Removing a Label from a Build Result

To remove a label from a build result,

1. Go to the build result.
2. Locate the 'Labels' link at the top of the screen:
   Labels: qa-passed  Edit
3. Click the 'Edit' link. This will display the following:

   Labels: qa-passed
   Enter labels to add to this build:
   Done  Cancel

4. Click the small red 'x' at the right of the label you want to remove.
5. Click the 'Done' button.

You must login to Bamboo before you can remove a label from a build result.

Viewing Labelled Build Results

To view all build results which have a particular label,
1. Go to any build result.
2. Click the ‘Labels’ link at the top of the screen (above the ‘Summary’ tab).
3. Click the link ‘See also labels in all projects’.
4. This will display a list of all labels that are used in Bamboo. Click the label of interest.
5. This will display a list of all build results which have that label.

Viewing Popular Labels

When labelling a build result, it can be useful to see which labels are most popular, that is, most frequently used by your colleagues.

To view the most popular labels,

1. Go to any build result (not necessarily a labelled one).
2. Click the ‘Labels’ link at the top of the screen (above the ‘Summary’ tab).
3. Click the link ‘See also labels in all projects’.
4. This will display a list all labels that are used in Bamboo. The most popular labels are indicated by the largest text.

Screenshot: ‘Labels’

Labels

This page lists all labels used in Bamboo. The bigger the text, the more build results are associated with this label. Click on a label to see the builds associated with it.

View the labels: Alphabetically | By Popularity

- atlassian_bamboo_0_8
- bamboo
- confluence23
- disabled
- javapolis
- published
- qa-passed
- qa_failed
- qa_passed
- random
- release

Handy Hint
You can click any label to see a list of all build results which have that label.

Subscribing to RSS Feeds

9. Subscribing to RSS Feeds

- Subscribing to an RSS Feed for All Build Results for All Plans
- Subscribing to an RSS Feed for All Build Results for a Particular Plan
- Subscribing to an RSS Feed for Failed Builds for All Plans
- Subscribing to an RSS Feed for Failed Builds for a Particular Plan
- Subscribing to an RSS Feed for Labelled Build Results
Subscribing to an RSS Feed for All Build Results for All Plans

To subscribe to an RSS feed for all build results for all plans,

1. Go to the Dashboard’s ‘All’ tab.
2. Locate the RSS icon at the bottom of the screen.
3. Right-click the ‘all builds’ link and copy its URL.
4. Paste the URL into your RSS reader.

Subscribing to an RSS Feed for All Build Results for a Particular Plan

To subscribe to an RSS feed for all build results for a particular plan,

1. Go to the plan.
2. Locate the RSS icon at the bottom of the screen.
3. Right-click the ‘all builds’ link and copy its URL.
4. Paste the URL into your RSS reader.

Subscribing to an RSS Feed for Failed Builds for All Plans

To subscribe to an RSS feed for failed builds for all plans,

1. Go to the Dashboard’s ‘All’ tab.
2. Locate the RSS icon at the bottom of the screen.
3. Right-click the ‘failed builds’ link and copy its URL.
4. Paste the URL into your RSS reader.

Subscribing to an RSS Feed for Failed Builds for a Particular Plan

To subscribe to an RSS feed for failed builds for a particular plan,

1. Go to the plan.
2. Locate the RSS icon at the bottom of the screen.
3. Right-click the ‘failed builds’ link and copy its URL.
4. Paste the URL into your RSS reader.

Subscribing to an RSS Feed for Labelled Build Results

To subscribe to an RSS feed for all build results with a particular label,
1. Go to the Dashboard.
2. Click any build result (not necessarily a labelled one).
3. Click the ‘Labels’ link at the top of the screen (above the ‘Summary’ tab).
4. This will display a list of any labels that are used in the build's plan. Click the link ‘See also labels in all projects’.
5. This will display a list of all labels that are used in Bamboo. Click the label of interest.
6. This will display a list of build results which have been labelled with your chosen label. Locate the RSS icon at the bottom of the screen:  
   Feed for builds labelled
7. Right-click the ‘Feed for builds labelled’ link and copy its URL.
8. Paste the URL into your RSS reader.

What is a label?

Working with Instant Messenger (IM) Notifications

Working with Instant Messenger (IM) notifications

- About Instant Messenger (IM) Notifications
- Commenting about a Build Result via IM
- Labelling a Build Result via IM

About Instant Messenger (IM) Notifications

Bamboo can send you notifications about build results for a particular plan(s). Each plan's recipients are specified by a Bamboo administrator, but you can choose whether you would like to receive your Bamboo notifications via email and/or instant messenger (IM). See Changing your Notification Preferences.

As well as receiving IM notifications, you can interact with Bamboo via IM. By responding to an IM notification, you can:

- Label a build result via IM
- Comment about a build result via IM

Labelling a Build Result via IM

You can respond to a Bamboo IM notification message with commands to label or comment on a build result.

To label a build result via Instant Messaging (IM),

In your Instant Messenger, type your comment in the following format:

```
label [build key] <labels>
```

Entering a build key is optional. If none is specified, Bamboo will look up the last time it corresponded with you and the build that was in context. The context gets updated when you specify a build key in your command, and when Bamboo sends you a notification about a particular build.

Screenshot: Interacting with Bamboo via IM
What is a label?

Commenting about a Build Result via IM

You can respond to a Bamboo IM notification message with commands to label or comment on a build result.

To comment on a build result via Instant Messaging (IM),

In your Instant Messenger, type your comment in the following format:

```
comment [build key] <comment message>
```

Entering a build key is optional. If none is specified, Bamboo will look up the last time it corresponded with you and the build that was in context. The context gets updated when you specify a build key in your command, and when Bamboo sends you a notification about a particular build.
Managing your User Profile

Managing your User Profile

- Associating your Author Name with your User Profile
- Changing your Notification Preferences
- Changing your Password
- Changing your Personal Details
- Viewing your Notifications

Changing your Personal Details

To change your Personal Details,

1. Click the 'Profile' link in the top right corner of the screen. The 'User Profile' screen will display with the 'Personal Details' tab active.
2. Click the 'Edit Profile' link.
3. Update your personal details as desired. Please note, if your user profile is managed via a single sign-on application, like Atlassian's Crowd, you will only be allowed to edit your 'Jabber Address' and 'Source Repository Alias'.

Read more about Associating your Author Name with your User Profile and Changing your Password.

Viewing your Notifications

A plan administrator can specify notifications for different users during build plan configuration. You can see which notifications are currently applicable to you, in your user profile. Please note, that you cannot change build plan notifications from your user profile.

To view your notifications,
Changing your Password

To change your Bamboo password,

1. Click the 'Profile' link in the top right corner of the screen. This will display the 'User Profile' screen.
2. Click the 'Change Password' link.
3. Type your old and new passwords.
4. Click the 'Change Password' button.

If your password is managed via a single sign-on application, like Atlassian's Crowd, this function will not be available.

Changing your Notification Preferences

Notifications in Bamboo are triggered by a range of events for a plan, including build completion, build outcomes and comments being posted against build results. You can configure whether notifications are sent for a particular event and who they are sent to. Users can choose whether to receive their notifications via email, IM, both or neither.

To change your notification preferences,

1. Click the 'Profile' link in the top right corner of the screen. This will display the 'User Profile' screen.
2. Click the 'Notifications' tab to view your notifications preferences.
3. Click the 'Edit Notification Preferences' link. The 'Edit Notification Preferences' screen will be displayed, as shown below.
4. Under 'How would you like Bamboo to send you notifications', choose how you would like to receive your notifications about build results:
   - 'Do not send notifications'
   - 'Send instant message' — if you select this option, you will need to specify your IM address in the 'Jabber Address' field on the 'Personal Details' tab.
   - 'Send email' — if you select this option, you will need to specify the format of your notification emails in the 'Email Format' field that displays.
   - 'Send email and instant message' — if you select this option, you will need to specify the format of your notification emails in the 'Email Format' field that displays, and your IM address on the 'Personal Details' tab.
5. If you have selected 'Send email' or 'Send email and instant message', the 'Email Format' field will also display. Select your preferred email format from the following options:
   - 'Text Email' — your notification emails will always be displayed in text format.
   - 'HTML Email (Multipart)' — your notification emails can be displayed in HTML format when viewed in an email reader that supports HTML rendering. Otherwise, your notification emails will display in text format.
6. Click the 'Save' button.
### Associating your Author Name with your User Profile

An author is any person who contributes to a build by checking-in code to a repository that is associated with a Bamboo plan. An author need not be a Bamboo user. Your Author Name is your login name for the source-code repository.

If your Bamboo User Profile has not yet been associated with your Author Name, then:
- your 'My Bamboo' screen will not contain any data about your recent builds.
- your 'Author' information will not include a 'User Details' tab.

To associate your Author Name with your User Profile,

1. Click the 'Profile' link in the top right corner of the screen. This will display the 'User Profile' screen.
2. Click the 'Edit Personal Details' link.
3. In the 'Source Repository Alias' field, select your Author Name from the list. If your Author Name does not appear in the list, select 'Add Alias' (the second item in the list) then type your Author Name in the 'New Alias' field. Note that your Author Name (Alias) need not be identical to your User Name.
4. Click the 'Save' button.
Bamboo Development Hub

Looking for existing plugins? See the existing plugins and extensions written by the community in the Bamboo Extensions space.

Developing Plugins

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.

The Bamboo Plugin Guide

These documents are specifically about plugins for Bamboo. There’s a page for each plugin module type that Bamboo supports. You can combine multiple plugin modules inside a single plugin to accomplish complex tasks. There is also other reference information including.

Understanding Bamboo Source Code

Setting up Bamboo Development Environment in IDEA

This document gives instructions on how to build an IDEA project from Bamboo's source distribution.

Building a Bamboo War Distribution From Source

This document gives instructions on how to build a bamboo.war distribution from Bamboo's source distribution.

Other resources

Reference Information

- Bamboo Plugin Module Types
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
- Web Resources

Tutorials and Example Plugins

- Bamboo Plugin Tutorial
  - Tutorial 1 - Getting Started with a Simple Post Build Labeller
  - Tutorial 2 - Configurable Regex Labeller

Sample plugin source available to be checked out from our source repository:

- Build Number Stamper Plugin - Places the build number in a file within the source code directory
- File Repository Plugin - An example repository plugin, based on the file system
- Bamboo Labeller Plugin - Parses build logs for a pre-configured regex expression and Labels the build if it is found
- Bamboo Auto-Favourite Plugin - Automatically marks a build plan as a favourite if you make a commit to that build
- Bamboo VCS Version plugin - Records the revision number for the build in the custom data map to make it available for variable
Please keep in mind that customisations are beyond the scope of what's covered by Atlassian Support.

Bamboo Plugins Overview

A Bamboo plugin is a single JAR containing code, a plugin descriptor (XML) and usually some Freemarker 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 post-build action) 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:

```
<atlassian-plugin key="com.atlassian.plugin.sample" name="Sample Plugin">
  <!-- the plugin key must be unique, think of it as the 'package' of the plugin -->
  <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"/>
</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.

**How does the Bamboo plugin system work?**

Below is a diagram which shows Bamboo's build process flow, as well as the available plugin modules you can build.

The following types of plugin modules are supported by Bamboo:

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Since version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>builder</td>
<td>1.0</td>
<td>Builder Plugin Module</td>
<td>Add new builders to Bamboo.</td>
</tr>
<tr>
<td>xwork</td>
<td>1.0</td>
<td>XWork Plugin Module</td>
<td>XWork actions and views bundled with the plugin. This enables building generic user interfaces.</td>
</tr>
<tr>
<td>report</td>
<td>1.0</td>
<td>Report Module</td>
<td>Defines a report of build telemetry data.</td>
</tr>
<tr>
<td>preBuildQueuedAction</td>
<td>2.1</td>
<td>Pre Build Queued Action</td>
<td>Prepends a custom synchronous process to the build. Occurs before a build is queued.</td>
</tr>
<tr>
<td>preBuildAction</td>
<td>1.1</td>
<td>Pre Build Action Module</td>
<td>Prepends a custom synchronous process to the build. Occurs before the builder has run.</td>
</tr>
<tr>
<td>buildProcessor</td>
<td>1.0</td>
<td>Build Processor Module</td>
<td>Append a custom synchronous process to the build. Occurs on the Agent (both local and remote) after the builder has run.</td>
</tr>
<tr>
<td>buildProcessorServer</td>
<td>2.0</td>
<td>Build Processor Server Module</td>
<td>Append a custom synchronous process to the build. Occurs on the Server after the build has been returned by the Agent.</td>
</tr>
<tr>
<td>buildCompleteAction</td>
<td>1.0</td>
<td>Build Complete Action Module</td>
<td>Add a custom asynchronous action after the build process has completed.</td>
</tr>
<tr>
<td>postBuildIndexWriter</td>
<td>1.0</td>
<td>Post Build Index Writer Module</td>
<td>Writes custom build data into the index for report generation.</td>
</tr>
<tr>
<td>indexReader</td>
<td>1.0</td>
<td>Index Reader Module</td>
<td>Reads out custom index information written by the postBuildIndexWriter from the index.</td>
</tr>
<tr>
<td>web-item</td>
<td>1.0</td>
<td>Web Item Module</td>
<td>Add new links to the Bamboo interface.</td>
</tr>
<tr>
<td>web-section</td>
<td>1.0</td>
<td>Web Section Module</td>
<td>Add a new section to the Bamboo interface.</td>
</tr>
<tr>
<td>notification-condition</td>
<td>1.1 to 2.2</td>
<td>Notification Condition Module</td>
<td>Add new notification condition.</td>
</tr>
<tr>
<td>notificationType</td>
<td>2.2</td>
<td>Notification Type Module</td>
<td>Add a new notification type</td>
</tr>
<tr>
<td>notificationRecipient</td>
<td>2.2</td>
<td>Notification Recipient Module</td>
<td>Add a new notification recipient type.</td>
</tr>
</tbody>
</table>
Setting up a Bamboo Plugin Project

Refer to the Getting started with Atlassian plugins guide for information on how to set up a Bamboo plugin using the Atlassian Plugin Archetypes. You will require Maven 2 and JDK 5.

If you have a Commercial License you can obtain the Bamboo Source code. Refer to the Building Bamboo from source page for more information on how set this up.

Testing your Bamboo plugin.

Plugins in Bamboo are not dynamically loadable. They must be installed in Bamboo's WEB-INF/lib and the application restarted. The Atlassian Plugin Maven Archetypes come with a profile called plugin-debug which makes testing your plugin easy. Running the command

```
mvn -Pplugin-debug
```

will compile your plugin's classes, create a jar in the /target directory, download a copy of the Bamboo war, copy the plugin jar into Bamboo's WEB-INF/lib directory and start up Bamboo. Maven will also install some default data such as the license and database configuration.

Bamboo will be running at http://localhost:1990/bamboo, so you can go directly there in your browser and see your latest changes. You'll be able to log in with the username admin and the password admin.

As you make changes, you'll have to kill the Bamboo process (ctrl-c) and then run mvn -Pplugin-debug again.

When you run Bamboo with the plugin-debug profile, it is automatically configured to accept a remote debugger. So you'll want to create a remote debug target in your IDE for your plugin and/or the bamboo source.

Deploy a Bamboo Plugin

Installing plugins in Bamboo is easy.

Once you have downloaded or created your plugin jar, follow these steps:

1. Shut down Bamboo
2. copy $MY_COOL_PLUGIN.jar into '/webapp/WEB-INF/lib/
3. Start up Bamboo. Your plugin should be automatically installed and activated.
4. Enjoy!

Built-in Bamboo system plugins

A number of functions and areas within Bamboo 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-actions-plugin.xml - the built in per-plan build expiry configuration
- system-builder-plugin.xml - the built in builders, including Ant, Maven, and Maven 2.
- system-clover-plugin.xml - the built in Clover analytics.
- system-jira-plugin.xml - the built in JIRA issue integration.
- system-labelling-plugin.xml - the built in automatic build labeller.
- `system-reports-plugin.xml` - the built reports of builds grouped by time periods under the Reports tab.
- `system-repository-plugin.xml` - definition for the CVS, SVN and Perforce repositories shipped with Bamboo.
- `system-triggerReason-plugin.xml` - definition of all the built in build trigger reasons such as manual build, dependency build and Scheduled build.
- `system-webUI-plugin.xml` - the built in menu items under the Administration tab as well as the tab menu items on the View Plan page and the View Build Results page.
- `system-notifications-plugin.xml` - the build in notification conditions, including all builds, failed builds, after X failed builds.

### Further Information

- Bamboo Plugin Module Types
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
- Web Resources

### Bamboo Plugin Module Types

The following types of plugin modules are supported by Bamboo:

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Since version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>builder</td>
<td>1.0</td>
<td>Builder Plugin Module</td>
<td>Add new builders to Bamboo.</td>
</tr>
<tr>
<td>xwork</td>
<td>1.0</td>
<td>XWork Plugin Module</td>
<td>XWork actions and views bundled with the plugin. This enables building generic user interfaces.</td>
</tr>
<tr>
<td>report</td>
<td>1.0</td>
<td>Report Module</td>
<td>Defines a report of build telemetry data.</td>
</tr>
<tr>
<td>preBuildQueuedAction</td>
<td>2.1</td>
<td>Pre Build Queued Action</td>
<td>Prepends a custom synchronous process to the build. Occurs before a build is queued.</td>
</tr>
<tr>
<td>preBuildAction</td>
<td>1.1</td>
<td>Pre Build Action Module</td>
<td>Prepends a custom synchronous process to the build. Occurs before the builder has run.</td>
</tr>
<tr>
<td>buildProcessor</td>
<td>1.0</td>
<td>Build Processor Module</td>
<td>Append a custom synchronous process to the build. Occurs on the Agent (both local and remote) after the builder has run.</td>
</tr>
<tr>
<td>buildProcessorServer</td>
<td>2.0</td>
<td>Build Processor Server Module</td>
<td>Append a custom synchronous process to the build. Occurs on the Server after the build has been returned by the Agent.</td>
</tr>
<tr>
<td>buildCompleteAction</td>
<td>1.0</td>
<td>Build Complete Action Module</td>
<td>Add a custom asynchronous action after the build process has completed.</td>
</tr>
<tr>
<td>postBuildIndexWriter</td>
<td>1.0</td>
<td>Post Build Index Writer Module</td>
<td>Writes custom build data into the index for report generation.</td>
</tr>
<tr>
<td>indexReader</td>
<td>1.0</td>
<td>Index Reader Module</td>
<td>Reads out custom index information written by the postBuildIndexWriter from the index.</td>
</tr>
<tr>
<td>web-item</td>
<td>1.0</td>
<td>Web Item Module</td>
<td>Add new links to the Bamboo interface.</td>
</tr>
<tr>
<td>web-section</td>
<td>1.0</td>
<td>Web Section Module</td>
<td>Add a new section to the Bamboo interface.</td>
</tr>
<tr>
<td>notification-condition</td>
<td>1.1 to 2.2</td>
<td>Notification Condition Module</td>
<td>Add new notification condition.</td>
</tr>
<tr>
<td>notificationType</td>
<td>2.2</td>
<td>Notification Type Module</td>
<td>Add a new notification type</td>
</tr>
<tr>
<td>notificationRecipient</td>
<td>2.2</td>
<td>Notification Recipient Module</td>
<td>Add a new notification recipient type.</td>
</tr>
<tr>
<td>repository</td>
<td>1.1</td>
<td>Source Repository Module</td>
<td>Add a custom source repository.</td>
</tr>
<tr>
<td>triggerReason</td>
<td>2.0</td>
<td>Trigger Reason Module</td>
<td>Add a new trigger reason.</td>
</tr>
</tbody>
</table>
**Bamboo Event Listener Module**

*Description*

The Bamboo Event Listener module allows you to register an event listener against any event that is currently thrown in Bamboo and perform arbitrary operations. Some examples include sending off notifications or terminating a build when it hung.

*Interface*

All event listener modules implement the com.atlassian.event.EventListener interface or alternatively the com.atlassian.bamboo.event.HibernateEventListener which will automatically provide you with a hibernate session to access to the database (required for writing notification plugins).

*Sample Module Descriptor Element*

```xml
  <bambooEventListener
    key="buildHungNotificationListener"
    name="Build Hung Notification Listener"
    class="com.atlassian.bamboo.notification.buildhung.BuildHungNotificationListener">
    <description>Listens for if a build has hung.</description>
  </bambooEventListener>
```

*Available Events to listen to*

There is a number of events currently available in Bamboo. Each event contains information regarding the context in which it was thrown.

<table>
<thead>
<tr>
<th>Event</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentConfigurationUpdatedEvent</td>
<td>Occurs when a specific agent is updated (e.g. the agent has been create or disabled)</td>
</tr>
<tr>
<td>AllAgentsUpdatedEvent</td>
<td>Occurs if there is an update that will effect all agents e.g. disabling/enabling all agents or updating capability sets</td>
</tr>
<tr>
<td>BambooErrorEvent</td>
<td>These are any errors generated by Bamboo (not your build). These are the same errors which are displayed as System Errors in the Bamboo UI. This may include failing checkouts, agents going offline etc.</td>
</tr>
<tr>
<td>BuildCommentedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildCompletedEvent</td>
<td>Thrown after build completed and results saved to the database, not necessarily before or after any CustomBuildCompleteActions are performed</td>
</tr>
<tr>
<td>BuildConfigurationUpdatedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildCreatedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildDeletedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildQueuedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildHungEvent</td>
<td>Thrown when Bamboo's build monitoring detects the build has met the configured build hanging criteria</td>
</tr>
<tr>
<td>BuildRequirementUpdatedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildResultDeletedEvent</td>
<td></td>
</tr>
<tr>
<td>BuildResultSummaryUpdatedEvent</td>
<td>Currently the only way to edit a build summary is editing the JIRA issues associated with the build.</td>
</tr>
</tbody>
</table>
**BuildsMovedEvent**
Thrown when moving Build Plan between projects.

**BuildTriggeredEvent**
Thrown after a bamboo has detected that the plan needs building, before the build is placed on the queue.

**ChangeDetectionRequiredEvent**

**ElasticConfigUpdatedEvent**
Thrown when a user updates the Elastic Bamboo Configuration

**EmailEvent**
Contain an email to be sent

**IMEvent**
Contain an instant message to be sent

**InitialBuildRequiredEvent**
Thrown if bamboo detects that an initial clean build is required for a plan (usually after plan creation or importing data)

**PostBuildCompletedEvent**
Thrown after the build is saved AND the build process has been completely cleaned up

Most of these events either extend a BuildEvent (build key available) or a BuildResultEvent (build key and build number available).

**Build Agent Requirement Filter Module**

**Description**

The build agent requirement filter module allows you to customise the list of Agents a plan can be built on. Bamboo will determine which agents meet the given requirement set, then send the list of agents through the filter for any further adjustments to the list.

**Interface**

Build Agent Requirement Filter modules must implement the `com.atlassian.bamboo.v2.build.BuildAgentRequirementFilter` interface.

**Sample Module Descriptor Element**

```xml
<buildAgentRequirementFilter key="labManagerAgentFiler" name="LabManager Agent filter"
  class="com.atlassian.bamboo.plugins.labmanager.LabManagerAgentFilter">
  <description>Ensures lab manager builds don't run on normal agents</description>
</buildAgentRequirementFilter>
```

**RELATED TOPICS**

- **Bamboo Plugin Guide**
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
    - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
    - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
- **Bamboo's Build Process**
- **Accessing Bamboo Components From Plugin Modules**
- **Common Bamboo Classes**
- **Bamboo Persistence using Bandana**
- **Downloadable Plugin Resources**
- **Web Resources**
- **Bamboo REST APIs**
Command Decorator Module

Description

The Command Decorator module allows you to add additional parameters to the command line of a builder before a build occurs.

Interface

Command Decorator modules must implement the `com.atlassian.bamboo.command.CommandDecorator` interface.

Sample Module Descriptor Elements

```xml
<commandDecorator
    key="cloverMavenCommandDecorator"
    name="Clover Maven Command Line Decorator"
    class="com.atlassian.bamboo.builder.command.CloverMavenCommandDecorator">
  <description>
    A command decorator which automatically collects code coverage for a Maven2 build.
  </description>
</commandDecorator>
```

```xml
<commandDecorator
    key="cloverAntCommandDecorator"
    name="Clover Ant Command Line Decorator"
    class="com.atlassian.bamboo.builder.command.CloverAntCommandDecorator">
  <description>
    A command decorator which automatically collects code coverage for an Ant build.
  </description>
</commandDecorator>
```

RELATED TOPICS

- Bamboo Plugin Guide
- Bamboo Plugin Module Types
  - Bamboo Event Listener Module
  - Build Agent Requirement Filter Module
  - Command Decorator Module
  - Notification Recipient Module
  - Notification Type Module
    - Building a Notification Plugin
  - Builder Plugin Module
  - Command Decorator Module
  - Configurable Plugin
  - Post-Build Processing in Bamboo 2.0
Notification Recipient Module

Description

The Notification Recipient module allows you to create a custom recipient. A recipient is responsible for evaluating the methods of delivery and destinations of the notification. They do this by generating the appropriate Notification Transports. A Recipient can also take in configuration information from the user.

As a user can configure any Recipient with any Notification Type you need to ensure your recipient will behave when it doesn't have the required contextual information as sometimes a particular Recipient won't really apply (e.g. a committer recipient when there is no build).

Notification Transport

Notification Transports are responsible for actually sending the notification. They set up the appropriate protocols, retrieve the content from the Notification object and fire off the resulting message through appropriate channels. The recipient is responsible for populating these with the appropriate information, e.g. the email address or the IM address. There are three types of transports in bamboo currently: Multipart Email, Text Email and Instance Message.
You can find more information on the various Notification components on the Building a Notification Plugin page.

**Interface**

Notification Recipients must implement the `com.atlassian.bamboo.notification.NotificationRecipient` interface. There is an abstract class `com.atlassian.bamboo.notification.recipients.AbstractNotificationRecipient` which you can extend that implements some of these methods for you with default values.

**Sample Module Descriptor**

```xml
<notificationRecipient key="recipient.user" name="User Recipient"
class="com.atlassian.bamboo.notification.recipients.UserRecipient"
weight="10">
  <description>User</description>
  <resource type="freemarker" name="edit" location=
  "templates/plugins/notifications/userRecipientEdit.ftl"/>
</notificationRecipient>
```

**Notification Type Module**

**Description**

The NotificationType module identifies your custom notification. Users will be able to select this from the drop down menu and provide any configuration information you require. This class is also responsible for determining whether the notification criteria is met (e.g. did the build fail?).

For more information on building a completed Notification Plugin see Building a Notification Plugin

**Interface**

Notification Types must implement the `com.atlassian.bamboo.notification.NotificationType` interface. There is an abstract class `com.atlassian.bamboo.notification.AbstractNotificationType` which you can extend that implements a lot of these methods for you.

**Sample Module Descriptor**

```xml
<notificationType key="buildError" name="Build Error"
class="com.atlassian.bamboo.notification.builderror.BuildErrorNotificationType"
weight="40">
  <description>Build Error</description>
  <resource type="freemarker" name="edit" location="templates/plugins/notifications/buildErrorNotificationEdit.ftl"/>
</notificationType>
```

**Building a Notification Plugin**

If you wish to customise the content of your notifications this can be achieved just by editing the notification templates.

Below I have tried to outline the concepts involved in creating a notification plugin. To create a notification plugin you will need to create the following classes:

- Notification Type
- Notification Listener
- Notification class

**Notification Components Overview**

Whilst there are 3 classes to be created there are other components which work together to send the notification.

- **Notification Type** - Identifies individual types of notifications and stores configuration data.
- **Notification Rule** - A user configured notification, consists of a NotificationType and a Notification Recipient.
- **Event** - Predefined trigger which you can use to trigger your notification.
- **Event Listener** - Listens for the predefined event. It filters the NotificationRules and extracts the NotificationRecipients. A Notification is then created containing these recipients and passed to the dispatcher.
- **Notification** - self contained object which is used to dispatch the required notifications. It manages recipients and generates notification content.
- **Notification Recipient** - An entity which will receive a notification (e.g. User: bob or Group: bamboo-admin)
- **Transport** - A method of notification (e.g. Text Email)
- **Notification Dispatcher** - Uses the Notification to dispatch individual notifications to recipients
Notification Type

The NotificationType object identifies your custom notification. Users will be able to select this from the drop down menu and provide any configuration information you require.

You can find more information in the Notification Type Plugin Module documentation.

Notification Rule

When a user configures a notification, it will create a NotificationRule which consists of the selected NotificationType and selected NotificationRecipient. A user can select any NotificationType to go with any NotificationRecipient. When retrieving the notification rules from the database, it will retrieve rules for all notification types, you will need to filter the rules to find those which correspond to your NotificationType.

Event Listener

The Event Listener will listen for an event to be fired in Bamboo which you would like to cause a notification. These events are predefined in Bamboo. You can find a list of available events as well as more information on implementing an event listener in the Bamboo Event Listener Module documentation.

A notification event listener is responsible for retrieving the Notification Rules from the database, filtering them (determining which will/wont be sent) and creating a Notification object containing the recipients that will be receiving the notification. This Notification object then needs to be passed to the NotificationDispatcher for processing. Context information is available in the event object and you can inject Managers to obtain any other required information.

Example Listener Class

The following is an example of a very basic notification listener.

```java
public class BuildCompletedNotificationListener implements HibernateEventListener {
    private NotificationManager notificationManager;
    private NotificationDispatcher notificationDispatcher;
    private BuildManager buildManager;

    public Class[] getHandledEventClasses() {
        Class[] array = {BuildCompletedEvent.class};
        return array;
    }

    public void handleEvent(Event event) {
        BuildCompletedEvent buildEvent = (BuildCompletedEvent) event;
        Build build = buildManager.getBuildByKey(buildEvent.getBuildPlanKey());

        Notification myNotification = new MyCustomNotification();
        myNotification.setEvent(event);

        Set<NotificationRule> rules = notificationManager.getNotificationRules(build);
        for (NotificationRule rule : rules) {
            NotificationType notificationType = rule.getNotificationType();
            if (notificationType instanceof MyCustomNotificationType) {
                if (notificationType.isNotificationRequired(event)) {
                    NotificationRecipient recipient = rule.getNotificationRecipient();
                    myNotification.addRecipient(recipient);
                }
            }
        }

        notificationDispatcher.dispatchNotifications(myNotification);
    }

    public void setNotificationManager(NotificationManager notificationManager) {
        this.notificationManager = notificationManager;
    }

    public void setNotificationDispatcher(NotificationDispatcher notificationDispatcher) {
        this.notificationDispatcher = notificationDispatcher;
    }

    public void setBuildManager(BuildManager buildManager) {
        this.buildManager = buildManager;
    }
}
```
Notification

The Notification class is a self contained object which is used to dispatch the required notifications. You are not required to register your Notification object as a plugin module as it will be instantiated by your own Event Listener. Notifications must implement the `com.atlassian.bamboo.notification.Notification` interface. There is an abstract class `com.atlassian.bamboo.notification.AbstractNotification` which you can extend that implements a lot of these methods for you.

A Notification will contain any context information which you will need to generate content (usually the event), a list of recipients to send the notification to and a list of recipients to exclude.

The notification class is responsible for generating the content (body) of the notifications. There is no specific generate the content of your notifications, we use Freemarker for our notifications as the templates are customisable outside of the application.

Example Content Generation

The following is an example of generating freemarker email content. As you are instantiating this class yourself it will not be automatically injected with managers. You will need to manually add them on creation (you can get them automatically injected into the event listener). The templateRenderer is a bean which you will need to pass into your Notification on creation.

```java
@Nullable
public String getHtmlEmailContent() throws Exception
{
    Event event = getEvent();
    if (event != null)
    {
        Map<String, Object> context = new HashMap<String, Object>();
        populateContext(context); // put any information the context which you want available in your
                                    // freemarker template e.g. the build, buildResultsSummary etc
        try
        {
            return templateRenderer.render("notification-templates/BuildCompletedTextEmail.ftl", context);
        }
        catch (Exception e)
        {
            log.error("Could not render email content", e);
            return null;
        }
    }
    else
    {
        log.error("Event is null, could not create Email content for " + getDescription());
        return null;
    }
}
```

Advanced Notification Interface

If you would like more freedom over how the notification emails get generated you can implement the "com.atlassian.bamboo.notification.ExtendedNotification" interface. This will give you the generated com.atlassian.mail.Email object for you to make any other changes, e.g you could add attachments or change who the email is from based on some context information (e.g the person who made the comment being notified)

The Notification Dispatcher

Once you send your Notification object off to the dispatcher it will do the following:

1. Retrieve the notification recipients from your Notification object
2. The notification recipients will then evaluate a list of notification transports
3. Retrieve and evaluate the transports which should get excluded from the notification and remove these from the transports list
4. Each transport will then send the notification (e.g set up communication protocols and send the instant message)

Notification Recipient

A Notification Recipient is responsible for evaluating the methods of delivery and destinations of the notification. They do this by generating the appropriate Notification Transports. A Recipient can also take in configuration information from the user

A user can select any NotificationType to go with any NotificationRecipient, so you need to ensure that your plugin can handle all types of recipients. For example, the committer recipient needs to have the commits populated by your listener. Or, you need to ensure committer recipients are not added to your Notification.

Notification Recipients are also pluginable so you can create your own. You can find more information in the notification recipient plugin module documentation.

Notification Transport

Notification Transports are responsible for actually sending the notification. They set up the appropriate protocols, retrieve the content from the
Notification object and fire off the resulting message through appropriate channels. The recipient is responsible for populating these with the appropriate information, e.g., the email address of the IM address. There are three types of transports in bamboo currently: Multipart Email, Text Email and Instance Message.

Questions?

Why do I have excluded recipients?
Recipients are not evaluated till dispatch, which means if a user has configured a group recipient, you can not control who out of that group will or won’t receive a notification. So we have provided a mechanism to allow you to exclude recipients from the list (which will also get dispatched at run time).

For example, the build-commented notification we do not want to get sent to the user who actually created the comment. We would add that user to the list, and when the groups get evaluated this user would get excluded but all other members would still receive the notification.

Can I just provide HTML content and not text content?
It is not mandatory to send out both HTML and text content, however, as a user can select to JUST receive text emails they will not get sent anything if the email is blank, so we recommend always providing a text option.

What is a multipart email?
A Multipart email contains both text content and HTML content. It is then up to the users’ mail browser to select which type of email they would like to view.

Builder Plugin Module

Description
A plugin module which defines a builder in Bamboo, such as Maven, Maven2, or Ant.

Interface
Builder modules must implement the `com.atlassian.bamboo.builder.Builder` interface.

Sample Module Descriptor Element

```
<builder key="mvn2" name="Maven 2.x Builder"

class="com.atlassian.bamboo.builder.Maven2Builder">

<description>A Maven 2.x Builder</description>

<resource type="freemarker" name="edit" location="templates/plugins/builder/mavenBuilderEdit.ftl"/>

<resource type="freemarker" name="view" location="templates/plugins/builder/mavenBuilderView.ftl"/>

</builder>
```

RELATED TOPICS

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
      - Building a Notification Plugin
    - Builder Plugin Module
  - Source Repository Module
    - Repository Extensions
  - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
  - Bamboo’s Build Process
  - Accessing Bamboo Components From Plugin Modules
  - Common Bamboo Classes
Source Repository Module

Description

A plugin module which defines a repository in Bamboo, such as CVS, Subversion, or Perforce.

Sample Module Descriptor Element

```xml
<repository key="svn" name="SVN Repository" class="com.atlassian.bamboo.repository.svn.SvnRepository">
  <description>A Subversion Repository</description>
  <resource type="freemarker" name="edit" location="templates/plugins/repository/svnRepositoryEdit.ftl"/>
  <resource type="freemarker" name="view" location="templates/plugins/repository/svnRepositoryView.ftl"/>
</repository>
```

Notes

Plugins of this type must implement the `com.atlassian.bamboo.repository.Repository` interface. For (comparative) simplicity, you should use the class `AbstractRepository` as a starting point and extend from that.

RELATED TOPICS

- Bamboo Plugin Guide
- Bamboo Plugin Module Types
  - Bamboo Event Listener Module
  - Build Agent Requirement Filter Module
  - Command Decorator Module
  - Notification Recipient Module
Repository Extensions

You can extend the standard repository functionality by implementing any of the optional interfaces described below.

On this page:

- com.atlassian.bamboo.v2.build.repository.RepositoryEventAware
  - Description
  - Methods
com.atlassian.bamboo.v2.build.repository.RepositoryEventAware

Description

The com.atlassian.bamboo.v2.build.repository.RepositoryEventAware interface allows you to instruct the repository to perform a custom action before and/or after the checkout/update occurs.

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void preRetrieveSourceCode(@NotNullBuildContextBuildContext) ;</code></td>
<td>This will run before the retrieveSourceCode method is called. It allows you to do custom actions to prepare for the checkout, such as some customised cleanup of the source directory.</td>
</tr>
<tr>
<td><code>void postRetrieveSourceCode(@NotNullBuildContextBuildContext) ;</code></td>
<td>This will run after the retrieveSourceCode method is called. The extension point allows you to easily set custom data, do any post checkout collection of data.</td>
</tr>
</tbody>
</table>

Trigger Reason Module

Description

A plugin module which defines a trigger reason in bamboo. For example, DependencyTriggerReason, InitialbuildTriggerReason, ScheduledTriggerReason. You want to implement a trigger reason if you are kicking off a build for some reason other than those which are currently defined in Bamboo.

Interface

Trigger Reason modules must implement the com.atlassian.bamboo.v2.build.trigger.TriggerReason interface.

Sample Module Descriptor Element

```xml
<triggerReason key="InitialBuildTriggerReason" name="InitialBuildTriggerReason"
    class="com.atlassian.bamboo.v2.trigger.InitialBuildTriggerReason">
  <renderer>com.atlassian.bamboo.v2.build.trigger.DefaultTriggerReasonRenderer</renderer>
  <description>Initial build trigger reason</description>
  <resource type="freemarker" name="longDescriptionTemplate" location="build/triggerReasons/initialBuildTriggerReason-long.ftl"/>
  <resource type="freemarker" name="shortDescriptionTemplate" location="build/triggerReasons/initialBuildTriggerReason-short.ftl"/>
</triggerReason>
```

RELATED TOPICS

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
      - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
      - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
Pre Build Queued Action

Description

Like the Build Processor Module, this allows you to specify a custom action to take place. However, the difference is that this will run before the build has been queued and the build made executable for both local and remote Bamboo agents. This is the first custom action that is run when a new Bamboo build is triggered, thus build state can be modified at this time.

Interface

Pre Build Queued modules must implement the CustomPreBuildQueuedAction interface.

Sample Module Descriptor Element

```xml
<preBuildQueuedAction key="startVirtualMachine" name="Start Virtual Machine Action"
    class="com.atlassian.bamboo.plugins.vm.StartVirtualMachineAction">
    <resource type="freemarker" name="edit" location="templates/vmEdit.ftl"/>
    <resource type="freemarker" name="view" location="templates/vmView.ftl"/>
    <description>Starts a virtual machine</description>
</preBuildQueuedAction>
```

RELATED TOPICS

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
Pre Build Action Module

Description

The **PreBuildAction** module allows you to define a custom process that runs before your build begins.

This will occur immediately before the builder begins execution. The **PreBuildAction** will have access to the **BuildResults** object which contains the information for the build.

Interface
Pre Build Action modules must implement the `com.atlassian.bamboo.build.CustomPreBuildAction` interface.

**Sample Module Descriptor Element**

```xml
<preBuildAction key="vcsVersion" name="VCS Version Collector" class="com.atlassian.bamboo.vcsversion.VCSVersionReader">
  <description>A custom action that reads the identifier of a source repository version and stores it into the custom data map of a build.</description>
  <resource type="freemarker" name="edit" location="templates/plugins/preBuildAction/vcsVersionReaderEdit.ftl"/>
  <resource type="freemarker" name="view" location="templates/plugins/preBuildAction/vcsVersionReaderView.ftl"/>
</preBuildAction>
```

**RELATED TOPICS**

- **Bamboo Plugin Guide**
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
      - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
      - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
  - Bamboo's Build Process
  - Accessing Bamboo Components From Plugin Modules
  - Common Bamboo Classes
  - Bamboo Persistence using Bandana
  - Downloadable Plugin Resources
  - Web Resources
- **Bamboo REST APIs**
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
  - Comparing the Bamboo Remote API to the Bamboo REST APIs
- **Bamboo Remote API**
  - Build Results Filters
  - Other Services
  - Authentication Services
  - Elastic Bamboo Services
  - Build Results Services
  - Build Services
- **Bamboo Developer FAQ**
  - How do I inject managers into my plugin?
  - How do I search for previous build result?
  - How do I start a build programatically?
  - How do I trigger off a build from my action?
- **Bamboo Plugin Tutorial**
  - Tutorial 1 - Getting Started with a Simple Post Build Labeller
  - Tutorial 2 - Configurable Regex Labeller
- **Changes by Version**
  - Changes for Bamboo 2.0
    - Changes to Bamboo's Configuration UI require the ConfigurablePlugin
    - Post-Build Processing in Bamboo 2.0
    - Build Process for 2.0
Build Processor Module

Description

The **BuildProcessor** module allows you to define a custom process that runs during the build.

This will occur immediately after the builder has completed execution and the test results have been captured. The **BuildProcessor** forms part of build execution run and the result of the execution is only registered after your custom **BuildProcessor** has completed. This means that your plugin has the capability to affect the final **BuildState** of your build (i.e. success/failure).

Bamboo has two different build processors, this one and the BuildProcessorServer. The BuildProcessor runs on the agent and has full access to the file system (but no access to many of the Bamboo managers and the database). The BuildProcessorServer, on the other hand, runs on the server side after a build has returned from the agent.

If you’re writing a plugin that will only ever be run in a non-distributed environment, you can use either plugin points.

Interface

Build Processor modules must implement the **CustomBuildProcessor** interface.

Sample Module Descriptor Element

```xml
<buildProcessor key="cloverResultCollector" name="Clover Results Collector"
    class="com.atlassian.bamboo.builder.coverage.CloverBuildProcessor">
    <skipIfFailed>true</skipIfFailed>
    <resource type="freemarker" name="edit" location="templates/plugins/buildProcessor/cloverProcessorEdit.ftl"/>
    <resource type="freemarker" name="view" location="templates/plugins/buildProcessor/cloverProcessorView.ftl"/>
    <description>A Clover report parser and data collector</description>
</buildProcessor>
```

RELATED TOPICS

- Bamboo Plugin Guide
- Bamboo Plugin Module Types
  - Bamboo Event Listener Module
  - Build Agent Requirement Filter Module
  - Command Decorator Module
  - Notification Recipient Module
  - Notification Type Module
  - Building a Notification Plugin
  - Builder Plugin Module
  - Source Repository Module
  - Repository Extensions
  - Trigger Reason Module
  - Pre Build Queued Action
  - Pre Build Action Module
  - Build Processor Module
  - Build Processor Server Module
  - Build Complete Action Module
  - Index Reader Module
  - Post Build Index Writer Module
  - Report Module
  - XWork Plugin Module
  - Web Item Module
  - Web Section Module
  - Notification Condition Module
- Bamboo’s Build Process
The **BuildProcessorServer** module allows you to define a custom process that runs during the build.

This will occur after the build is returned to the server (from both local and remote agents). The build has completed execution and the test results have been captured. The BuildProcessorServer forms part of build execution run and the result of the execution is only registered after your custom BuildProcessorServer has completed. This means that your plugin has the capability to affect the final BuildState of your build (i.e. success/failure).

Because this module runs on the server you have access to any of the Managers and data that is stored in Bamboo. However, it may not have access to the source code or build environment.

Bamboo has two different build processors, this one and the BuildProcessor. The BuildProcessorServer runs on the server side after a build has returned from the agent. It has access to Managers and the database. The BuildProcessor on the other hand runs on the agent and has full access to the file system (but no access to many of the Bamboo managers and the database).

If you're writing a plugin that will only ever be run in a non-distributed environment, you can use either plugin points.

**Interface**

BuildProcessorServer modules must implement the **CustomBuildProcessorServer** interface.

**Sample Module Descriptor Element**
<buildProcessorServer key="cloverDeltaCalculator" name="Clover Delta Calculator"
class="com.atlassian.bamboo.builder.coverage.CloverDeltaCalculator">
  <skipIfFailed>true</skipIfFailed>
  <description>Calculates coverage deltas between builds</description>
</buildProcessorServer>

RELATED TOPICS

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
      - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
      - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
  - Bamboo's Build Process
  - Accessing Bamboo Components From Plugin Modules
  - Common Bamboo Classes
  - Bamboo Persistence using Bandana
  - Downloadable Plugin Resources
  - Web Resources
- Bamboo REST APIs
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
  - Comparing the Bamboo Remote API to the Bamboo REST APIs
- Bamboo Remote API
  - Build Results Filters
  - Other Services
  - Authentication Services
  - Elastic Bamboo Services
  - Build Results Services
  - Build Services
- Bamboo Developer FAQ
  - How do I inject managers into my plugin?
  - How do I search for previous build result?
  - How do I start a build programatically?
  - How do I trigger off a build from my action?
- Bamboo Plugin Tutorial
  - Tutorial 1 - Getting Started with a Simple Post Build Labeller
  - Tutorial 2 - Configurable Regex Labeller
- Changes by Version
  - Changes for Bamboo 2.0
    - Changes to Bamboo's Configuration UI require the ConfigurablePlugin
    - Post-Build Processing in Bamboo 2.0
    - Build Process for 2.0
    - Repository plugin changes in 2.0
    - Updates to the build processing plugins
  - Changes for Bamboo 2.1
  - Changes for Bamboo 2.1.5
  - Changes for Bamboo 2.2
  - Changes for Bamboo 2.3
- Building a Bamboo War Distribution From Source
- Setting up Bamboo Development Environment in IDEA
  - Building Bamboo In IDEA - Troubleshooting
Build Complete Action Module

Description

Like the Build Processor Module, this allows you to specify a custom action to take place. However, the difference is that this will run after the full build result has been registered. The build is deemed to have completed before the BuildCompleteAction is fired. Hence, build complete actions cannot impact the state of the build.

Interface

Build Complete Action modules must implement the CustomBuildCompleteAction interface.

Sample Module Descriptor Element

```xml
<buildCompleteAction key="autoLabeller" name="Build Automatic Labeller" class="com.atlassian.bamboo.labels.AutoLabelBuildCompleteAction">
  <resource type="freemarker" name="edit" location="templates/plugins/buildCompleteAction/autoLabellingEdit.ftl"/>
  <resource type="freemarker" name="view" location="templates/plugins/buildCompleteAction/autoLabellingView.ftl"/>
  <description>An automatic labelling plugin.</description>
</buildCompleteAction>
```

RELATED TOPICS

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
    - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
    - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
- Web Resources
- Bamboo REST APIs
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
  - Comparing the Bamboo Remote API to the Bamboo REST APIs
- Bamboo Remote API
  - Build Results Filters
  - Other Services
  - Authentication Services
  - Elastic Bamboo Services
  - Build Results Services
  - Build Services
- Bamboo Developer FAQ
  - How do I inject managers into my plugin?
How do I search for previous build result?
How do I start a build programatically?
How do I trigger off a build from my action?

Bamboo Plugin Tutorial
- Tutorial 1 - Getting Started with a Simple Post Build Labeller
- Tutorial 2 - Configurable Regex Labeller

Changes by Version
- Changes for Bamboo 2.0
  - Changes to Bamboo's Configuration UI require the ConfigurablePlugin
  - Post-Build Processing in Bamboo 2.0
  - Build Process for 2.0
  - Repository plugin changes in 2.0
  - Updates to the build processing plugins
- Changes for Bamboo 2.1
- Changes for Bamboo 2.1.5
- Changes for Bamboo 2.2
- Changes for Bamboo 2.3

Building a Bamboo War Distribution From Source
Setting up Bamboo Development Environment in IDEA
Building Bamboo In IDEA - Troubleshooting

Bamboo Documentation Home

Index Reader Module

Description
Written in conjunction with Post Build Index Writer Module, the IndexReader will translate the fields in the index and re-insert the information into a BuildResultSummary object, which has a specially designated customBuildData map for this purpose.

Interface
Index Reader modules must implement the com.atlassian.bamboo.index.CustomIndexReader interface.

Sample Module Descriptor Element

```xml
<indexReader key="cloverIndexReader" name="Reads Clover result values from index"
    class="com.atlassian.bamboo.builder.coverage.CloverIndexReader">
  <description>Reads the clover result from an index document and populates into build result summary</description>
</indexReader>
```

RELATED TOPICS
- Bamboo Plugin Guide
- Bamboo Plugin Module Types
  - Bamboo Event Listener Module
  - Build Agent Requirement Filter Module
  - Command Decorator Module
  - Notification Recipient Module
  - Notification Type Module
    - Building a Notification Plugin
  - Builder Plugin Module
  - Source Repository Module
    - Repository Extensions
  - Trigger Reason Module
  - Pre Build Queued Action
  - Pre Build Action Module
  - Build Processor Module
  - Build Processor Server Module
  - Build Complete Action Module
  - Index Reader Module
  - Post Build Index Writer Module
  - Report Module
  - XWork Plugin Module
  - Web Item Module
  - Web Section Module
  - Notification Condition Module
- Bamboo's Build Process
Bamboo Documentation Home

Post Build Index Writer Module

Description

The PostBuildIndexWriter allows you to write your custom data for a build into the index, which allows for future retrieval in your custom Report Module. The PostBuildIndexWriter will be invoked in three places in Bamboo: when a build completes and it indexes, operations which requires a re-index of a particular build (result), and when you run the re-index all action under the Administration tab.

The PostBuildIndexWriter should always be written in conjunction with a Index Reader Module which will be able to retrieve the data in the index.

Interface

Post Build Index Writer modules must implement the com.atlassian.bamboo.index.CustomPostBuildIndexWriter interface.

Sample Module Descriptor Element

```xml
<postBuildIndexWriter
    key="cloverIndexWriter"
    name="Write Clover Result to Index"
    class="com.atlassian.bamboo.builder.coverage.CloverPostBuildIndexWriter">
  <description>
    Writes the clover result in a build results to an index document
  </description>
</postBuildIndexWriter>
```

RELATED TOPICS

- Bamboo Plugin Guide
- Bamboo Plugin Module Types
  - Bamboo Event Listener Module
Report Module

Description

This defines a report module. A report module will appear under the Reports tab.

A report typically consists of two objects:

- A ReportCollector object implementing the `com.atlassian.bamboo.reports.collector.ReportCollector` interface. This
takes in a list of builds and generates a **DataSet**.

- A **ReportLineChart** object extending the `com.atlassian.bamboo.reports.charts.BambooReportLineChart` class. This chart will be responsible for rendering the dataset results generated by the ReportCollector. Charts in Bamboo are generated via **jFreeChart**.

**Sample Module Descriptor Element**

```xml
<report key="ratioOfSuccess" name="Percentage of Successful Builds" class="com.atlassian.bamboo.reports.collector.RatioOfSuccessCollector">
  <description>
    Comparing success percentages gives you an idea of how stable a build is compared to one another.
    100% means your build is always rock solid. 0% means something is seriously wrong.
  </description>
  <chartClass>com.atlassian.bamboo.reports.charts.BuildSummarySuccessRatioLineChart</chartClass>
</report>
```

**RELATED TOPICS**

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
    - Building a Notification Plugin
  - Builder Plugin Module
  - Source Repository Module
  - Repository Extensions
  - Trigger Reason Module
  - Pre Build Queued Action
  - Pre Build Action Module
  - Build Processor Module
  - Build Processor Server Module
  - Build Complete Action Module
  - Index Reader Module
  - Post Build Index Writer Module
  - Report Module
  - XWork Plugin Module
  - Web Item Module
  - Web Section Module
  - Notification Condition Module
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
- Web Resources
- Bamboo REST APIs
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
  - Comparing the Bamboo Remote API to the Bamboo REST APIs
- Bamboo Remote API
  - Build Results Filters
  - Other Services
  - Authentication Services
  - Elastic Bamboo Services
  - Build Results Services
  - Build Services
- Bamboo Developer FAQ
  - How do I inject managers into my plugin?
  - How do I search for previous build result?
  - How do I start a build programatically?
  - How do I trigger off a build from my action?
- Bamboo Plugin Tutorial
  - Tutorial 1 - Getting Started with a Simple Post Build Labeller
  - Tutorial 2 - Configurable Regex Labeller
- Changes by Version
  - Changes for Bamboo 2.0
    - Changes to Bamboo's Configuration UI require the ConfigurablePlugin
XWork Plugin Module

Description

Each XWork module is deployed as a plugin module of type xwork and contains one or more XWork package elements.

Below is an example atlassian-plugin.xml file containing a single XWork module.

The XWork plugin module allows you to define your own XWork package and actions that you can access.

To build the action into the system, you will typically need to add a Web Item Module to link to your action.

Sample Module Descriptor Element

```xml
<xwork key="viewCloverResult" name="View Clover Result">
  <package name="cloverPlugin" extends="buildView">
    <action name="viewCloverResult" class="com.atlassian.bamboo.build.ViewBuildResults">
      <result name="success" type="freemarker">/plugins/clover-plugin/viewCloverResult.ftl</result>
      <result name="error" type="freemarker">/error.ftl</result>
    </action>
  </package>
</xwork>
```

RELATED TOPICS

- Bamboo Plugin Guide
- Bamboo Plugin Module Types
- Bamboo Event Listener Module
- Build Agent Requirement Filter Module
- Command Decorator Module
- Notification Recipient Module
- Notification Type Module
- Building a Notification Plugin
- Builder Plugin Module
- Source Repository Module
- Repository Extensions
- Trigger Reason Module
- Pre Build Queued Action
- Pre Build Action Module
- Build Processor Module
- Build Processor Server Module
- Build Complete Action Module
- Index Reader Module
- Post Build Index Writer Module
- Report Module
- XWork Plugin Module
- Web Item Module
- Web Section Module
- Notification Condition Module
- Bamboo's Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
Web Item Module

Description

The WebItem allows you to define a link in the Bamboo system. (Usually in some form of menu).

Currently, you can use the web-item to add links to three locations:

- The Administration Menu
- The Plan Sub Menu (tabs on the View Plan page)
- The Results Sub Menu (tabs on the View Build Results page)

Sample Module Descriptor Element

```xml
<web-item key="pipelineConfig" name="Build Queues" section="system.admin/builds" weight="20">
  <label key="webitems.system.admin.build.queues"/>
  <link>/admin/configurePipeline!default.action</link>
  <condition class="com.atlassian.plugin.web.conditions.AdminPermissionCondition"/>
</web-item>
```

Module Components

- **key** - this is the unique identifier of the web-item, it is also used by Bamboo to give the link an id.
- **name** - in the plan sub menu and results sub menu this is used to determine if the current link (tab) is active
- **section** - the section is made of of the parent section's location followed buy the name of the parent section. In Bamboo this is used to retrieve the appropriate web-items for the menu. (see Web Section Module)
- **weight** - this is used to determine the order of the items on the page
- **label** - this will be displayed on the screen and can be plain text or a property key
- **link** - the link is the url the link will point to. It can be absolute or relative to Bamboo's context path
- **condition** - by implementing the com.atlassian.plugin.web.Condition class you can add rules to determine whether the link will be displayed or not.
Both the link and the id can make use of parameters passed to the page. For example:
<link>/build/viewBuildFiles.action?buildKey=${buildKey}</link> where ${buildKey} is the parameter name.
Web Section Module

**Description**

The **WebSection** module is used to provide a section or grouping of **Web Item Module**.

Currently, Web Sections are used to group Web Items for the Administration Menu, Plan Sub Menu, Results Sub Menu

**Sample Module Descriptor Element**

```xml
<web-section key="builds" name="Builds" location="system.admin" weight="100">
  <label key="websections.system.admin.build"/>
  <icon height="16" width="16">
    <link>/images/icons/icon_spanner.gif</link>
  </icon>
</web-section>
```

**Notes**

The section is only displayed on the Administration Menu but it is required for all locations as Bamboo uses it to place the web items.

Which menu the item gets placed in is determined by the `location` attribute.

The following table indicates the location attribute required for each menu:

<table>
<thead>
<tr>
<th>Menu</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration Menu</td>
<td>system.admin</td>
</tr>
<tr>
<td>Plan Sub Menu</td>
<td>build.subMenu</td>
</tr>
<tr>
<td>Build Results Sub menu</td>
<td>results.subMenu</td>
</tr>
</tbody>
</table>

**RELATED TOPICS**

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
    - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
    - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
- Bamboo’s Build Process
- Accessing Bamboo Components From Plugin Modules
- Common Bamboo Classes
- Bamboo Persistence using Bandana
- Downloadable Plugin Resources
- Web Resources
- Bamboo REST APIs
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
Notification Condition Module

This plugin point has been deprecated in Bamboo 2.2. Please see the Notification Type Module for more information as well as Building a Notification Plugin.

Description

A plugin module which allows you to define your own notification condition

You can use this plugin to

- Set conditions as to whether or not to send a notification
- Generate customised notification content

Interface

Notification Condition modules must implement the com.atlassian.bamboo.notification.NotificationCondition interface.

Sample Module Descriptor Element

```
<notification-condition key="buildCompleted.XFailedBuilds" name="After X Failed Builds Completed"
class="com.atlassian.bamboo.notification.conditions.AfterXFailedBuildsCondition">
  <description>Send Notification After X Failed Builds</description>
  <resource type="freemarker" name="edit" location="templates/plugins/notifications/afterXFailedEdit.ftl"/>
  <resource type="freemarker" name="view" location="templates/plugins/notifications/afterXFailedView.ftl"/>
</notification-condition>
```

Other Information regarding the NotificationCondition class

Several of the interface methods for this class accept an Event object as a parameter. Current functionality only allows this to be a BuildCompletedEvent. This event will contain the Build object, BuildResults object and BuildResultsSummary object for you to use.
Bamboo's Build Process

Bamboo's build process consists of a series of **BuildTasks**, each responsible for a particular part of the build process. Each build task has the ability to store a **BuildContext**. A **BuildContext** represents the definition of the part of the build process that the build task is responsible for.
The steps of a build process are described below:

1. **Change Detection** *(performed by the server)* — Changes are detected via polling, manual builds, etc. When a new build is required, the change logs are defined for the build and the build number for the new build is determined. The build context is also created.
2. **Build Queued** *(performed by the server)* — The server decides which agents can execute the build and queues the build.
3. **Agent Picks Up Build** *(performed by the agent)* — The capability context is set. The build is removed from queue. The agent begins looping through the build tasks.
4. **Update Source** *(performed by the agent)* — The agent runs the `CheckoutUpdateForBuild` task. The `Repository#retrieveSourceCode` is called. If the `Repository` is `RepositoryEventAware` the appropriate methods will be called. The agent checks if the repository has changed since the last build and clears the source directory if it has.
5. **Prepare for the Build** *(performed by the agent)* — The agent runs the `PrepareBuildTask`. This begins streaming the logs back to the server. The agent also runs the `CustomPreBuildAction` plugin point. From this point on, the build will be saved.
6. **Executes the Build** *(performed by the agent)* — Timer begins. The agent runs the `Builder#executeBuild`. After the `Builder` has been run, all the `CustomBuildProcessors` are run. Timer is stopped.
7. **Results Processing** *(performed by the server)* — The server runs `CustomBuildProcessorServer`, checks if the build has passed and saves and indexes the build result. The server also fires off the `BuildCompletedEvent` and `PostBuildCompletedEvent` events.
8. **On BuildCompletedEvent** *(performed by the server)* — Notifications are sent in this phase.

### Accessing Bamboo Components From Plugin Modules

Bamboo is built around **Spring**, an open-source component framework for Java.

If you are familiar with Spring, then you may only wish to know that Bamboo plugin modules (and their implementing classes) are autowired by name. Thus, if you want to access a Bamboo component from your plugin, just include the appropriate setter method in your implementing class.

If you want to write Bamboo plugins but are unfamiliar with Spring, the rest of this page should give you more than enough information on how to have your plugin interact with Bamboo.

**Interacting with Bamboo**

When you are writing anything but the simplest Bamboo plugin, you will need to interact with the Bamboo application itself in order to retrieve, change or store information. This document describes how this can be done.

**Manager Objects**

At the core of Bamboo is a group of "Manager" objects. For example, the `buildManager` is in charge of Bamboo’s build plans, the `buildResultsSummaryManager` of build results, and so on.

**Dependency Injection**

Traditionally, in a component-based system, components are retrieved from some kind of central repository. For example, in an EJB-based system, you would retrieve the bean from the application server’s JNDI repository.

Bamboo works the other way round. When a plugin module is instantiated, Bamboo determines which components the module needs, and delivers them to it.

Bamboo determines which components a module needs by reflecting on the module’s methods. Any method with a signature that matches a standard JavaBeans-style setter of the same name as a Bamboo component will have that component passed to it when the module is initialised.

So, if your plugin module needs to access the `buildManager`, all you need to do is put the following setter method on your module’s implementing class:

```java
public void setBuildManager(BuildManager buildManager)
{
    this.buildManager = buildManager;
}
```

**Manager Classes**

There are many managers for different areas of functionality in Bamboo. Not all managers are available on both the Server and any agents that might be running. The following table lists some of the more commonly used ones:

<table>
<thead>
<tr>
<th>Manager class</th>
<th>Responsibility</th>
<th>Agent vs Server</th>
<th>Sample methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>JiraServerManager</td>
<td>Jira Server Connection Details</td>
<td>Server</td>
<td>get_defaultJiraServer()</td>
</tr>
</tbody>
</table>
Note that these are all interfaces. The actual implementation will be injected in your class by Spring, if you include the appropriate setter method in your class as described above.

Do not directly use implementations or cast the injected class to a particular implementation. Implementation classes are subject to change across versions without warning. Where possible, interface methods will be marked as deprecated for two major versions before being removed.

**Other classes that can be injected**

There are also many other classes available to be injected into you plugin which may make your life easier.

<table>
<thead>
<tr>
<th>Class</th>
<th>Responsibility</th>
<th>Agent vs Server</th>
<th>Sample methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapabilityContext</td>
<td>Agent/Server Capabilities</td>
<td>Agent and Server</td>
<td>getCapabilitySet()</td>
</tr>
<tr>
<td>AgentContext</td>
<td>Agent Information</td>
<td>Agent</td>
<td>getBuildAgent()</td>
</tr>
</tbody>
</table>

**More information**

- Bamboo Javadocs
- Bamboo Developer FAQ

**Common Bamboo Classes**

This page outlines some of the more commonly used classes in Bamboo plugins and what sort of information you can retrieve from them.

**The BuildContext**

Every build task contains a `BuildContext` object. This class encapsulates how to build a particular plan and its state at a given time. You can make sure you have this available to your plugin by either extending the AbstractBuildTask or including the following code:

```java
privateBuildContextbuildContext;
publicvoidinit(@NotNullBuildContextbuildContext)
{
this.buildContext = buildContext;
}
```

The build context provide you with:

<table>
<thead>
<tr>
<th>TriggerReason</th>
<th>Interface that encapsulates the reason a build was triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildPlanDefinition</td>
<td>All information required to execute a particular build plan</td>
</tr>
<tr>
<td>BuildChanges</td>
<td>Encapsulates the repository changes for the particular build result</td>
</tr>
<tr>
<td>CurrentBuildResult</td>
<td>Subset of the full build result, defines the partial output of a plan being built</td>
</tr>
</tbody>
</table>

**BuildPlanDefinition**

The `BuildPlanDefinition` interface encapsulates all information required to execute a particular build plan. The typical case is how to check out given a vcs revision key, then run the builder command and then collect all the various artifacts.
Repository

The RepositoryV2 knows how to deal with your source code. Each type of repository stores its own configuration data required to checkout the source and check for code changes. Bamboo ships with an SvnRepository, CvsRepository and PerforceRepository.

This is where you find information such as

<table>
<thead>
<tr>
<th>Source code directory</th>
<th>Where the source gets checked out to</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>buildContext.getRepository().getSourceCodeDirectory(planKey);</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Repository urls</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((SvnRepository) buildContext.getBuildPlanDefinition().getRepository()).getRepositoryUrl()</td>
</tr>
</tbody>
</table>

Current Build Results

The CurrentBuildResults represents the state of the build results throughout the build process. There is variety of information stored in this class, however, different information becomes available at different times throughout the build.

Information you might find here:

- Build state
- Build return code
- Test Results
- Customer build data
- Build Errors

Build

The Build represents a Bamboo plan. It contains all the configuration information for the plan, as well as giving you the ability to navigate through the plan’s build results. The easiest way to obtain a build is via the BuildManager which can be injected into your plugin.

```
Build build = buildManager.getBuildByKey(buildContext.getPlanKey());
```

Once you have the build you have access to such things as the

- build definition - contains all the configuration specifics
- build requirements - set of requirements associated for this build to run
- parent and child builds
- build logger

```
BuildLogger buildLogger = build.getBuildLogger();
buildLogger.addBuildLogEntry(...)
```

Build Result

A Build Result is a specific build of a plan e.g BAM-MAIN-567. Currently in Bamboo, build results information is split into two locations. The BuildResultSummary (or ExtendedBuildResultsSummary) is stored in the database and the BuildResult is what is stored as xml in bamboo-home/xml-data/builds/MYBUILDPLAN/results/.
AdministrationConfiguration

The AdministrationConfiguration stores Bamboo’s system wide properties such as the default url, instance name, global vairables (as defined in the admin section of Bamboo) and ‘System Properties’ - which is a map containing any other properties you may want to store for the instance.

You can obtain the administration configuration like so:

```java
public AdministrationConfiguration getAdministrationConfiguration()
{
    return (AdministrationConfiguration) ContainerManager.getComponent("administrationConfiguration");
}
```

AdminErrorHandler

The AdminErrorHandler error handler is responsible for dealing with errors that occur while build builds. Currently these errors are displayed on the dashboard of those who have admin privileges.

You can obtain the error handler like so:

```java
private AdminErrorHandler getAdminErrorHandler()
{
    if (adminErrorHandler == null)
    {
        adminErrorHandler = (AdminErrorHandler) ContainerManager.getComponent("adminErrorHandler");
    }
    return adminErrorHandler;
}
```

Bamboo Persistence using Bandana

Bandana is only available in Bamboo 2.3 or later.

Bandana is an Atlassian framework for persistence which uses XStream to convert arbitrary Java objects into XML for storage. The concepts used in Bandana are very simple:

- Bandana stores data in contexts. In Bamboo, there is one global context, and one context per plan.
- Each context stores key-value pairs. The key is a String and the value can be any Object (it should typically implement Serializable).
- We recommend that you make your objects as simple as possible. Bandana will attempt to persist the entire object and you may have issues changing package names, field names, etc, if you implement complex objects. You can mark fields as ‘transient’ to prevent them from being persisted. Read more about this in the XStream FAQs.

- If you are defining your own type within a plugin, please provide a no argument constructor to avoid class loading issues.

Based on this design, the BandanaManager has methods for storing and retrieving values from a context by key:
• `void setValue(BandanaContext context, String key, Object value)` — store a value against a key in the Bandana context.
• `Object getValue(BandanaContext context, String key)` — get a key’s value from the Bandana context. Returns null if no matching context and key exists.
• `Object getValue(BandanaContext context, String key, boolean lookUp)` — same as above, except if `lookUp` is true and the context is a space context, this method will also check the global context if no matching key is found in the space context.

For plugins, it is recommended to use a key for your Bandana values that includes the full package name of your plugin. For example, an auto-favourites plugin might use a key like `org.acme.bamboo.autofavourites.importantPreference`.

This XML is written to the BANDANA table in the database.

To get access to the BandanaManager from your plugin code, normally you only need to include a private BandanaManager field. A setter method will be called the first time your plugin is called.

```java
public class DefaultBandanaManagerTest extends TestCase {
    public void testBandanaManagerSample() throws Exception {
        // setup
        BandanaManager bandanaManager = new DefaultBandanaManager(new MemoryBandanaPersister());
        Build mockPlan = EasyMock.createMock(Build.class);
        // expectations
        Map<String, Number> config = new HashMap<String, Number>();
        config.put("testvalue", 200);
        EasyMock.replay(mockPlan);
        // execute
        bandanaManager.setValue(PlanAwareBandanaContext.GLOBAL_CONTEXT, "system.bamboo.repository.Hello:mykey", config);
        final Map<String, Number> returnValue = (Map<String, Number>)bandanaManager.getValue(new PlanAwareBandanaContext(mockPlan),
            "system.bamboo.repository.Hello:mykey", false);
        Map<String, Number> perProjectConfig = new HashMap<String, Number>();
        perProjectConfig.put("testvalue", 100);
        bandanaManager.setValue(new PlanAwareBandanaContext(mockPlan), perProjectConfig."
            system.bamboo.repository.Hello:mykey", perProjectConfig);
        final Map<String, Number> returnValue2 = (Map<String, Number>)
            bandanaManager.getValue(new PlanAwareBandanaContext(mockPlan),
                "system.bamboo.repository.Hello:mykey"false);
        Map<String, Number> perProjectConfig = new HashMap<String, Number>();
        perProjectConfig.put("testvalue", 100);
        bandanaManager.setValue(new PlanAwareBandanaContext(mockPlan),
            "system.bamboo.repository.Hello:mykey", perProjectConfig);
        final Map<String, Number> returnValue2 = (Map<String, Number>)
            bandanaManager.getValue(new PlanAwareBandanaContext(mockPlan),
                "system.bamboo.repository.Hello:mykey"false);
        EasyMock.verify(mockPlan);
    }
}
```

Downloadable Plugin Resources

Bamboo plugins may define downloadable resources. If your plugin requires Bamboo to include additional static files such as images, Javascript or CSS, you will need to use downloadable plugin resources to make them available. However, we recommend that you consider defining Javascript and CSS files as Web Resources instead. Web resources are included in the header of a page and can take advantage of caching (i.e. only download a resource if it has changed) and batching (i.e. serve multiple files in one request).

Please see the Web Resources documentation, if you want to include Javascript or CSS files for download from the header of a page.
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.

```
<resource type="download" name="icn_auto_fav.gif" location="icn_auto_fav.gif"/>
```

- 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/autofavourite/icons/"/>
```

- The name and location must both have trailing slashes
- Subdirectories are also exposed, so in the example above, icons/small/icn_auto_fav.gif will be mapped to the resource templates/extra/autofavourite/icons/small/icn_auto_fav.gif

Referring to Downloadable Resources

The URL for a downloadable resource is as follows:
```
{server root}/download/resources/{plugin key}:{module key}/[resource name]
```

(module key) is optional.

For example:
```
http://bamboo.example.com/download/resources/com.atlassian.bamboo.plugin.autofavourite:autofavourite-resources/icn_auto_fav.gif
```

Web Resources

Bamboo plugins may define downloadable resources. If your plugin requires Bamboo to include additional static Javascript or CSS files, you will need to use downloadable web resources to make them available. Web resources are included for download, at the top of the page in the header. Web resources can also take advantage of caching (i.e. only download a resource if it has changed) and batching (i.e. serve multiple files in one request). If you would like to include other static files for download, such as images, please see Downloadable Plugin Resources.

```
This is only available as of Bamboo 2.3 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.

```
<web-resource key="autofavourite-resources" name="Auto-Favourite Resources">
  <resource type="download" name="autofavourite.css" location="autofavourite.css"/>
</web-resource>
```

- Resources must be contained within a <webresource> tag.
- 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 Freemaker or your plugin's Action, you need to refer to a WebResourceManager, and call the requireResource() method. Getting a reference to a WebResourceManager is application specific, but in Bamboo it would be:

```java
${webResourceManager.requireResource {
    "com.atlassian.bamboo.plugin.autofavourite:autofavourite-resources"
}}
```

The URL for a downloadable resource is application specific. For Bamboo it would be follows:

```
{server root}/s/{build num}/v/{plugin version}/s/system counter]/c/download/resources/{plugin key}:{module key}/c/resource name
```

For the above example:

```html
<script type="text/css"
    src="http://bamboo.example.com/s/170/1.0/1/_/download/resources/com.atlassian.bamboo.plugin.autofavourite :autofavourite-resources/autofavourite.js"></script>
```

will be inserted in the header of the page.

**Bamboo REST APIs**

The Bamboo REST APIs are for developers who want to integrate their application with Bamboo and for administrators who want to script interactions with the Bamboo server. The Bamboo REST APIs supersede the deprecated Bamboo Remote API. However, only a subset of the methods from the remote API have been implemented as methods in the REST APIs. For a comparison of methods between the Bamboo remote API and the Bamboo REST APIs, please see Comparing the Bamboo Remote API to the Bamboo REST APIs.

**Introduction to Bamboo's REST APIs**

Bamboo's REST APIs provide access to resources (data entities) via URI paths. To use a REST API, your application will make an HTTP request and parse the response. By default, the response format is XML. If you wish, you can request JSON instead of XML. Currently the HTTP GET method is the only method available in the Bamboo REST API.

Because the REST API is based on open standards, you can use any web development language to access the API.

A common use case would be a build service, which provides information about build results.

Bamboo's REST APIs provide the following capabilities:

- Retrieve a list of projects.
- Retrieve the plans for each project as well as plan details and available actions.
- Retrieve a specific plan as well as plan details and available actions.
- Retrieve the build results for a specific plan in a project.
- Retrieve the artifacts for a build.
- Retrieve all report type definitions as well as links to related images (for report types of specified parameters).

**Getting Started**

If you would like to know more about REST in general, start with the RESTwiki's guide to REST In Plain English.

Then jump right in and try our REST resources:

- Read our guide to using the REST APIs.
- Find the REST resources you need in our REST resources reference guide.

**Advanced Topics**

Below are some links to in-depth information on development of REST APIs and plugins:

- **Developing your own REST APIs for Bamboo**: Bamboo uses the Atlassian REST plugin to implement the Bamboo APIs. The REST plugin is bundled with Bamboo. You can add your own REST APIs to Bamboo by creating a Bamboo plugin that includes the REST
**Understanding the principles behind the Atlassian REST API design:** You may be interested in the guidelines followed by the Atlassian developers who are designing REST APIs for Atlassian applications, including the Bamboo REST APIs.

**Related Topics**

Bamboo Development Hub

**Bamboo REST Resources**

The Bamboo REST APIs allow you to address the Bamboo data entities as 'resources'. This means that they are identified by URIs and operated on by HTTP requests, such as GET. Whenever you GET one of these resources, you receive a representation encoded using XML or JSON.

The REST API usage guide describes common factors in the APIs, including the full structure of the URIs, API versions, etc. Below are details of the resources made available by the APIs.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<resources>
  <resources-expand=
  <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/" />
  <resources size="5" max-result="5" start-index="0">
    <resource name="project">
      <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/project" />
    </resource>
    <resource name="plan">
      <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/plan" />
    </resource>
    <resource name="build">
      <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/build" />
    </resource>
    <resource name="chart">
      <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/chart" />
    </resource>
    <resource name="metadata">
      <link rel="self" href="http://myhost:8080/bamboo/rest/api/latest/" />
    </resource>
  </resources>
</resources>
```

On this page:

- General Information
- Project Service
- Plan Service
- Build Service - All Builds
- Build Service - Specific Project
- Build Service - Specific Plan
- Build Service - Specific Build Result
- Chart Service
- Report Service
- Build Queue Service
- Comments Service
- Labels Service
- Server Information Service

**General Information**

You can always view the root level resources for a particular version of a Bamboo REST API by navigating to the root URI.

E.g.

```
http://myhost.com:8085/bamboo/rest/api/latest/
```

The response of a REST call will always contains information about size of whole collection (size attribute) as well as the start-index and...
The number of resources in returned lists is limited to 25, even if you specify max-result to be greater than 25.

### Project Service

<table>
<thead>
<tr>
<th>URI</th>
<th>/project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Lists all the projects set up on the Bamboo server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>URI Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ?expand=projects or no parameter — List projects with projects list expanded and links to projects available.</td>
</tr>
<tr>
<td>• ?expand=projects.project — List projects with projects list expanded and project details visible.</td>
</tr>
<tr>
<td>• ?expand=projects.project.plans — List of projects and plans belonging to the project with plans expanded and links to plans available.</td>
</tr>
<tr>
<td>• ?expand=projects.project.plans.plan — List of projects and plans belonging to the project with plans expanded and plan details visible.</td>
</tr>
<tr>
<td>• ?expand=projects.project.plans.plan.actions — List of projects and plans belonging to the project with plans expanded, and plan details and actions visible.</td>
</tr>
<tr>
<td>• ?favourite&amp;expand=projects.project.plans — List of projects and favourite plans belonging to the project with plans expanded and links to plans available. Plans which are not favourites will not be returned.</td>
</tr>
<tr>
<td>• ?favourite&amp;expand=projects.project.plans.plan — List of projects and favourite plans belonging to the project with plans expanded and plan details visible. Plans which are not favourites will not be returned.</td>
</tr>
<tr>
<td>• ?favourite&amp;expand=projects.project.plans.plan.actions — List of projects and favourite plans belonging to the project with plans expanded, and plan details and actions visible. Plans which are not favourites will not be returned.</td>
</tr>
<tr>
<td>• ?cloverEnabled&amp;expand=projects.project.plans — List of projects and Clover-enabled plans belonging to the project with plans expanded and links to plans available. Plans which are not favourites will not be returned.</td>
</tr>
<tr>
<td>• ?cloverEnabled&amp;expand=projects.project.plans.plan — List of projects and Clover-enabled plans belonging to the project with plans expanded and plan details visible. Plans which are not favourites will not be returned.</td>
</tr>
<tr>
<td>• ?cloverEnabled&amp;expand=projects.project.plans.plan.actions — List of projects and Clover-enabled plans belonging to the project with plans expanded, and plan details and actions visible. Plans which are not favourites will not be returned.</td>
</tr>
</tbody>
</table>

1 A plan is Clover-enabled if Clover output will be produced by the build (See specifying the builder for a plan).

### HTTP Methods

- GET — Returns a list of projects.

#### Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <projects expand="projects">
  - <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/project" />
  - <project name="MyProject1" key="MYPROJ1">
    - <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/project/MYPROJ1" />
  </project>
  - <project name="MyProject2" key="MYPROJ2">
    - <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/project/MYPROJ2" />
  </project>
</projects>
```

#### Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>projects/project</td>
<td>• name — Name of the project</td>
</tr>
<tr>
<td></td>
<td>• key — Project key</td>
</tr>
<tr>
<td>projects/project/link</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the project</td>
</tr>
</tbody>
</table>
Plan Service

**URI**
/plan

**Description**
Lists all the plans set up on the Bamboo server.

**URI Parameters**
- `?expand=plans` or no parameter — List all plans for all projects with list expanded (but no plan details).
- `?expand=plans.plan` — List all plans for all projects with list expanded and plan details visible.
- `?expand=plans.plan.actions` — List all plans for all projects with list expanded and plan details and plan actions visible.
- `?favourite&expand=plans.plan` — List all favourite plans for all projects with list expanded and plan details visible. Plans which are not favourites will not be returned.
- `?favourite&expand=plans.plan.actions` — List all favourite plans for all projects with list expanded and plan details and plan actions visible. Plans which are not favourites will not be returned.
- `?cloverEnabled&expand=plans.plan` — List all Clover-enabled plans plans for all projects with list expanded and plan details visible. Plans which are not favourites will not be returned.
- `?cloverEnabled&expand=plans.plan.actions` — List all Clover-enabled plans plans for all projects with list expanded and plan details and plan actions visible. Plans which are not favourites will not be returned.

1 A plan is Clover-enabled if Clover output will be produced by the build (See specifying the builder for a plan).

**HTTP Methods**
- GET — Returns a list of plans.

Example of response JSON:

```json
[
  {
    "expand":"projects","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/project"},
    "projects":[
      {
        "expand": "project","size":2,"max-result":2,"start-index":0,"project":{
          "name": "MyProject1","key": "MYPROJ1","link": {"rel": "self","href": "http://panda :8085/bamboo/rest/api/latest/project/MYPROJ1"},
          "name": "MyProject2","key": "MYPROJ2","link": {"rel": "self","href": "http://panda :8085/bamboo/rest/api/latest/project/MYPROJ2"}
        }
      }
    ]
  }
]
```

Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<plans expand="plans">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/plan"/>
  <plan enabled="true" plan name="My Plan 1" key="MYPLAN1">
    <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN1"/>
  </plan>
  <plan enabled="true" name="My Plan 2" key="MYPLAN2">
    <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN2"/>
  </plan>
  <plan enabled="true" name="My Plan 3" key="MYPLAN3">
    <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN3"/>
  </plan>
</plans>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| plans/plan   | • name — Name of the plan  
|              | • key — Plan key                       |
| plans/plan/link | • rel — Relationship between link and element (defaults to "self")  
|              | • href — URL for the plan              |
Build Service — All Builds

<table>
<thead>
<tr>
<th>URI</th>
<th>/build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Lists the latest builds for all plans on the Bamboo server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>URI Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ?expand=builds or no parameter — Lists the latest builds for each of the plans on the Bamboo server, in descending order (by completion time).</td>
</tr>
<tr>
<td>• ?expand=builds.build — Lists the latest builds for each of the plans on the Bamboo server with build details expanded, in descending order (by completion time) (Warning: This can be a very resource-intensive operation. You may wish to consider limiting the number of build results returned).</td>
</tr>
<tr>
<td>• ?expand=builds[0:N].build — Lists the last N builds for the plans on the Bamboo server with build result details expanded, in descending order (by completion time).</td>
</tr>
</tbody>
</table>

**Modifying the build array parameters**

You can modify the build array parameters (i.e. [0:N]) to return different sets of results.

* Use [0] to return the latest build result.
* Use negative indexes to return the last entries in build list, e.g. [-5:-1] returns last five builds in list.
* Use open ended arrays to return all results from start or end of builds list, e.g. [1:] returns all builds from first to the list limit, [5:] returns first 6 entries in build list.

• ?favourite — List the latest builds for all favourite plans on the Bamboo server.
• ?label=labelname — List the latest builds which have a build result labelled with a particular label(s). labelname can be the name of a single label (e.g. deploy_failed) or a comma-separated list of label names (e.g. deploy_failed,deploy_successful)
• ?issueKey=issuekey — List the latest builds which have a build result related to one or many JIRA issues. issuekey can be the key of a single JIRA issue (e.g. BAM-1111) or a comma-separated list of label names (e.g. BAM-1111,BAM-1112,BAM-1113)
• ?cloverEnabled — List the latest builds all Clover-enabled plans plans on the Bamboo server.

1 A plan is Clover-enabled if Clover output will be produced by the build (See specifying the builder for a plan).

• ?expand=builds.build.artifacts — Lists the artifacts produced by each of the latest builds for all plans on the Bamboo server.
• ?expand=builds.build.labels — Lists the labels associated with each of the latest builds for all plans on the Bamboo server.
• ?expand=builds.build.comments — Lists the comments associated with each of the latest builds for all plans on the Bamboo server.
• ?expand=builds.build.jiraIssues — Lists the JIRA issues associated with each of the latest builds for all plans on the Bamboo server.

Example of response JSON:

```
{
"expand":"plans","link":{
"rel":"self","href":http://myhost:8085/bamboo/rest/api/latest/plan"},
"plans":{
"expand":"plan","size":2,"max-result":2,"start-index":0,"plan":[
{
"name":"My Plan 1","key":"MYPLAN1","link":{
"rel":"self","href":http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN1"}
},
{
"name":"My Plan 2","key":"MYPLAN2","link":{
"rel":"self","href":http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN2"}
},
{
"name":"My Plan 3","key":"MYPLAN3","link":{
"rel":"self","href":http://myhost:8085/bamboo/rest/api/latest/plan/MYPLAN3"}
]}
}
```
Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>builds/build</td>
<td>number — Build number</td>
</tr>
<tr>
<td></td>
<td>state — State of the build (e.g. Successful, Failed)</td>
</tr>
<tr>
<td></td>
<td>key — Key of the build result</td>
</tr>
<tr>
<td>builds/build/link</td>
<td>rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>href — URL for the build result</td>
</tr>
<tr>
<td>build/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>build/buildCompletedTime</td>
<td>Description: Finish time of build</td>
</tr>
<tr>
<td>build/buildDurationInSeconds</td>
<td>Description: Build duration in seconds</td>
</tr>
<tr>
<td>build/buildDurationDescription</td>
<td>Description: Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>build/buildRelativeTime</td>
<td>Description: Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>build/queueStartedTime</td>
<td>Description: Time when build was queued</td>
</tr>
<tr>
<td>build/queueTimeInSeconds</td>
<td>Description: How long build was queued</td>
</tr>
<tr>
<td>build/vcsUpdateStartedTime</td>
<td>Description: Time when VCS update occurred</td>
</tr>
<tr>
<td>build/vcsUpdateInSeconds</td>
<td>Description: How long VCS took to complete</td>
</tr>
<tr>
<td>build/buildTestSummary</td>
<td>Description: Summary of test results</td>
</tr>
<tr>
<td>build/successfulTestCount</td>
<td>Description: Number of successful tests in build</td>
</tr>
<tr>
<td>build/failedTestCount</td>
<td>Description: Number of failed tests in build</td>
</tr>
<tr>
<td>build/buildReason</td>
<td>Description: Reason for build</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/name</td>
<td>Description: Name of the build artifact</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/link</td>
<td>rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>href — URL for the build artifact</td>
</tr>
<tr>
<td>Builds/build/labels/label/name</td>
<td>Description: Name of the label</td>
</tr>
</tbody>
</table>
Example of response JSON:

```json
{
  "expand": "builds",
  "link": {
    "rel": "self",
    "href": "http://myhost:8085/bamboo/rest/api/latest/build"},
  "builds": [
    {
      "number": ,
      "state": "Successful",
      "key": "MYPROJ1-MYPLAN1",
      "link": {
        "rel": "self",
        "href": "http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-2"},
      "number": ,
      "state": "Successful",
      "key": "MYPROJ1-MYPLAN2",
      "link": {
        "rel": "self",
        "href": "http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN2-5"},
      "number": ,
      "state": "Successful",
      "key": "MYPROJ1-MYPLAN3",
      "link": {
        "rel": "self",
        "href": "http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN3-4"},
      "number": ,
      "state": "Successful",
      "key": "MYPROJ2-MYPLAN4",
      "link": {
        "rel": "self",
        "href": "http://panda:8085/bamboo/rest/api/latest/build/MYPROJ2-MYPLAN4-5"},
      "number": ,
      "state": "Successful",
      "key": "MYPROJ3-MYPLAN5",
      "link": {
        "rel": "self",
        "href": "http://panda:8085/bamboo/rest/api/latest/build/MYPROJ3-MYPLAN5-5"}
    }
  ]
}
```
URI Parameters

- `?expand=builds` or no parameter — Lists the latest builds for each of the plans of a project on the Bamboo server, in descending order (by completion time).
- `?expand=builds.build` — Lists the latest builds for each of the plans of a project with build details expanded, in descending order (by completion time) (Warning: This can be a very resource-intensive operation. You may wish to consider limiting the number of build results returned).
- `?expand=builds[0:N].build` — Lists the last N builds for the plans of the project with build result details expanded, in descending order (by completion time).

✅ Modifying the build array parameters
You can modify the build array parameters (i.e. [0:N]) to return different sets of results.
* Use [0] to return the latest build result.
* Use negative indexes to return the last entries in build list, e.g. [5-1] returns last five builds in list
* Use open ended arrays to return all results from start or end of builds list, e.g. [1:] returns all builds from first to the list limit, [5:] returns first 6 entries in build list.

- `?favourite` — List the latest builds for all favourite plans for a specific project on the Bamboo server.
- `?label=labelname` — List the latest builds for a specific project, which have a build result labelled with a particular label(s). labelname can be the name of a single label (e.g. deploy_failed) or a comma-separated list of label names (e.g. [deploy_failed, deploy_successful])
- `?issueKey=issuekey` — List the latest builds for a specific project, which have a build result related to one or many JIRA issues. issuekey can be the key of a single JIRA issue (e.g. BAM-1111) or a comma-separated list of label names (e.g. [BAM-1111, BAM-1112, BAM-1113])
- `?cloverEnabled` — List the latest builds for a specific project, all Clover-enabled plans on the Bamboo server.
  * A plan is Clover-enabled if Clover output will be produced by the build (See specifying the builder for a plan).
- `?expand=builds.build.artifacts` — Lists the artifacts produced by each of the latest builds for all plans for a specific project.
- `?expand=builds.build.labels` — Lists the labels associated with each of the latest builds for all plans for a specific project.
- `?expand=builds.build.comments` — Lists the comments associated with each of the latest builds for all plans for a specific project.
- `?expand=builds.build.jiraIssues` — Lists the JIRA issues associated with each of the latest builds for all plans for a specific project.

Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <builds expand="builds">
  - <build number="2" state="Successful" key="MYPROJ1-MYPLAN1-2">
  </build>
  - <build number="5" state="Successful" key="MYPROJ1-MYPLAN2-5">
    <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN2-5"/>
  </build>
  - <build number="4" state="Successful" key="MYPROJ1-MYPLAN3-4">
    <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN3-4"/>
  </build>
</builds>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>builds/build</td>
<td>• number — Build number&lt;br&gt;• state — State of the build (e.g. Successful, Failed)&lt;br&gt;• key — Key of the build result</td>
</tr>
<tr>
<td>builds/build/link</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)&lt;br&gt;• href — URL for the build result</td>
</tr>
<tr>
<td>build/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>build/buildCompletedTime</td>
<td>Finish time of build</td>
</tr>
<tr>
<td>build/buildDurationInSeconds</td>
<td>Build duration in seconds</td>
</tr>
<tr>
<td>build/buildDurationDescription</td>
<td>Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>build/buildRelativeTime</td>
<td>Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>build/queueStartedTime</td>
<td>Time when build was queued</td>
</tr>
<tr>
<td>build/queueTimeInSeconds</td>
<td>How long build was queued</td>
</tr>
<tr>
<td>build/vcsUpdateStartedTime</td>
<td>Time when VCS update occurred</td>
</tr>
<tr>
<td>build/vcsUpdateInSeconds</td>
<td>How long VCS took to complete</td>
</tr>
<tr>
<td>build/buildTestSummary</td>
<td>Summary of test results</td>
</tr>
<tr>
<td>build/successfulTestCount</td>
<td>Number of successful tests in build</td>
</tr>
<tr>
<td>build/failedTestCount</td>
<td>Number of failed tests in build</td>
</tr>
<tr>
<td>build/buildReason</td>
<td>Reason for build</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/name</td>
<td>Name of the build artifact</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/link</td>
<td>- rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>- href — URL for the build artifact</td>
</tr>
<tr>
<td>builds/build/labels/label/name</td>
<td>Name of the label</td>
</tr>
<tr>
<td>builds/build/comments/comment</td>
<td>Author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/content</td>
<td>Author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/creationDate</td>
<td>Date that comment was created</td>
</tr>
<tr>
<td>builds/build/comments/comment/modificationDate</td>
<td>Date that comment was last updated</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue</td>
<td>Key — Key of the JIRA issue</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue/url</td>
<td>- rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>- href — URL for the JIRA issue</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{"expand":"builds","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/build"},
 "builds":["expand":"build","size":3,"max-result":3,"start-index":0,"build":{
 "number":0,"state":"Successful","key":"MYPROJ1-MYPLAN1","link":{"rel":"self","href":"http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-2"},
 {"number":1,"state":"Successful","key":"MYPROJ1-MYPLAN2","link":{"rel":"self","href":"http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN2-5"}},
 {"number":2,"state":"Successful","key":"MYPROJ1-MYPLAN3","link":{"rel":"self","href":"http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN3-4"}}
 ]}
```
<table>
<thead>
<tr>
<th>Description</th>
<th>Provides information about builds executed for a build plan from latest to the first build</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI</strong></td>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds or no parameter — List all build results for the build plan (build details not visible).</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build — List all build results for the build plan with build details expanded, in descending order. (Warning: This is a very resource-intensive operation. You should consider limiting the number of builds returned).</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds[0:N].build — List the last N build results for the build plan with build details expanded, in descending order.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Modifying the build array parameters" /> You can modify the build array parameters (i.e. [0:N]) to return different sets of results.</td>
</tr>
<tr>
<td></td>
<td>* Use [0] to return the latest build result.</td>
</tr>
<tr>
<td></td>
<td>* Use negative indexes to return the last entries in build list, e.g. [-5:-1] returns last five builds in list</td>
</tr>
<tr>
<td></td>
<td>* Use open ended arrays to return all results from start or end of builds list, e.g. [1:] returns all builds from first to the list limit, [5] returns first 6 entries in build list.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build.artifacts — List all build results for the build plan with build details expanded and artifacts displayed, in descending order. (Warning: This is a very resource-intensive operation. You should consider limiting the number of builds returned).</td>
</tr>
<tr>
<td></td>
<td>• ?expand=plan — Displays the build plan information (expanded), including a summary of build results information.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=plan.actions — Displays the build plan information (expanded), including a summary of build results information and available actions.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build.artifacts — Lists the artifacts produced by each of the latest builds for the specified plan.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build.labels — Lists the labels associated with each of the latest builds for the specified plan.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build.comments — Lists the comments associated with each of the latest builds for the specified plan.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=builds.build.jiraIssues — Lists the JIRA issues associated with each of the latest builds for the specified plan.</td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td></td>
<td>• GET — Returns a list of plans.</td>
</tr>
</tbody>
</table>

Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<builds expand="builds">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build"/>
  - <builds expand="builds" size="2" max-result="2" start-index="0">
    - <build number="1" state="Successful" key="MYPROJ1-MYPLAN6-1">
        <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN6-1"/>  
    </build>
    <build number="2" state="Successful" key="MYPROJ1-MYPLAN6-2">
        <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN6-2"/>
    </build>
  </builds>
</builds>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| builds/build        | • number — Build number  
|                     | • state — State of the build (e.g. Successful, Failed)  
|                     | • key — Key of the build result                                           |
| builds/build/link   | • rel — Relationship between link and element (defaults to "self") 
<p>|                     | • href — URL for the build result                                         |
| build/buildStartedTime | Description: Start time of build                                         |
| build/buildCompletedTime | Description: Finish time of build                                       |
| build/buildDurationInSeconds | Description: Build duration in seconds                                   |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>build/buildDurationDescription</td>
<td>Description: Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>build/buildRelativeTime</td>
<td>Description: Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>build/queueStartedTime</td>
<td>Description: Time when build was queued</td>
</tr>
<tr>
<td>build/queueTimeInSeconds</td>
<td>Description: How long build was queued</td>
</tr>
<tr>
<td>build/vcsUpdateStartedTime</td>
<td>Description: Time when VCS update occurred</td>
</tr>
<tr>
<td>build/vcsUpdateInSeconds</td>
<td>Description: How long VCS took to complete</td>
</tr>
<tr>
<td>build/buildTestSummary</td>
<td>Description: Summary of test results</td>
</tr>
<tr>
<td>build/successfulTestCount</td>
<td>Description: Number of successful tests in build</td>
</tr>
<tr>
<td>build/failedTestCount</td>
<td>Description: Number of failed tests in build</td>
</tr>
<tr>
<td>build/buildReason</td>
<td>Description: Reason for build</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/name</td>
<td>Description: Name of the build artifact</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/link</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the build artifact</td>
</tr>
<tr>
<td>builds/build/labels/label/name</td>
<td>Description: Name of the label</td>
</tr>
<tr>
<td>builds/build/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/content</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/creationDate</td>
<td>Description: Date that comment was created</td>
</tr>
<tr>
<td>builds/build/comments/comment/modificationDate</td>
<td>Description: Date that comment was last updated</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue</td>
<td>• key — Key of the JIRA issue</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue/url</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the JIRA issue</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{
"expand":"builds","link":{
"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/build"},
"builds":{
"expand":"build","size":2,"max-result":2,"start-index":0,"build":[
{"number":,"state":"Successful","key":"MYPROJ1-MYPLAN6","link":{
"rel":"self","href":"http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN6-1"},
{"number":,"state":"Successful","key":"MYPROJ1-MYPLAN6","link":{
"rel":"self","href":"http://panda:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN6-1"}}
]}
```
### URI Parameters

- no parameter — Displays the latest build result for the plan.
- ?expand=artifacts — Displays the latest build result for the plan and an expanded list of artifacts.
- ?expand=tests — Displays the latest build result for the plan and list of tests.
  - /successful?expand=tests — Displays the latest build result for the plan and list of successful tests.
  - /failed?expand=tests — Displays the latest build result for the plan and list of failed tests.
- ?expand=changes — Displays the latest build result for the plan and list of changes for build.
- ?expand=builds.build.artifacts — Lists the artifacts for the specified build result.
- ?expand=builds.build.labels — Lists the labels for the specified build result.
- ?expand=builds.build.comments — Lists the comments for the specified build result.
- ?expand=builds.build.jiraIssues — Lists the JIRA issues for the specified build result.
- ?expand=changes — Lists the changesets for the specified build result.
- ?expand=changes.change — Lists the changesets (changeset details expanded) for the specified build result.
- ?expand=changes.change.files — Lists the changesets (changeset details and file details expanded) for the specified build result.
- ?expand=testResults — Lists the test results for the specified build result.
- ?expand=testResults.failed — Lists the failed test results for the specified build result.
- ?expand=testResults.failed.testResult — Lists the failed test results (test result details expanded) for the specified build result.
- testResults.failed.testResult.errors — Lists the failed tests (test result details and errors expanded) for the specified build result.
- ?expand=metadata — Lists the metadata for the specified build result.

### HTTP Methods

- GET — Returns the latest build result.

### Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<build number="5157" state="Successful" key="MYPLAN1-5157" expand="artifacts">
  <failedTestCount>0</failedTestCount>
  <successfulTestCount>1617</successfulTestCount>
  <buildStartedTime>2009-06-15T14:56:06.821+1000</buildStartedTime>
  <buildCompletedTime>2009-06-15T15:14:10.994+1000</buildCompletedTime>
  <buildDurationInSeconds>1084.0</buildDurationInSeconds>
  <buildDurationDescription>18 minutes</buildDurationDescription>
  <buildRelativeTime>1 hour ago</buildRelativeTime>
  <buildTestSummary>1617 passed</buildTestSummary>
  <buildReason><a href="http://myhost:8085/bamboo/browse/MYPLAN1-5157/commit">Updated by Austin Powers</a></buildReason>
  <queueStartedTime>2009-06-15T14:53:54.499+1000</queueStartedTime>
  <queueTimeInSeconds>1.0</queueTimeInSeconds>
  <vcsUpdateStartedTime>2009-06-15T14:53:55.546+1000</vcsUpdateStartedTime>
  <vcsUpdateInSeconds>131.0</vcsUpdateInSeconds>
  <artifacts size="1" max-result="1" start-index="0" />
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPLAN1-5157" />
</build>
```

### Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>number — number of the build result&lt;br&gt;state — state of the build result&lt;br&gt;key — key for the build result&lt;br&gt;expand — elements to expand in response</td>
</tr>
<tr>
<td>link</td>
<td>rel — relationship between link and element (defaults to &quot;self&quot;)&lt;br&gt;href — URL for the build result</td>
</tr>
<tr>
<td>build/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>build/buildCompletedTime</td>
<td>Description: Finish time of build</td>
</tr>
<tr>
<td>build/buildDurationInSeconds</td>
<td>Description: Build duration in seconds</td>
</tr>
<tr>
<td>path</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>build/buildDurationDescription</td>
<td>Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>build/buildRelativeTime</td>
<td>Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>build/queueStartedTime</td>
<td>Time when build was queued</td>
</tr>
<tr>
<td>build/queueTimeInSeconds</td>
<td>How long build was queued</td>
</tr>
<tr>
<td>build/vcsUpdateStartedTime</td>
<td>Time when VCS update occurred</td>
</tr>
<tr>
<td>build/vcsUpdateInSeconds</td>
<td>How long VCS took to complete</td>
</tr>
<tr>
<td>build/buildTestSummary</td>
<td>Summary of test results</td>
</tr>
<tr>
<td>build/successfulTestCount</td>
<td>Number of successful tests in build</td>
</tr>
<tr>
<td>build/failedTestCount</td>
<td>Number of failed tests in build</td>
</tr>
<tr>
<td>build/buildReason</td>
<td>Reason for build</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/name</td>
<td>Name of the build artifact</td>
</tr>
<tr>
<td>builds/build/artifacts/artifact/link</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the build artifact</td>
</tr>
<tr>
<td>builds/build/labels/label/name</td>
<td>Name of the label</td>
</tr>
<tr>
<td>builds/build/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/content</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>builds/build/comments/comment/creationDate</td>
<td>Description: Date that comment was created</td>
</tr>
<tr>
<td>builds/build/comments/comment/modificationDate</td>
<td>Description: Date that comment was last updated</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue</td>
<td>• key — Key of the JIRA issue</td>
</tr>
<tr>
<td>builds/build/jiraIssues/issue/url</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the JIRA issue</td>
</tr>
<tr>
<td>builds/build(changes)/change</td>
<td>• changesetid — Id of the changeset</td>
</tr>
<tr>
<td></td>
<td>• author — Author of the changeset</td>
</tr>
<tr>
<td>builds/build(changes)/change/comment</td>
<td>Description: Comment associated with the changeset</td>
</tr>
<tr>
<td>builds/build(changes)/change/date</td>
<td>Description: Date of the changeset</td>
</tr>
<tr>
<td>builds/build(changes)/change/files/file/name</td>
<td>Description: Name of the file associated with the changeset</td>
</tr>
<tr>
<td>builds/build(changes)/change/files/file/revision</td>
<td>Description: Revision of the file associated with the changeset</td>
</tr>
<tr>
<td>builds/build/testResults/successful</td>
<td>• size — Number of successful tests</td>
</tr>
<tr>
<td>builds/build/testResults/failed</td>
<td>• size — Number of failed tests</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult status</td>
<td>status — status of test, i.e. “failed”</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/methodName</td>
<td>methodName — Method associated with failed test</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/className</td>
<td>className — Class associated with failed test</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/duration</td>
<td>Description: Duration of test</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/durationInSeconds</td>
<td>Description: Duration of test in seconds</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/errors</td>
<td></td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/errors/error/message</td>
<td>size — Number of errors in test</td>
</tr>
<tr>
<td>builds/build/testResults/failed/testResult/errors/error/message</td>
<td>Description: Error message for error in failed test.</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{
    "number":5157,
    "state":"Successful",
    "key":"MYPLAN1-5157",
    "expand":"artifacts",
    "failedTestCount":0,
    "successfulTestCount":1617,
    "buildStartedTime":"2009-06-15T14:56:06.821+1000",
    "buildDurationInSeconds":1084.0,
    "buildDurationDescription":"18 minutes",
    "buildRelativeTime":"2 hours ago",
    "buildTestSummary":"1617 passed",
    "buildReason":"
    "queueStartedTime":"2009-06-15T14:53:54.499+1000",
    "queueTimeInSeconds":1.0,
    "vcsUpdateStartedTime":"2009-06-15T14:53:55.546+1000",
    "vcsUpdateInSeconds":131.0,
    "artifacts":{
        "size":1,
        "max-result":1,
        "start-index":0,
    },
    "link":{
        "rel":"self",
        "href":"http://myhost:8085/bamboo/rest/api/latest/build/MYPLAN1-5157"
    }
}
```

**Chart Service**

<table>
<thead>
<tr>
<th>URI</th>
<th>/chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Returns detailed information on a specific chart, or returns the list of reports set up on the Bamboo server.</td>
</tr>
</tbody>
</table>

**URI Parameters**

- **reportKey** — report class as returned by `/chart/reports` resource in key tag
- **buildKeys** — comma-separated list of build keys (PROJECTKEY-PLANKEY) as returned by `/chart/reports` resource
- **groupByPeriod** — period to group results by in the report chart. Valid values = AUTO, DAY, WEEK, MONTH
- **dateFilter** (optional) — date period to filter results by in the report chart. Valid values = LAST_7_DAYS, LAST_30_DAYS, LAST_90_DAYS, ALL, RANGE (requires a valid dateFrom and dateTo as described below)
- **dateFrom** (optional) — start date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE
- **dateTo** (optional) — end date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE
- **width** (optional) — width of the chart in pixels
- **height** (optional) — height of the chart in pixels

**HTTP Methods**

- **GET** — Returns a list of reports.

Example of response XML (list all available reports):

```xml
Chart Service
URI /chart
Description Returns detailed information on a specific chart, or returns the list of reports set up on the Bamboo server.

URI Parameters

- reportKey — report class as returned by `/chart/reports` resource in key tag
- buildKeys — comma-separated list of build keys (PROJECTKEY-PLANKEY) as returned by `/chart/reports` resource
- groupByPeriod — period to group results by in the report chart. Valid values = AUTO, DAY, WEEK, MONTH
- dateFilter (optional) — date period to filter results by in the report chart. Valid values = LAST_7_DAYS, LAST_30_DAYS, LAST_90_DAYS, ALL, RANGE (requires a valid dateFrom and dateTo as described below)
- dateFrom (optional) — start date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE
- dateTo (optional) — end date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE
- width (optional) — width of the chart in pixels
- height (optional) — height of the chart in pixels

HTTP Methods

- GET — Returns a list of reports.
```
<xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<chart>
<location>jfreechart-onetime-3627918147160569458.png</location>
<imageMapName>OZ6yx_map</imageMapName>
<imageMap>
<map id="OZ6yx_map" name = "OZ6yx_map">
<area shape = "poly" coords = "373,45,379,45,379,51,373,51,373,45,373,45" title = "INT-TST on average took 38 seconds for June 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "284,121,290,121,290,127,284,127,284,121,284,121" title = "INT-TST on average took 32 seconds for April 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "199,190,205,190,205,196,199,196,199,190,199,190" title = "INT-TST on average took 27 seconds for February 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "154,14,160,14,160,20,154,20,154,14,154,14" title = "INT-TST on average took 40 seconds for January 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "109,204,115,204,115,210,109,210,109,204,109,204" title = "INT-TST on average took 26 seconds for December 2008" alt = "" nohref = ""/>
<area shape = "poly" coords = "65,210,71,210,71,216,65,216,65,210,65,210" title = "INT-TST on average took 26 seconds for November 2008" alt = "" nohref = ""/>
<area shape = "poly" coords = "379,58,378,60,376,61,374,60,373,58,374,56,376,55,378,54,379,58,379,58" title = "ACC-TST on average took 37 seconds for June 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "334,107,333,109,331,110,329,109,328,107,329,105,331,104,333,105,334,107,334,107" title = "ACC-TST on average took 33 seconds for May 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "205,182,204,184,202,185,200,184,199,182,200,180,202,179,204,180,205,182,205,182" title = "ACC-TST on average took 28 seconds for February 2009" alt = "" nohref = ""/>
<area shape = "poly" coords = "115,212,114,214,212,215,109,214,109,212,109,210,112,209,114,210,115,212,115,212" title = "ACC-TST on average took 28 seconds for December 2008" alt = "" nohref = ""/>
<area shape = "poly" coords = "71,195,70,197,68,198,66,197,65,195,66,193,68,192,70,193,71,195,71,195" title = "ACC-TST on average took 27 seconds for November 2008" alt = "" nohref = ""/>
</map>
</imageMap>
</chart>
</xml>
Returns the list of reports set up on the Bamboo server.

**URI Parameters**
- no parameter — List all available reports
- ?expand=reports — List all available reports with report details expanded.

**HTTP Methods**
- GET — Returns a list of reports.

Example of response XML (list all available reports):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<reports expand="reports">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/chart/reports" />
  <reports size="7" max-result="7" start-index="0">
    <report>
      <key>com.atlassian.bamboo.plugin.system.reports:numberOfBuilds</key>
      <name>Build Activity</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.system.reports:averageDuration</key>
      <name>Build Duration</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.system.reports:averageQueuedDuration</key>
      <name>Build Queued Duration</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.myproject1:myproject1Coverage</key>
      <name>My Project 1 Code Coverage</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.myproject1:myproject1LinesOfCode</key>
      <name>My Project 1 Lines of Code</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.system.reports:numberOfFailures</key>
      <name>Number of Build Failures</name>
    </report>
    <report>
      <key>com.atlassian.bamboo.plugin.system.reports:numberOfTests</key>
      <name>Number of Tests</name>
    </report>
  </reports>
</reports>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| report  | - key — key of the Bamboo report  
           - name — name of the Bamboo report |

Example of response JSON (list all available reports):

```json
{
  "expand": "reports",
  "link": {
    "rel": "self",
    "href": "http://myhost:8085/bamboo/rest/api/latest/chart/reports"
  },
  "reports": {
    "size": 7,
    "max-result": 7,
    "start-index": 0,
    "report": [
      {
        "key": "com.atlassian.bamboo.plugin.system.reports:numberOfBuilds",
        "name": "Build Activity"
      },
      {
        "key": "com.atlassian.bamboo.plugin.system.reports:averageDuration",
        "name": "Build Duration"
      },
      {
        "key": "com.atlassian.bamboo.plugin.system.reports:averageQueuedDuration",
        "name": "Build Queued Duration"
      },
      {
        "key": "com.atlassian.bamboo.plugin.myproject1:myproject1Coverage",
        "name": "My Project 1 Code Coverage"
      },
      {
        "key": "com.atlassian.bamboo.plugin.myproject1:myproject1LinesOfCode",
        "name": "My Project 1 Lines of Code"
      },
      {
        "key": "com.atlassian.bamboo.plugin.system.reports:numberOfFailures",
        "name": "Number of Build Failures"
      },
      {
        "key": "com.atlassian.bamboo.plugin.system.reports:numberOfTests",
        "name": "Number of Tests"
      }
    ]
  }
}
```
Build Queue Service

<table>
<thead>
<tr>
<th><strong>URI</strong></th>
<th>/queue</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Lists all the builds waiting in the build queue, adds or removes a build from the build queue.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>URI Parameters</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- no parameter — Displays the number of builds waiting in the build queue.</td>
<td></td>
</tr>
<tr>
<td>- ?expand=queuedBuilds — Lists all the builds waiting in the build queue.</td>
<td></td>
</tr>
<tr>
<td>- ?expand=queuedBuilds.queuedBuild.changes — Lists all the builds waiting in the build queue with associated changes expanded.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>HTTP Methods</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- GET — Returns a list of projects.</td>
<td></td>
</tr>
<tr>
<td>- POST — Adds a build to the build queue (e.g. /queue/MYPROJ1-MYPLAN1). Response will be empty.</td>
<td></td>
</tr>
<tr>
<td>- DELETE — Adds a build to the build queue (e.g. /queue/MYPROJ1-MYPLAN1). Response will be empty.</td>
<td></td>
</tr>
</tbody>
</table>

Example of response XML (retrieving builds in the build queue):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <queue expand="queuedBuilds">
  - <queuedBuilds expand="queuedBuild" size="1" max-result="1" start-index="0">
    - <queuedBuild buildResultKey="MYPROJ1-MYPLAN1-7" buildNumber="7" planKey="MYPROJ1-MYPLAN1" expand="changes">
      - <triggerReason>Manual build</triggerReason>
    </queuedBuild>
  </queuedBuilds>
</queue>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>queue/queuedBuilds</td>
<td>• size — Number of builds in the build queue</td>
</tr>
<tr>
<td>queue/queuedBuilds/queuedBuild</td>
<td>• buildResultKey — Key for the build result</td>
</tr>
<tr>
<td></td>
<td>• buildNumber — Build number</td>
</tr>
<tr>
<td></td>
<td>• planKey — Key for the build plan</td>
</tr>
<tr>
<td>queue/queuedBuilds/queuedBuild/triggerReason</td>
<td>Description: Build trigger reason</td>
</tr>
<tr>
<td>queue/queuedBuilds/queuedBuild/changes</td>
<td>• size — Number of changes associated with build</td>
</tr>
<tr>
<td>queue/queuedBuilds/queuedBuild/changes/change</td>
<td>• changesetId — Id of the changeset</td>
</tr>
<tr>
<td></td>
<td>• author — Author of the changeset</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{"expand":"queuedBuilds","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/queue"},
"queuedBuilds":
  {"expand":"queuedBuild","size":1,"max-result":1,"start-index":0,"queuedBuild":
  {"buildResultKey":"MYPROJ1-MYPLAN1-7","buildNumber":7,"planKey":"MYPROJ1-MYPLAN1","expand":"changes","triggerReason":"Manual build",
  "changes":
  ["expand":"change","size":1,"max-result":1,"start-index":0,"change":
  {"changesetId":98662,"author":"apowers"}]]
}
```
Comments Service

<table>
<thead>
<tr>
<th><strong>URI</strong></th>
<th>/build/PROJECTKEY-PLANKEY-BUILDNUMBER/comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Retrieves the comments for a build result, adds a comment to a build result.</td>
</tr>
<tr>
<td><strong>URI Parameters</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• no parameter — Retrieves the comments for a build result.</td>
</tr>
<tr>
<td></td>
<td>• ?expand=comments.comment — Retrieves the comments for a build result with expanded comment details.</td>
</tr>
<tr>
<td><strong>HTTP Methods</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• GET — Retrieves the comments for a build result.</td>
</tr>
<tr>
<td></td>
<td>• POST — Adds a comment to the build result. Response will be empty.</td>
</tr>
</tbody>
</table>

Example of response XML (retrieving comments for a build result):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <comments expand="comments">
-   <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-5/comment"/>
-   <comments expand="comment" size="1" max-result="1" start-index="0">
-     - <comment author="apowers">
       <content>Groovy build result baby!</content>
-     </comment>
-   </comments>
</comments>
</comments>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| comments/link | • rel — Relationship between link and element (defaults to "self")
| | • href — URL for the comment |
| comments/comments | • size — Number of comments for the build result |
| comments/comments/comment | • author — Author of the comment |
| comments/comments/comment/content | Description: Content of the comment |
| comments/comments/comment/creationDate | Description: Creation date of the comment |
| comments/comments/comment/modificationDate | Description: Last modified date of the comment |

Example of response JSON:

```json
{ "expand":"comments", "link":{"rel":"self", "href":"http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-5/comment"}, "comments": [{ "expand":"comment", "size":1, "max-result":1, "start-index":0, "comment": [ { "author":"apowers", "content":"Groovy build result baby!", "creationDate":"2009-09-11T22:35:34.436+0200", "modificationDate":"2009-09-11T22:35:34.436+0200"} ] }]
}}
```

Example of request XML (posting a comment to a build result):
Elements in the request:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>• author — The author of the comment</td>
</tr>
<tr>
<td>comment/content</td>
<td>Description: Content of the comment to be added to the build result</td>
</tr>
</tbody>
</table>

Labels Service

| URI | /build/PROJECTKEY-PLANKEY-BUILDNUMBER/label |
| Description | Retrieves the labels for a build result, adds or removes a label to a build result. |
| URI Parameters | • no parameter — Retrieves the labels for a build result. |
| HTTP Methods | • GET — Retrieves the labels for a build result.  
• POST — Adds a label to the build result. Response will be empty.  
• DELETE — Removes a label from the build result, build/PROJECTKEY-PLANKEY-BUILDNUMBER/label/labelname e.g. Response will be empty. |

Example of response XML (retrieving labels for a build result):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <labels expand="labels">
  - <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-5/label"/>
  - <labels size="1" max-result="1" start-index="0"/>
  - <label name="label1"/>
</labels>
</labels>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| labels/link | • rel — Relationship between link and element (defaults to “self”)  
• href — URL for the label |
| labels/labels | • size — Number of labels for the build result |
| labels/labels/label | • name — Name of the label |

Example of response JSON:

```json
{"expand":"labels","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/build/MYPROJ1-MYPLAN1-5/label"},
"labels": [{
  "size":1,"max-result":1,"start-index":0,"label":{
  "name": "label1"}
}]
```
Example of request XML (adding a label to a build result):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<label name="label1" />
```

Elements in the request:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>name — The name of the label</td>
</tr>
</tbody>
</table>

Server Information Service

<table>
<thead>
<tr>
<th>URI</th>
<th>/info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Retrieves information about a Bamboo server.</td>
</tr>
<tr>
<td>URI Parameters</td>
<td>no parameter — Retrieves information about a Bamboo server.</td>
</tr>
<tr>
<td>HTTP Methods</td>
<td>GET — Retrieves information about a Bamboo server.</td>
</tr>
</tbody>
</table>

Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
-
<info>
  <version>2.4</version>
  <edition/>
  <buildDate>2009-09-11T20:47:44.000+0200</buildDate>
  <buildNumber>1503</buildNumber>
</info>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>info/version</td>
<td>Description: Bamboo version of server</td>
</tr>
<tr>
<td>info/buildDate</td>
<td>Description: Server build date</td>
</tr>
<tr>
<td>info/version</td>
<td>Description: Server build number</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{ "version": "2.4", "edition": "", "buildDate": "2009-09-11T20:47:44.000+0200", "buildNumber": "1503" }
```

Using the Bamboo REST APIs

There are a number of characteristics that are common across the Bamboo REST APIs. These are described below.

On this page:

- REST Authentication
- REST Resources and URI Structure
- Media Types
- API Versions
- HTTP Response Codes
- Methods
REST Authentication

You can authenticate yourself for the REST APIs in two ways:

- **Log into Bamboo manually.** You will then be authenticated for the REST APIs for that same browser session.
- **Use HTTP basic authentication** (Authorisation HTTP header) containing 'Basic username:password'. Please note however, *username:password* must be base64 encoded. The URL must also contain the 'os_authType=Basic' query parameter.

REST Resources and URI Structure

URLs for a Bamboo REST API resource have the following structure:

```
http://host:port/bamboo/rest/api-name/api-version/resource-name
```

**Example:**

```
http://myhost.com:8085/bamboo/rest/api/latest/plan
```

Here is an explanation for each part of the URI:

- **host** and **port** define the host and port where the Bamboo application lives.
- **bamboo** signifies that the request is to be directed to the 'bamboo' application.
- **rest** denotes the REST API.
- **api-name** identifies a specific API. This will always be **api** for a Bamboo REST API.
- **api-version** is the API version number, e.g. latest, 1, 2, etc. See the section on API version control.
- **resource-name** identifies the required resource. In some cases, this may be a generic resource name such as **/foo**. In other cases, this may include a generic resource name and key. For example, **/foo** returns a list of the **foo** items and **/foo/{key}** returns the full content of the **foo** identified by the given key.

Refer to the details of the specific REST resources.

Media Types

The Bamboo REST APIs return HTTP responses in one of the following formats:

<table>
<thead>
<tr>
<th>Response Format</th>
<th>Requested via...</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON</td>
<td>Requested via one of the following:</td>
</tr>
<tr>
<td></td>
<td>- application/json in the HTTP Accept header</td>
</tr>
<tr>
<td></td>
<td>- .json extension</td>
</tr>
<tr>
<td>XML</td>
<td>Requested via one of the following:</td>
</tr>
<tr>
<td></td>
<td>- application/xml in the HTTP Accept header</td>
</tr>
<tr>
<td></td>
<td>- .xml extension</td>
</tr>
</tbody>
</table>

**Example:**

To request an HTTP response on JSON format:

```
http://myhost.com:8085/bamboo/rest/api/latest/plan.json
```

API Versions

The Bamboo REST APIs are version controlled. The version number of an API appears in its URI. For example, use this URI structure to request version 1 of the Bamboo REST API:
To get the latest version of the API, you can also use the `latest` keyword. For example, if versions 1 and 2 of the 'admin' API are available, the following two URIs will point to the same resources:

- `http://myhost.com:8085/bamboo/rest/api/2/...

Notes:

- The API version number is an integer, such as 1 or 2.
- The API version is independent of the Bamboo release number.
- The API version may, or may not, change with a new Bamboo release. The API version number will change only when the updates to the API break the API contract, requiring changes in the code which uses the API. An addition to the API does not necessarily require a change to the API version number.

HTTP Response Codes

An error condition will return an HTTP error code as described in the Atlassian REST Guidelines.

Methods

You will use the standard HTTP methods to access Bamboo via the REST APIs. Please refer to the resource descriptions for further details.

Comparing the Bamboo Remote API to the Bamboo REST APIs

The Bamboo remote API has been deprecated in favour of the Bamboo REST APIs. We are working towards replacing the deprecated remote API with new REST APIs, however currently some methods in the deprecated remote API do not have an equivalent method in the new REST APIs.

This table below compares the methods from the Bamboo remote API with methods from the Bamboo REST APIs:

<table>
<thead>
<tr>
<th>Remote API Method</th>
<th>REST API equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Logout</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Update and Build</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Execute Build</td>
<td>No REST API equivalent (however, a build can be queued via the Queue Service)</td>
</tr>
<tr>
<td>List Projects</td>
<td>Project Service</td>
</tr>
<tr>
<td>Get Project Details</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>List Build Plans</td>
<td>Plan Service</td>
</tr>
<tr>
<td>Add Comment To Build Result</td>
<td>Comment Service</td>
</tr>
<tr>
<td>Add Label To Build Result</td>
<td>Label Service</td>
</tr>
<tr>
<td>Get Build Result Details</td>
<td>Build Result Service (partially equivalent)</td>
</tr>
<tr>
<td>Get Latest Build Result For Plan</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>Get Recently Completed Build Results For Plan</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>Get Latest Build Results For Project</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>Get Recently Completed Build Results For Project</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>Get Latest Builds By User</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>Get Latest Builds For Users Favourite Plans</td>
<td>Build Result Service</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>List Running Instances</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Start Elastic Instances</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Stop Elastic Instances</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Stop All Elastic Instances</td>
<td>No REST API equivalent</td>
</tr>
<tr>
<td>Get Bamboo Build Number</td>
<td>Server Information Service</td>
</tr>
</tbody>
</table>

**Bamboo Remote API**

⚠️ **Remote API Deprecated**
The Bamboo remote API described on this page has been deprecated in favour of the new Bamboo REST APIs. Please read the Bamboo REST API documentation before referring to the documentation below.

This section describes the ways of accessing the Bamboo remote API. Note that the terminology used in the REST API method and parameter names and descriptions may differ from what is used in the Bamboo UI. In some cases it follows conventions used within the Bamboo code.

To use Bamboo's remote API you will need to enable remote API access from within Bamboo itself. Instructions on how to do this can be found here: Enabling Bamboo's Remote API.

### RESTful

Whilst the Remote API capabilities of Bamboo are not true REST methods, they are RESTful and follow a similar format. The various API methods are accessed from `/api/rest/METHOD-NAME`

### Authentication

Bamboo requires authentication to access the remote API. All methods accept an authentication token parameter.

An authentication token can be acquired (and released) using the `login()` and `logout()` methods. The authentication will expire after 30 minutes of inactive use.

### Error Responses

The successful response is outlined for each individual method, error responses have the following format:

```xml
<errors>
  <error>Error message one</error>
  <error>Error message two</error>
</errors>
```

### Methods

**Authentication Services**

- Login
- Logout
Build Services

- Update and Build
- Execute Build
- List Projects
- Get Project Details
- List Build Plans

Build Results Services

- Add Comment To Build Result
- Add Label To Build Result
- Get Build Result Details

Build Results Filters

- Get Latest Build Result For Plan
- Get Recently Completed Build Results For Plan
- Get Latest Build Results For Project
- Get Recently Completed Build Results For Project
- Get Latest Builds By User
- Get Latest Builds For Users Favourite Plans

Elastic Bamboo Services

- List Running Instances
- Start Elastic Instances
- Stop Elastic Instances
- Stop All Elastic Instances

Other Services

- Get Bamboo Build Number

Build Results Filters

The following REST methods allow you to retrieve various lists of the build results. The following methods require authentication and results will be restricted by the user's permissions.

A lot of these methods return build result data. You can find more information on the data returned at the bottom of the page.

- Get Latest Build Result For Plan
- Get Recently Completed Build Results For Plan
- Get Latest Build Results For Project
- Get Recently Completed Build Results For Project
- Get Latest Builds By User
- Get Latest Builds For Users Favourite Plans

Get Latest Build Result For Plan

Provides the last build result for the given buildKey.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>buildKey</td>
<td>Key representing the build plan to obtain latest build result for</td>
</tr>
</tbody>
</table>

URL:

/api/rest/getLatestBuildResults.action

Successful Response
Get Recently Completed Build Results For Plan

Provides the overall most recently completed build results for the given build (as opposed to getLatestBuildResults, which gives the most recent build result for the build).

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>buildKey</td>
<td>Key representing the build plan to obtain latest build results for.</td>
</tr>
</tbody>
</table>

URL:

`/api/rest/getRecentlyCompletedBuildResultsForBuild.action`

Successful Response:

```
<response>
  <build>
    ...  
    Build result data (see below)
    ...
  </build>
  ...
  <build>
    ...  
    Build result data (see below)
    ...
  </build>
</response>
```

Get Latest Build Results For Project

Provides the latest result for each build in the given project.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method.</td>
</tr>
<tr>
<td>projectKey</td>
<td>Key representing the project to obtain latest build results for.</td>
</tr>
</tbody>
</table>

URL:

`/api/rest/getLatestBuildResultsForProject.action`

Successful Response:
Get Recently Completed Build Results For Project

Provides the overall most recently completed build results for the given project (as opposed to getLatestBuildResultsForProject, which gives the most recent build result for each build in the project).

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method.</td>
</tr>
<tr>
<td>projectKey</td>
<td>Key representing the project to obtain latest build results for.</td>
</tr>
</tbody>
</table>

URL:

/api/rest/getRecentlyCompletedBuildResultsForProject.action

Successful Response:

```
<response>
  <build>
    ...  
    Build result data (see below)  
    ...  
  </build>
  <build>
    ...  
    Build result data (see below)  
    ...  
  </build>
</response>
```

Get Latest Builds By User

Returns recent builds triggered by a particular author.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method.</td>
</tr>
<tr>
<td>username</td>
<td>Username</td>
</tr>
</tbody>
</table>

REST:

/api/rest/getLatestBuildsByUser.action

Successful Response:
Get Latest Builds For Users' Favourite Plans

Provides the latest build details for each plan in the user's list of favourite plans.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method.</td>
</tr>
<tr>
<td>username</td>
<td>the username of the user to grab the favourites for</td>
</tr>
</tbody>
</table>

URL:

/api/rest/getLatestUserBuilds.action

Successful Response:

```
<response>
  <build>
    ... Build result data (see below) ...
  </build>
  <build>
    ... Build result data (see below) ...
  </build>
</response>
```

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Build Name</td>
<td>User Friendly Build Name as shown in Bamboo</td>
</tr>
<tr>
<td>key</td>
<td>Build Key</td>
<td>String identifier for the plan (in the format &lt;projectKey&gt;-&lt;planKey&gt;)</td>
</tr>
<tr>
<td>state</td>
<td>Build State</td>
<td>current (currently building) / success / fail / none</td>
</tr>
<tr>
<td>buildNumber</td>
<td>Number for the last build result</td>
<td>Number &gt; 0</td>
</tr>
<tr>
<td>failedTestCount</td>
<td>Number of failed tests for this build</td>
<td>Number &gt;= 0</td>
</tr>
<tr>
<td>successfulTestCount</td>
<td>Number of successful tests in this build</td>
<td>Number &gt;= 0</td>
</tr>
<tr>
<td>buildTime</td>
<td>The time the build was started</td>
<td>Readable string representation of the time</td>
</tr>
</tbody>
</table>
Build Result Data

Many of the methods above will return a build result with the following information.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectName</td>
<td>Name of the build's project</td>
<td>User friendly name for the project</td>
</tr>
<tr>
<td>buildName</td>
<td>Name of the plan</td>
<td>User friendly name for the plan (does not include project name)</td>
</tr>
<tr>
<td>buildKey</td>
<td>Plan Key</td>
<td>Identifier for this plan in the format &lt;projectKey&gt;-&lt;planKey&gt;</td>
</tr>
<tr>
<td>buildState</td>
<td>State of the Build</td>
<td>Successful / Failed</td>
</tr>
<tr>
<td>buildNumber</td>
<td>The number of the build</td>
<td>Number &gt; 0</td>
</tr>
<tr>
<td>failedTestCount</td>
<td>The number of tests that failed</td>
<td>Number &gt;= 0</td>
</tr>
<tr>
<td>successfulTestCount</td>
<td>The number of tests that passed</td>
<td>Number &gt;= 0</td>
</tr>
<tr>
<td>buildTime</td>
<td>Time stamp of when the build started</td>
<td>Formatted Date (yyyy-MM-dd HH:mm:ss)</td>
</tr>
<tr>
<td>buildCompletedDate</td>
<td>Time stamp of when the build finished</td>
<td>Formatted Date/yyyy-MM-dd'T'HH:mm:ssZ</td>
</tr>
<tr>
<td>buildDurationIn Seconds</td>
<td>How long the build took</td>
<td>Number of seconds</td>
</tr>
<tr>
<td>buildDurationDescription</td>
<td>How long the build took</td>
<td>Pretty Print of duration</td>
</tr>
<tr>
<td>buildRelativeBuildDate</td>
<td>How long ago the build occurred</td>
<td>Pretty Print of duration since the build occurred</td>
</tr>
<tr>
<td>buildTestSummary</td>
<td>Summary of the test results</td>
<td>Pretty Print summary of test results</td>
</tr>
<tr>
<td>buildReason</td>
<td>Why the build occurred</td>
<td>String representation of the trigger reason</td>
</tr>
<tr>
<td>commits</td>
<td>List of any commits included in this build</td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>An individual commit included in this build</td>
<td></td>
</tr>
<tr>
<td>commit: author</td>
<td>The author that made the commit</td>
<td>The author name as provided by the vcs system</td>
</tr>
</tbody>
</table>

Example Response

```xml
<projectName>Bamboo</projectName>
<buildName>HEAD</buildName>
<buildKey>BAM-TRUNK</buildKey>
<buildState>Successful</buildState>
<buildNumber>4681</buildNumber>
<failedTestCount>0</failedTestCount>
<successfulTestCount>1497</successfulTestCount>
<buildTime>2009-02-24T16:39:46</buildTime>
<buildCompletedDate>2009-02-24T16:58:42+1100</buildCompletedDate>
<buildDurationInSeconds>1136</buildDurationInSeconds>
<buildDurationDescription>18 minutes</buildDurationDescription>
<buildRelativeBuildDate>42 minutes ago</buildRelativeBuildDate>
<buildTestSummary>1497 passed</buildTestSummary>
<buildReason>Code has changed</buildReason>
<commits>
<commit author="bob"/>
</commits>
```

Other Services

Below you will find any REST methods that do not fall under any other category. All methods on this page require authentication.

- Get Bamboo Build Number

Get Bamboo Build Number

Returns the application build number of the Bamboo Server

Arguments:
### Authentication Services

Bamboo requires authentication to access the remote API. Most methods accept an authentication token parameter. An authentication token can be acquired (and released) using the login() and logout() methods. The authentication will expire after 30 minutes of inactive use.

- Login
- Logout

#### Login

Login and create an authentication token. Returns the token if login was successful, or returns an error otherwise.

**Arguments:**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>username to authenticate with</td>
</tr>
<tr>
<td>password</td>
<td>password corresponding to the username</td>
</tr>
</tbody>
</table>

**URL:**

```
/api/rest/login.action
```

**Example:**

```
/api/rest/login.action?username=bob&password=bobsPassword
```

**Successful Response:**

```
<response>
  <auth>AuthenticationIdentifier</auth>
</response>
```

**Example:**

```
<response>
  <auth>aU5ybWbzfw</auth>
</response>
```

#### Logout

Disables the given authentication token.
Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token for the session to log out from</td>
</tr>
</tbody>
</table>

URL:

/api/rest/logout.action

Example:

/api/rest/logout.action?auth=aU5ybWbzfw

Successful Response:

```
<response>
  <auth/>
</response>
```

Elastic Bamboo Services

The following methods are available to interact with the Elastic Bamboo functionality. For more information on Elastic Bamboo concepts and usage please refer to Working with Elastic Bamboo. All methods on this page require authentication by a user with administration privileges.

- List Running Instances
- Start Elastic Instances
- Stop Elastic Instances
- Stop All Elastic Instances

List Running Instances

Provide a list of any running elastic instances.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
</tbody>
</table>

URL:

/api/rest/listRunningInstances.action

Successful Response:

```
<response>
  <instance>
    <id>i-ee54c287</id>
    <state>RUNNING</state>
    <uptime>94430</uptime>
  </instance>
</response>
```

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The elastic instance ID</td>
<td>This ID is provided by Amazon when the instance is started up. It can be null</td>
</tr>
</tbody>
</table>
state | The state of the elastic instance | STARTING (instance pending) / IDENTIFIED (instance pending with identification details available) / RUNNING / SHUTTING_DOWN / TERMINATED / FAILED_TO_START (error state) / UNKNOWN (error state)
---|---|---
uptime | The length of time the instances has been running (according to AWS) | Duration in milliseconds

Start Elastic Instances

Start a given number of elastic instances (and the corresponding elastic agents).

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>numAgentsToCreate</td>
<td>Number of instance you would like to start</td>
</tr>
<tr>
<td>elasticImageConfigurationName</td>
<td>(optional) The name of the elastic image configuration that you want to start your instance from, e.g. &quot;Default&quot;. If you do not specify a value for this parameter, Bamboo will use the default image configurations that is shipped with Bamboo to start your instance from.</td>
</tr>
</tbody>
</table>

URL:

/api/rest/startElasticInstances.action

Example:

/api/rest/startElasticInstances.action?auth=6Tsb5E6G6o&numAgentsToCreate=2&elasticImageConfigurationName=Default

Successful Response:

<response/>

Stop Elastic Instances

Request termination of an Elastic Instance. Any builds running on the instance will be abandoned.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>instanceId</td>
<td>The ID for the instance to be stopped</td>
</tr>
</tbody>
</table>

URL:

/rest/api/StopElasticAgents.action

Example:

/rest/api/StopElasticAgents.action?auth=6Tsb5E6G6o&instanceId=i-ee54c287

Successful Response:
Stop All Elastic Instances

Terminates all running Elastic Instances. Any builds running on the instance will be abandoned.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
</tbody>
</table>

URL:

```
/api/rest/stopAllElasticInstances.action
```

Successful Response:

```
<response/>
```

Build Results Services

The following methods allow you to interact with Bamboo's build results. The following methods require authentication and actions/results will be restricted by the user's permissions.

- Add Comment To Build Result
- Add Label To Build Result
- Get Build Result Details

Add Comment To Build Result

Adds a comment to the given build result

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>buildKey</td>
<td>Key representing the build plan.</td>
</tr>
<tr>
<td>buildNumber</td>
<td>The build number you wish to add the comment to.</td>
</tr>
<tr>
<td>content</td>
<td>The comment to add. Remember to ensure it is escaped correctly</td>
</tr>
</tbody>
</table>

URL:

```
/api/rest/addCommentToBuildResults.action
```

Example:

```
/api/rest/addCommentToBuildResults.action?auth=YOwcnQHh5q&buildKey=MYPROJECT-TRUNK&buildNumber=109&content=my%20comment
```

Successful Response:

```
<response/>
```
Add Label To Build Result

Add a label to the given build result

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method.</td>
</tr>
<tr>
<td>buildKey</td>
<td>Key representing the build plan.</td>
</tr>
<tr>
<td>buildNumber</td>
<td>The build number you wish to add the label to.</td>
</tr>
<tr>
<td>label</td>
<td>The label to add</td>
</tr>
</tbody>
</table>

URL:

/api/rest/addLabelToBuildResults.action

Example:

/api/rest/addLabelToBuildResults.action?auth=YOwcnQHh5q&buildKey=MYPROJECT-TRUNK&buildNumber=109
&label=myLabel

Successful Response

<response/>

Get Build Result Details

Returns more specific information about a particular build result

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>buildKey</td>
<td>Key representing the build plan</td>
</tr>
<tr>
<td>buildNumber</td>
<td>Number identifying the build result to retrieve</td>
</tr>
</tbody>
</table>

URL:

/api/rest/getBuildResultsDetails.action

Example:

/api/rest/getBuildResultsDetails.action?auth=YOwcnQHh5q&buildKey=MYPROJECT-TRUNK&buildNumber=109

Successful Response
### Element Description Value Range

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcsRevisionKey</td>
<td>The revision built in this builds</td>
<td>Dependent on vcs system e.g. svn uses numbering system, CVS uses dates</td>
</tr>
<tr>
<td>commits</td>
<td>List of commits included in this build</td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td>Represents an individual change included in this build</td>
<td></td>
</tr>
<tr>
<td>commit: author</td>
<td>The author that committed the change</td>
<td>Author name as provided by the vcs system</td>
</tr>
<tr>
<td>commit: date</td>
<td>Timestamp of when the change was committed</td>
<td>String representation of the data provided by the vcs system</td>
</tr>
<tr>
<td>comment</td>
<td>Any comment provided along with the commit</td>
<td>Freeform content.</td>
</tr>
<tr>
<td>files</td>
<td>List of files changed in the commit</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>Represents and individual file changed as part of the commit</td>
<td></td>
</tr>
<tr>
<td>file: name</td>
<td>The name of the file</td>
<td>Full path of file as provided by the vcs system</td>
</tr>
<tr>
<td>file: revision</td>
<td>The revision of the individual file</td>
<td>Depends on the vcs system, can be numeric, can be date, can be the same as the vcsRevision</td>
</tr>
<tr>
<td>failedTests</td>
<td>Lists any tests that failed as part of this build</td>
<td></td>
</tr>
<tr>
<td>successfulTests</td>
<td>Lists any tests that passed in this build</td>
<td></td>
</tr>
<tr>
<td>testResult</td>
<td>Represents an individual test</td>
<td></td>
</tr>
<tr>
<td>testResult: testClass</td>
<td>The class name of the test</td>
<td>Fully qualified class name</td>
</tr>
<tr>
<td>testResult: testMethod</td>
<td>The method name of the test</td>
<td>Unchanged method name</td>
</tr>
<tr>
<td>testResult: duration</td>
<td>How long it took the test to execute</td>
<td>Duration in seconds</td>
</tr>
</tbody>
</table>
Build Services

The following methods allow you to interact with your plans. You will require an authentication token for most of these methods.

- Update And Build
- Execute Build
- List Projects
- Get Project Details
- List Build Plans

Update And Build

Instruct Bamboo to checkout the latest version of the build and build it. This is typically used by a CV/SVN trigger script.

This method does not use the standard API authentication mechanism. Bamboo will only accept this remote call if it originated from the IP address of the source control server.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildKey</td>
<td>key identifier of the build you want to trigger</td>
</tr>
</tbody>
</table>

URL:

/api/rest/updateAndBuild.action

Successful Response:

```xml
<response>
  <success>A build of Moo was triggered by remote http call.</success>
</response>
```

Execute Build

Instruct Bamboo to checkout the latest version of the build and build it. This is typically used to trigger a manual build.

This method uses the standard authentication API.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>buildKey</td>
<td>key identifier of the build you want to trigger</td>
</tr>
</tbody>
</table>

URL:

/api/rest/executeBuild.action

Example:

/api/rest/executeBuild.action?auth=HGYj89Kju&buildKey=BAM-MAIN

Successful Response:
List Projects

Provides a list of all the projects on this Bamboo server.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
</tbody>
</table>

URL:

/api/rest/listProjectNames.action

Successful Response:

```
<response>
  <project>
    <name>My Awesome Project</name>
    <key>MY_PROJECT</key>
  </project>
</response>
```

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>project</td>
<td>Represents an individual project</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>Project Name</td>
<td>User friendly name that appears in Bamboo.</td>
</tr>
<tr>
<td>key</td>
<td>Project Key</td>
<td>Project identifier</td>
</tr>
</tbody>
</table>

Get Project Details

Provides details about a specific Bamboo Project.

Arguments:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
<tr>
<td>projectKey</td>
<td>Identifying key of the project required</td>
</tr>
</tbody>
</table>

URL:

/api/rest/getProjectDetails.action

Example:

/api/rest/getProjectDetails.action?auth=m30aZl4Ck2&projectKey=MY_PROJECT

Successful Response:
**List Build Plans**

Provides a list of all the build plans on this Bamboo server.

**Arguments:**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Authentication token retrieved via the login method</td>
</tr>
</tbody>
</table>

**URL:**

```
/api/rest/listBuildNames.action
```

**Successful Response:**

```
<response>
  <build enabled="true">
    <name>My Project - Trunk</name>
    <key>MYPROJECT-TRUNK</key>
  </build>
  <build enabled="false">
    <name>My Project - Branch</name>
    <key>MYPROJECT-BRANCH</key>
  </build>
</response>
```

**Bamboo Developer FAQ**

This is a constantly updated FAQ listing questions and answers asked by people developing Bamboo plugins and working with the Bamboo code base in general. For general questions, check the Bamboo FAQ.
Questions

How do I inject managers into my plugin?

How do I search for previous build result?

How do I inject managers into my plugin?

You can inject manager objects into your plugin through setter based injection. Managers in Bamboo provide various services that's available to the application.

For example, if you want to inject BuildResultsSummaryManager into your plugin, you can add:

```java
private BuildResultsSummaryManager buildResultsSummaryManager;

public void setBuildResultsSummaryManager(final BuildResultsSummaryManager buildResultsSummaryManager) {
    this.buildResultsSummaryManager = buildResultsSummaryManager;
}
```

Spring will search for the bean named "buildResultsSummaryManager" and call the setter, passing in the manager.

How do I search for previous build result?

For retrieving history of a build, you should have the BuildResultsSummaryManager injected into your plugin through setter injection. Once that is done there's a series of search methods that you can get a handle on. The most flexible one being findBuildResultsSummaries, where you can have arbitrary search criteria.

How do I start a build programatically?

Below is the code for starting a manual build (it'll show the changes since the last build). The work happens through the BuildExecutionManager. The buildExecutionManager.tryToDetectAndBuild takes a callback function BuildExecutionManager.BuildDetectionAction that creates theBuildContext. You can get away with not populating the BuildChangesImpl if you don't want to (no changes will be shown for the generated build result)
or if you just want to fire off a manual build, you can use call the method

```java
public void startManualBuild(@NotNull final Build buildPlan, @NotNull final User user)
```

or if you just want to fire off a manual build, you can use call the method

```java
buildExecutionManager.tryToDetectAndBuild(buildKey, new BuildExecutionManager.BuildDetectionAction()
{
    public BuildContext process()
    {
        try
        {
            BuildLogger buildLogger = buildPlan.getBuildLogger();
            BuildDefinition buildDefinition = buildPlan.getBuildDefinition();
            Repository repository = buildDefinition.getRepository();
            // Some feedback to the UI for what's going down
            BuildExecutionManager.TryToDetectAndBuild(buildKey, new BuildExecutionManager.BuildDetectionAction()
            {
                buildLogger.addBuildLogEntry("Manual build triggered by " + user);
            },
            new BuildDefinitionImpl(buildDefinition),
            new BuildRepositoryImpl(repository))
            {
                buildChanges = changeDetectionManager.collectChangesSinceLastBuild(buildPlan.getKey(),
                repository, lastVcsRevisionKey);
            }
            else
            {
                buildChanges = new BuildChangesImpl();
            }
            // A trigger reason
            TriggerReason triggerReason =
                triggerReasonManager.getTriggerReason(ManualBuildTriggerReason.KEY, buildChanges,
                buildContext = newBuildContextImpl(buildPlan,
                buildManager.set:
                // needed to get the correct build result
                triggerReason, buildDefinition, buildChanges));
            return buildContext;
        }
        catch (Exception e)
        {
            String message = "Error encountered while triggering manual build: " +
            e.getMessage();
            log.error(buildPlan.getBuildLogger().addBuildLogEntry(message, e);
            errorUpdateHandler.recordError(buildPlan.getKey(), message, e);
        }
        return null;
    }
}, true);
```

How do I trigger off a build from my action?

Below is the code for starting a manual build (it'll show the changes since the last build). The
buildExecutionManager.tryToDetectAndBuild takes a callback function BuildExecutionManager.BuildDetectionAction that
creates the BuildContext. You can get away with not populating the BuildChangesImpl if you don't want to (no changes will be shown for
the generated build result)
Bamboo Plugin Tutorial

Introduction

The purpose of this tutorial is to demonstrate how you can add custom functionality to Bamboo via plugins. The tutorial aims to give you a good starting point for Bamboo plugin development, and how the different Bamboo plugin modules can work together. In this tutorial, we will run through the development of a plugin derived from a real use-case requirement.

Bamboo Labeller Plugin

There are many cases when builds in Bamboo fail because of particular, recurring errors. For example, a functional test in Confluence may periodically fail because of an OutOfMemoryError when things get hectic. It would be useful for developers to keep track of these particular builds, so they can look into it further. For this to happen, Bamboo will need to:
• Parse the error logs after a failed build.
• Look for the text `java.lang.OutOfMemoryError` in the log
• If found, tag the build with a label, say `out_of_memory`

Once that's done, the developer can set up a RSS feed on the `out_of_memory` tag. They will then be able to keep track of the builds which fail with an OutOfMemoryError. We can even extend this concept further, by replacing the search for `java.lang.OutOfMemoryError` with any regular expression, tagging it with a label of choice.

Contents

• Tutorial 1 - Getting Started with a Simple Post Build Labeller
• Tutorial 2 - Configurable Regex Labeller

RELATED TOPICS

• Bamboo Plugin Guide
  • Bamboo Plugin Module Types
    • Bamboo Event Listener Module
    • Build Agent Requirement Filter Module
    • Command Decorator Module
    • Notification Recipient Module
    • Notification Type Module
      • Building a Notification Plugin
    • Builder Plugin Module
    • Source Repository Module
      • Repository Extensions
    • Trigger Reason Module
    • Pre Build Queued Action
    • Pre Build Action Module
    • Build Processor Module
    • Build Processor Server Module
    • Build Complete Action Module
    • Index Reader Module
    • Post Build Index Writer Module
    • Report Module
    • XWork Plugin Module
    • Web Item Module
    • Web Section Module
    • Notification Condition Module
  • Bamboo’s Build Process
  • Accessing Bamboo Components From Plugin Modules
  • Common Bamboo Classes
  • Bamboo Persistence using Bandana
  • Downloadable Plugin Resources
  • Web Resources
• Bamboo REST APIs
  • Bamboo REST Resources
  • Using the Bamboo REST APIs
  • Comparing the Bamboo Remote API to the Bamboo REST APIs
• Bamboo Remote API
  • Build Results Filters
  • Other Services
  • Authentication Services
  • Elastic Bamboo Services
  • Build Results Services
  • Build Services
• Bamboo Developer FAQ
  • How do I inject managers into my plugin?
  • How do I search for previous build result?
  • How do I start a build programatically?
  • How do I trigger off a build from my action?
• Bamboo Plugin Tutorial
  • Tutorial 1 - Getting Started with a Simple Post Build Labeller
  • Tutorial 2 - Configurable Regex Labeller
• Changes by Version
  • Changes for Bamboo 2.0
    • Changes to Bamboo’s Configuration UI require the ConfigurablePlugin
    • Post-Build Processing in Bamboo 2.0
    • Build Process for 2.0
    • Repository plugin changes in 2.0
    • Updates to the build processing plugins
  • Changes for Bamboo 2.1
Tutorial 1 - Getting Started with a Simple Post Build Labeller

The source code to the plugin used in this tutorial is available on the Atlassian public source repository. You can check out the source code [here](http://www.atlassian.com).

**Step 1 - Setting up the project.**

The first thing you need to do is to set up your Bamboo Plugin project and source directories. See the Getting started with Atlassian plugins guide and the The Bamboo Plugin Guide for instructions.

In the `atlassian-plugin.xml` located under `/src/main/resources/`, you will need to give the plugin a unique key, as well as some meta information about this plugin. As our plugin simply labels, we have called it "labeller". Below is the `atlassian-plugin.xml` for our labelling plugin:

```xml
<atlassian-plugin key="com.atlassian.bamboo.plugin.labeller" name="Build Labeller">
    <plugin-info>
        <description>Bamboo Labeller</description>
        <version>1.0</version>
        <application-version min="1.0" max="1.0"/>
        <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
    </plugin-info>
</atlassian-plugin>
```

Now we are ready to move onto writing some code to make our plugin do something.

**Step 2 - Adding the first Build Complete Labeller Module**

In this plugin, we want Bamboo to perform a custom action immediately after a build has completed. To do this, we write a Build Complete Action Module. You can see all the available Bamboo module types [here](http://www.atlassian.com).

To start things off, we would like to keep our custom action pretty simple and make sure things work. Our first cut of the `BuildLabeller` will simply label the build as "out_of_memory" if the "OutOfMemoryError" was found in the logs.
Our custom module must implement the `CustomBuildCompleteAction` interface, which defines a `run` method and a `validate` method.

The `run` method is what gets called when a build completes. Our run method in this plugin is fairly simple. It loops through each line of the build logs and searches for the exact string - "OutOfMemoryError". Once found, it stops looping and labels the build.

In the run method, we make use of the services of the LabelManager (a dependency), which is responsible for tagging of a build. Dependencies in plugins are automatically handled by Bamboo Spring container. As long as the plugin has the correct "setter" method, the dependency will be automatically injected.

You may notice that the other method defined by the `CustomBuildCompleteAction` interface: validate currently doesn't do anything. We will return to this in the next tutorial.

**Step 3 - Registering the Build Complete Labeller Module**

Once you have written your labeller module, we must now register the plugin module into our plugin descriptor (`atlassian-plugin.xml`).
**Step 4 - Build and Test**

That's it. We now need to test our code. To do this, we can build our plugin by returning to the command line in the root directory of your source directory, and run the command: `mvn package`. This created a `labeller-plugin-1-1.0.jar`. We can now drop this into Bamboo (`/webapp/WEB-INF/lib`), and see it in action.

Here is what our plugin produced after we ran a build with a `OutOfMemoryError`:

![Bamboo Plugin Result]

**Next Steps**

So we have made our first basic plugin. Right now, it's not very configurable, and runs for every build. In the next tutorial, we will introduce configurability to our Labeller.

Next >

**RELATED TOPICS**

- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - Bamboo Event Listener Module
    - Build Agent Requirement Filter Module
    - Command Decorator Module
    - Notification Recipient Module
    - Notification Type Module
      - Building a Notification Plugin
    - Builder Plugin Module
    - Source Repository Module
      - Repository Extensions
    - Trigger Reason Module
    - Pre Build Queued Action
    - Pre Build Action Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Complete Action Module
    - Index Reader Module
    - Post Build Index Writer Module
    - Report Module
    - XWork Plugin Module
    - Web Item Module
    - Web Section Module
    - Notification Condition Module
  - Bamboo’s Build Process
  - Accessing Bamboo Components From Plugin Modules
  - Common Bamboo Classes
  - Bamboo Persistence using Bandana
  - Downloadable Plugin Resources
  - Web Resources
- Bamboo REST APIs
  - Bamboo REST Resources
  - Using the Bamboo REST APIs
  - Comparing the Bamboo Remote API to the Bamboo REST APIs
Tutorial 2 - Configurable Regex Labeller

In the previous tutorial, we have made our plugin label the build whenever the logs had the words "OutOfMemoryError". This, however, is not very useful for the other builds which don't have this memory problem. Also, it is not very useful to only be able to tag with "out_of_memory". In this tutorial, we will extend on the plugin module so that we can configure when to label, and what to label a build with.

The source code to the plugin used in this tutorial is available on the Atlassian public source repository. You can check out the source code [here](https://example.com).

**Step 1 - Adding configuration views**

To do this, we must first add the views for configuring the labeller. The BuildCompleteAction module type comes with the capability to accept Freemarker templates which allows you to edit and view custom configuration in the Build Plan Configuration page, under the Post Action tab.

**Edit Configuration View**

The Freemarker template to edit our Labeller configuration is below (regexLabellerEdit.ftl):

```freemarker
[@ui.bambooSection title='Pattern matching labelling.' ]
[ww.textfield name='custom.bamboo.labeller.regex' label='Regex Pattern' description='The regular expression for which to match the log files on.' ]
[ww.textfield name='custom.bamboo.labeller.label' label='Label(s)' description='The label(s) for the build if it matches the specified regex pattern.' ]
[/@ui.bambooSection ]
```

Here, we define a section with a title ‘Pattern matching labelling.’ Inside our configuration section are two text fields, one for the regex expression for matching against the logs, and one for the label(s) that we want to tag a build with if the regex expression matches.

We have named our two text fields `custom.bamboo.labeller.regex` and `custom.bamboo.labeller.label`. These are the keys to your custom configuration property stored in Bamboo.

Please note that these keys **must** start with "custom." for Bamboo to recognize and store within the plan's configuration. You may also notice that the keys are "namespaced". This is a good idea to prevent a clash of custom configuration properties.

**Display Configuration View**

We also define a Freemarker view for viewing the configuration (read-only). The display configuration view is below (regexLabellerView.ftl):

```freemarker
[@ui.bambooSection title='Pattern matching labelling.' ]
[@ww.textfield name='custom.bamboo.labeller.regex' label='Regex Pattern' description='The regular expression for which to match the log files on.' ]
[@ww.textfield name='custom.bamboo.labeller.label' label='Label(s)' description='The label(s) for the build if it matches the specified regex pattern.' ]
[/@ui.bambooSection ]
```
Here we simply build display the configuration by retrieving your custom properties via the same keys we used in the edit view.

**Registering the views in the Plugin Descriptor**

We need to register these two Freemarker templates as part of our `BuildCompleteAction` module. We do this by adding `<resource>` tags with the file path of the templates within the module descriptor definition.

```xml
<buildCompleteAction key="labeller" name="Build Labeller class="
  com.atlassian.bamboo.plugins.labeller.BuildLabeller">
  <description>An automatic labelling plugin.</description>
  <resource type="freemarker" name="edit" location="templates/buildCompleteAction/regexLabellerEdit.ftl"/>
  <resource type="freemarker" name="view" location="templates/buildCompleteAction/regexLabellerView.ftl"/>
</buildCompleteAction>
```

Once that's done, we can see the templates in action.

Under the edit configuration page:

<table>
<thead>
<tr>
<th>Pattern matching labelling.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regex Pattern:</strong></td>
</tr>
<tr>
<td><strong>Label(s):</strong></td>
</tr>
</tbody>
</table>

And under the view configuration page:

<table>
<thead>
<tr>
<th>Pattern Matching Labelling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regex Pattern:</strong></td>
</tr>
<tr>
<td><strong>Labels:</strong></td>
</tr>
</tbody>
</table>

**Step 2 - Adding validation**

Inserting the templates has allowed us to view and edit custom plan configuration properties. However, we should validate the input we provide for the `BuildLabeller`, to catch invalid labels or regex patterns.

This is where we use the `validate` method within our `BuildLabeller` class, which we have previously left to return `null` in the first tutorial. Bamboo will run this validate method before trying to save custom configuration properties.
The `BuildConfiguration` object passed to the validation method is the in-memory version of the build plan configuration. You can get your custom property by simply calling `getString` on the object, providing the custom property key that you used in the Freemarker templates.

### Step 3 - Applying the configuration

At this stage, we can edit, validate, and view our custom configuration for this plugin module. We now need to modify our original `run` method within the `BuildLabeller` to read the custom configuration properties.
/**
 * This action will run after a build has completed.
 * The build will be tagged with a specified set of labels if the logs matches the specified
 * regex pattern.
 *
 * @param build
 * @param buildResults
 */
public void run(Build build, BuildResults buildResults)
{
    // grab the custom configuration object
    Map customConfiguration = build.getBuildDefinition().getCustomConfiguration();

    if (customConfiguration != null)
    {
        if (customConfiguration.containsKey("custom.bamboo.labeller.label"))
        {
            List logs = buildResults.getBuildLog();
            String pattern = (String) customConfiguration.get("custom.bamboo.labeller.regex");

            Pattern regexPattern = Pattern.compile(pattern);

            // Go through the logs
            for (Iterator iterator = logs.iterator(); iterator.hasNext();)
            {
                SimpleLogEntry log = (SimpleLogEntry) iterator.next();
                Matcher matcher = regexPattern.matcher(log.getLog());

                // Use a matcher to see if the logs contained the specified regex
                if (matcher.find())
                {
                    String labelsInput = (String) customConfiguration.get("custom.bamboo.labeller.label");

                    // Our configuration also allows for multiple labels.
                    List labels = LabelParser.split(labelsInput);

                    for (Iterator iterator2 = labels.iterator(); iterator2.hasNext();)
                    {
                        String label = (String) iterator2.next();
                        getLabelManager().addLabel(label, buildResults, null);
                     
                    }
                }
            }

        }
    }
}
Changes by Version

The introduction of new features and improvements in progressive versions of Bamboo have required significant changes to the underlying architecture of the application. If you have developed any plugins for Bamboo, please refer to the following version-specific pages for important information regarding these changes:

- Changes for Bamboo 2.0
- Changes for Bamboo 2.1
- Changes for Bamboo 2.1.5
- Changes for Bamboo 2.2
- Changes for Bamboo 2.3

Changes for Bamboo 2.0

The introduction of new features and improvements in Bamboo 2.0 required significant changes to the underlying architecture of the application. If you have developed any plugins for Bamboo, please refer to the following pages for important information regarding these changes:

- Changes to Bamboo's Configuration UI require the ConfigurablePlugin
- Post-Build Processing in Bamboo 2.0
- Build Process for 2.0
Repository plugin changes in 2.0

Updates to the build processing plugins

Changes to Bamboo's Configuration UI require the ConfigurablePlugin

In Bamboo 2.0, any plug-in that adds to Bamboo's configuration user interface must also implement com.atlassian.bamboo.v2.build.ConfigurablePlugin. This is most easily achieved by extending com.atlassian.bamboo.v2.build.BaseConfigurablePlugin.

If you do not implement the ConfigurablePlugin, the user interface for your custom plugin may not display correctly (even when the plugin works correctly otherwise).

Post-Build Processing in Bamboo 2.0

Due to Bamboo 2.0's support for distributed builds, the plug-in point for performing post-build processing has been split into two:

CustomBuildProcessor: These plug-in modules are executed on the agent. They have access to the build filesystem, but not to any Bamboo managers.

CustomBuildProcessorServer: These plug-in modules are executed on the Bamboo server, once the build result has been transmitted from the agent. They have access to Bamboo managers, but they are not guaranteed to have access to the build filesystem.

If you have a plug-in that needs to read data from the build filesystem, and make use of Bamboo managers to retrieve and/or store data, you must use the following approach:

On the Bamboo agent:

1. Have a CustomBuildProcessor read the data you require from the build filesystem on the agent;
2. Store the data you require as key-value pairs in the Map<String, String> obtained from calling getCustomData() of the BuildResult;

Then, on the Bamboo server:

1. Have a CustomBuildProcessorServer read your key-value pairs from the BuildResult on the Bamboo server;
2. Make use of the the Bamboo managers to retrieve and store data as required.

Build Process for 2.0

The build process has been refactored in Bamboo 2.0 into a series of BuildTasks, each responsible for a particular part of the build process. Each build task has the ability to store aBuildContext. ABuildContext represents the definition of the part of the build process that the build task is responsible for.

The steps of a build process are described below:

1. Change Detection (performed by the server) — Changes are detected via polling, manual builds, etc. When a new build is required, the change logs are defined for the build and the build number for the new build is determined. The build context is also created.
2. Build Queued (performed by the server) — The server decides which agents can execute the build and queues the build.
3. Agent Picks Up Build (performed by the agent) | The capability context is set. The build is removed from queue. The agent begins looping through the build tasks.
4. Update Source (performed by the agent) — The agent runs the CheckoutUpdateForBuild task. The Repository#retrieveSourceCode is called. If the Repository is RepositoryEventAware the appropriate methods will be called. The agent checks if the repository has changed since the last build and clears the source directory if it has.
5. Prepare for the Build (performed by the agent) — The agent runs the PrepareBuildTask. This begins streaming the logs back to the server. The agent also runs the CustomPreBuildAction plugin point. From this point on, the build will be saved.
6. Executes the Build (performed by the agent) — Timer begins. The agent runs the Builder#executeBuild. After the Builder has been run, all the CustomBuildProcessors are run. Timer is stopped.
7. Results Processing (performed by the server) — The server runs CustomBuildProcessorServer, checks if the build has passed and saves and indexes the build result. The server also fires off the BuildCompletedEvent and PostBuildCompletedEvent events.
8. On BuildCompletedEvent (performed by the server) — Notifications are sent in this phase.

Repository plugin changes in 2.0

The Repository plugin has changed significantly in Bamboo 2.0. Previously, the getChangesSinceLastBuild method was responsible for detecting the changes and updating the repository to the latest code. This has been broken into two separate methods now, as described below:

1. collectChangesSinceLastBuild
The `collectChangesSinceLastBuild` method returns a `BuildChanges` object that encapsulates the commits and changes between the last build and the current source repository (through the `vcsRevisionKey`).

2. `retrieveSourceCode` method

The `retrieveSourceCode` updates the code to the `vcsRevisionKey` as returned from the previous method. If the `vcsRevisionKey` is null, then the method should check out to latest. The return value is what the source code was actually updated to.

Updates to the build processing plugins

The various pre and post build plugin points have been updated to to reflect the new build agent task structure.

- `com.atlassian.bamboo.build.CustomPreBuildAction`
- `com.atlassian.bamboo.build.CustomBuildProcessor`
- `com.atlassian.bamboo.build.CustomBuildProcessorServer`

Essentially the method:

```java
public void run(Build build, BuildResults buildResult);
```

has been replaced with the `BuildTask` interface.

```java
/**
 * Interface the defines a basic interface for a task in Bamboo. All [link `call()`] methods return
 * [link `BuildContext`] objects.
 *
 * public interface BuildTask extends Callable<BuildContext>
 * {
 *     void init(@NotNull BuildContext buildContext);
 *
 *     @NotNull
 *     BuildContext call() throws InterruptedException, Exception;
 * }
 */
```

You should be able to access most of the things you need through the `BuildContext` that can be retained from the `init` method.

Changes for Bamboo 2.1

Bamboo 2.1 included a rework of the Jira Integration functionality. This included some changes to the API's.

Moved Interfaces and Classes

Due to the addition of extra jira functionality the package structure has been updated for better understanding.

The package structure used to look like
It now has the following structure

```
com/atlassian/bamboo/jira
com/atlassian/bamboo/jira/jiraissues
com/atlassian/bamboo/jira/jirarun
com/atlassian/bamboo/jira/jirasearch
com/atlassian/bamboo/jira/jiraserver
com/atlassian/bamboo/jira/jirametadata
```

If you were relying on any of the jira classes you will need to update the packages.

**Jira Soap Client**

We updated the Jira Soap Client we use to the latest (Jira version 3.12). This code has now been separated out of the Bamboo source code into its own module. Maven should download this dependency for you.

**Changes to Class Functionality**

The `JiraServerDefinition` class is no longer responsible for talking to JIRA. It only holds the configuration information required to do so.

The `JiraRemoteIssueManager` class has had some of the functionality removed and lots of extra added. If you were using this class check out the javadocs for the updated functionality. [http://docs.atlassian.com/atlassian-bamboo/latest/com/atlassian/bamboo/util/JiraIssueUtils.html](http://docs.atlassian.com/atlassian-bamboo/latest/com/atlassian/bamboo/util/JiraIssueUtils.html)

**Changes for Bamboo 2.1.5**

**Plugin interface changes.**

`com.atlassian.bamboo.notification.NotificationCondition`

Declaration of the interface has been altered from this:

```java
public interface NotificationCondition extends BambooPluginModule
```

to this:

```java
public interface NotificationCondition extends BambooPluginModule, Comparable<NotificationCondition>
```

`com.atlassian.bamboo.notification.NotificationManager`

Declarations of 3 methods in the NotificationManager interface have been altered, from this:

```java
public NotificationRule createNotificationRule(NotificationSet notificationSet,
        String conditionKey, String conditionData,
        String emailRecipients, String imRecipients,
        String users, String groups, Set roles);

public NotificationRule updateNotificationRule(NotificationRule oldRule,
        String conditionKey, String conditionData,
        String notificationEmailString, String notificationIMString,
        String userArray, String groupArray, Set roles);

public List getAllConditions();
```

to this:

```java
```
Changes for Bamboo 2.2

Bamboo 2.2 includes a significant rework of Bamboo's notifications, brand new Elastic Bamboo as well as new plugin points and additional Remote API. Whilst we have tried very hard to ensure backwards compatibility this release, there may be unforeseen changes. If there is anything we have missed please let us know so we can keep this information up-to-date.

Notifications

We have deprecated the old NotificationCondition plugin point, though old notification plugins will still work. The replacement plugin points allow for much greater flexibility. You can find more information on building a new notification plugin here: Building a Notification Plugin.

We have also added the ability to customise your notifications by editing the Freemarker Templates.

Elastic Bamboo

Whilst adding in Elastic Bamboo functionality we have altered the way we deal with Agents and Capability sets. In particular the BuildAgent interface has lost the getCapabilitySet() method. Instead you retrieve CapabilitySets via the following:

```
CapabilitySet capabilitySet = CapabilitySetProvider.getAgentCapabilitySet(buildAgent);
```

or

```
CapabilitySet capabilitySet = CapabilitySetProvider.getAgentCapabilitySet(pipelineDefinition);
```

Variable Substitution Bean

To allow greater access to the VariableSubstitutionBean (e.g on remote/elastic agents) we have altered the interfaces in this class. See the javadocs for full details of the new API.

As a quick summary the following methods are no longer available

```
public String[] substituteBambooVariables(String[] commandLine, String buildPlanKey, BuildLogger buildLogger)

public String substituteBambooVariables(String unparsedString, String buildPlanKey, BuildLogger buildLogger)

public String substituteBambooVariables(@Nullable final String commandLine)

public String substituteBambooVariables(@Nullable final String commandLine, @Nullable final BuildContext buildContext, final BuildLogger buildLogger)

public String[] substituteBambooVariables(@NonNull final String[] commandLine, @NonNull finalBuildContext, @NonNull final BuildLogger buildLogger)

public String substituteBambooVariables(@NonNull final String commandLine, @NonNull final BuildResults buildResults, final BuildLogger buildLogger)
```

They have been replaced with the following options

```
public String substituteBambooVariables(@Nullable final String commandLine)

public String substituteBambooVariables(@Nullable final String commandLine, @Nullable final BuildContext buildContext, final BuildLogger buildLogger)

public String[] substituteBambooVariables(@NonNull final String[] commandLine, @NonNull final BuildContext buildContext, final BuildLogger buildLogger)

public String substituteBambooVariables(@NonNull final String commandLine, @NonNull final BuildResults buildResults, final BuildLogger buildLogger)

```

New Plugin Points
REST API

There have been a number of additions to our Remote API. The main ones include addition of methods to interact and manager your elastic instances. Remote API documentation is now being maintained here, rather than in the application.

- Bamboo Remote API

Changes for Bamboo 2.3

Bamboo 2.3 includes several improvements as well as new plugin points for developers. This page will be updated with details on these changes shortly.

**Remote API Changes in Bamboo 2.3**

Please note, we have made significant changes to Bamboo's remote API to improve it in Bamboo 2.3. However, it is likely that a number of existing Bamboo plugins will not work as a result.

We strongly recommend that you read through all of the changes listed below and update your plugins accordingly. If you encounter any issues that are not covered by the information on this page, please let us know by logging a ticket in the 'Bamboo' project on http://jira.atlassian.com/ and we will amend the information below, as required.

On this page:
- Method Deprecation for Builder Plugins
- Renamed Methods in BuildPlanDefinition

Method Deprecation for Builder Plugins

The following abstract method on the Command class has been deprecated

```
public Commandline getCommandLine(ReadOnlyCapabilitySet capabilitySet)
```

This will effect plugins who extend this class directly or extends the AbstractBuilder. Instead you should use the following methods

```
/**
 * What is the executable for the command line to execute. Variable substitution is not
 * required for executable, substitution will be performed later.
 * @param capabilitySet- for the current context
 * @return the executable as a string
 */
@NotNull
public String getCommandExecutable(ReadOnlyCapabilitySet capabilitySet)

/**
 * What are the arguments for the command line to execute Variable substitution is not required
 * for arguments, substitution will be performed later.
 * @param capabilitySet - for the current context
 * @return the arguments as a string[]. Spaces/division of strings in the array should not
 * matter.
 */
@NotNull
public String[] getCommandArguments(ReadOnlyCapabilitySet capabilitySet)
```

Renamed Methods in BuildPlanDefinition

The methods getBuilder() and getRepository() in the BuildPlanDefinition interface have been renamed to getBuilderV2() and getRepositoryV2() respectively, (see below). You will need to update these method calls in your plugins to be compatible with 2.3

```
@Nullable
RepositoryV2 getRepositoryV2();

BuilderV2 getBuilderV2();
```
Building a Bamboo War Distribution From Source

This guide describes building a `atlassian-bamboo-web-app.war` distribution from the Bamboo source code. Plugin developers who wish to use source code as an aid in building plugins should also refer to the Setting up Bamboo Development Environment in IDEA documentation.

**Building a war distribution**

You can download Bamboo Source code from our website if you have a Commercial License. If you do not have access to the source code download site, log in to my.atlassian.com as your billing contact or contact our sales department.

Bamboo is built using Maven. When you build Bamboo, Maven will download dependencies and store them in a local repository. Some of these dependencies require manual installation for legal distribution reasons. Maven will tell you, as you build, which dependencies it requires you to download.

**Coping with Sun JAVA libraries**

Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories. If you are developing plugins for Bamboo or building Bamboo from source, you might need `javax.mail` and `javax.transaction:jta:jar` for Bamboo to build successfully. Please visit our confluence page on Working with Sun JAVA libraries for further details.

1. Install Maven 2.0.7
2. Install JDK 1.5 (Doesn't work with 1.4).
3. Download the settings.xml attached to this page, to your `<USER_HOME>/m2/` directory.
4. In the `atlassian-bamboo` directory run `mvn clean package -Dmaven.test.skip=true -Pall`.

If the build is run successfully you should have a `atlassian-bamboo-web-app-*.war` file created in `./atlassian-bamboo/bamboo-web-app/target/`. If the build was unsuccessful, please take a look at our Troubleshooting Page for some more common errors otherwise create a support issue at http://support.atlassian.com and attach the full output from the script to your support request.

**Setting up Bamboo Development Environment in IDEA**

To get bamboo started, you need to acquire a Commercial License to gain access to Bamboo Source code. Once you have downloaded Bamboo source code, please follow the instructions below to set-up Bamboo in IDEA. If you have any problems check with our Troubleshooting Guide.

**Building Bamboo 2.2**

1. Install Maven 2.0.9
2. Install JDK 1.5 (Doesn't work with 1.4).
3. Download the settings.xml attached to this page, to your `<USER_HOME>/m2/` directory.
4. In the `atlassian-bamboo/components` directory run `mvn clean install -Dmaven.test.skip=true -Pall`.

**Coping with Sun JAVA libraries**

Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories. If you are developing plugins for Bamboo or building Bamboo from source, you might need `javax.mail` and `javax.transaction:jta:jar` for Bamboo to build successfully. Please visit our confluence page on Working with Sun JAVA libraries for further details.

5. In the `atlassian-bamboo` directory run `mvn idea:clean idea:idea -DdownloadSources=true -Pall`
6. This should generate `atlassian-bamboo.*` files in your source directory. Open the project with Idea.
7. Right click on `atlassian-bamboo-web-server` module (on the left hand panel under Projects) and click on module settings. Click on the dependencies tab and add `atlassian-bamboo-web-app` as a dependent module.
8. Add the application (Run -> Edit Configurations -> Add New Configuration of type "Application"):

<table>
<thead>
<tr>
<th>Main Class</th>
<th>com.atlassian.bamboo.server.Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Params</td>
<td>-Dorg.mortbay.xml.XmlParser.NotValidating=true -Dbamboo.home=home1 -server -Xmx512m -XX:MaxPermSize=128m</td>
</tr>
<tr>
<td>Program Params</td>
<td>8085 path-to-bamboo/web-app/src/main/webapp/</td>
</tr>
<tr>
<td>Working Dir</td>
<td>D:\dev\src\atlassian\bamboo-home</td>
</tr>
</tbody>
</table>
Building Bamboo 2.0.x & 2.1.x

1. Install Maven 2.0.7
2. Install JDK 1.5 (Doesn't work with 1.4).
3. Download the settings.xml attached to this page, to your <USER_HOME>/.m2/ directory
4. In the atlassian-bamboo directory run `mvn clean install -Dmaven.test.skip=true -Pall`.

5. **Coping with Sun JAVA libraries**
   Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories. If you are developing plugins for Bamboo or building Bamboo from source, you might need javax.mail and javax.transaction:jta:jar for Bamboo to build successfully. Please visit our confluence page on Working with Sun JAVA libraries for further details.

6. Run `mvn idea:clean idea:idea -DdownloadSources=true -Pall`.
7. This should generate atlassian-bamboo.* files in your source directory. Open the project with Idea.
8. Right click on atlassian-bamboo-web-server module (on the left hand panel under Projects) and click on module settings. Click on the dependencies tab and add all other modules except atlassian-bamboo-web-server. Thus, making the atlassian-bamboo module to be dependent on all the other modules except atlassian-bamboo-web-server.
9. Do the same for the atlassian-bamboo-web-server but make it depended only on the atlassian-bamboo module.
10. Add the application. Mine looks like:

    **Main Class**
    com.atlassian.bamboo.server.Server

    **VM Params**
    -Dorg.mortbay.xml.XmlParser.NotValidating=true -Dbamboo.home=home1 -server -Xmx512m -XX:MaxPermSize=126m

    **Program Params**
    8085 pathtobamboo/bamboo-web-app/src/main/webapp/

    **Working Dir**
    D:\dev\src\atlassian\bamboo-home

    **Class Path of Module**
    atlassian-bamboo-web-server

If you have an external instance of Bamboo running and you wish to debug it you can set up a remote debugger following these instruction. This will provide you with some command line arguments to add to your startup script. If you are using the Atlassian Plugin Archetype and running `mvn -Pplugin-debug` this already has those command line options included.

If you have any problems take a look at the troubleshooting page.

Building Bamboo 1.2.x

1. Install maven 2.0.7
2. Install JDK 1.5 (Doesn't work with 1.4).
3. Download the settings.xml attached to this page, to your <USER_HOME>/.m2/ directory
4. In the Bamboo directory run `mvn clean install -Dmaven.test.skip=true`.

5. **Coping with Sun JAVA libraries**
   Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories. If you are developing plugins for Bamboo or building Bamboo from source, you might need javax.mail and javax.transaction:jta:jar for Bamboo to build successfully. Please visit our confluence page on Working with Sun JAVA libraries for further details.

6. Run `mvn idea:clean idea:idea -DdownloadSources=true`.
7. This should generate atlassian-bamboo.* files in your source directory. Open the project with Idea.
8. Right click on atlassian-bamboo module (on the left hand panel under Projects) and click on module settings. Click on the dependencies tab and add all other modules except atlassian-bamboo-web-server. Thus, making the atlassian-bamboo module to be dependent on all the other modules except atlassian-bamboo-web-server.
9. Do the same for the atlassian-bamboo-web-server but make it depended only on the atlassian-bamboo module.
10. Add the application. Mine looks like:

    **Main Class**
    com.atlassian.bamboo.server.Server

    **VM Params**
    -Dorg.mortbay.xml.XmlParser.NotValidating=true -Dbamboo.home=home1 -server -Xmx512m -XX:MaxPermSize=126m
Atlassian Module Sources

The Bamboo Source Distribution also ships with the source for several of Atlassian's common modules. If you wish to view the source for these in IDEA:

1. Open Project settings
2. Find the corresponding dependency in amongst the bamboo modules (this is the fun part!) and click edit.
3. Select 'Attach Sources'
4. Locate the module source and select the directory above the root package (in Atlassian modules this usually is \src\main\java)

Now you should be able to navigate and debug into the source of that module.

Building Bamboo In IDEA - Troubleshooting

Below are some known errors and possible solutions.

Running the Maven Commands

Error:
Solution: You are missing the settings.xml file in your .m2 directory. You can find it attached to this page: Setting up Bamboo Development Environment in IDEA

Error:
The system is out of resources. Consult the following stack trace for details.
java.lang.OutOfMemoryError: Java heap space
at com.sun.tools.javac.util.Name.fromUtf (Name.java:84)
at com.sun.tools.javac.util.NameTable.fromUtf (Name.java:494)
at com.sun.tools.javac.util.ByteBuffer.toName (ByteBuffer.java:150)
at com.sun.tools.javac.jvm.ClassWriter.typeSig (ClassWriter.java:403)
at com.sun.tools.javac.jvm.ClassWriter.writeCode (ClassWriter.java:1025)
at com.sun.tools.javac.jvm.ClassWriter.writeMethod (ClassWriter.java:938)

Solution: Set your MAVEN_OPTS to increase your memory and rerun the failing command

```bash
export MAVEN_OPTS="-Xmx1024M -XX:MaxPermSize=512m"
```

Error:
Missing:
```
1) javax.mail:mail:jar:1.3.3.
```
Try downloading the file manually from:

Then, install it using the command:
```
mvn install:install-file -DgroupId=javax.mail -DartifactId=mail -Dversion=1.3.3 -Ppackaging=jar -Dfile=/path/to/file
```
Alternatively, if you host your own repository you can deploy the file there:
```
mvn deploy:deploy-file -DgroupId=javax.mail -DartifactId=mail -Dversion=1.3.3 -Ppackaging=jar -Dfile=/path/to/file -Durl=[url] -DrepositoryId=[id]
```
Path to dependency:
```
1) com.atlassian.bucket:atlassian-bucket:jar:0.17
2) javax.mail:mail:jar:1.3.3
```

Solution: Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories. You will need to download the javax.mail (version is displayed in the error) manually. Please visit our confluence page on Working with Sun JAVA libraries for further details.

In IDEA
Error:

Information: 0 warnings

Error: Compiling Java Module 'atlassian-bamboo-web-app': Deployment item

Error: Compiling Java Module 'atlassian-bamboo-web-app': Deployment item

Error: Compiling Java Module 'atlassian-bamboo-web-server': Deployment item

Solution:
If the offending module is the atlassian-bamboo-web-app module check the path to see where its looking for the web.xml file. If its any other module (e.g atlassian-bamboo-web-server) you can just delete the reference to the file. In you project configuration.

Bamboo FAQ

On this page:

- Bamboo FAQ
- Bamboo Evaluator's FAQ
- Need more help
Answers to commonly raised questions about configuring and using Bamboo:

- **What is Continuous Integration?**
- **Installation FAQ**
  - Adding MIME types to Bamboo Standalone
  - Bamboo 1.2 on Tomcat 5
  - Can I have Bamboo.home pointing to a resource on a network share?
  - Changing Bamboo's port from the default 8085
  - Changing the Root Context Path
  - Configuring Bamboo on start-up
  - Creating an external database for Bamboo
  - Getting Bamboo Standalone to use the jetty.xml file
  - Hardware sizing considerations
  - Installation notes for Bamboo on JBoss 4.x
  - Moving Bamboo between machines
  - Running Bamboo as a Service on Windows
  - Running Bamboo behind a firewall with Remote Agents outside the firewall
  - Running Bamboo over HTTPS
  - Running Bamboo service on Windows as the local user
  - Setting up JNDI on Jetty
  - Supported databases
  - Upgrading to Bamboo 2.0 with a datasource configured
- **Usage FAQ**
  - How do I manually set the version of new Subversion workspaces?
  - Securing your repository connection
  - Troubleshooting remote agent connectivity problems
  - Analysing Subversion Connectivity Issues
  - Backing up Bamboo instances over 4GB
  - Bamboo Database Schema
  - Binding Bamboo to one IP address
  - Can Bamboo build and test non-Java projects?
  - Can multiple plans share a common 3rd-party directory? — For example, you might have three repository directories, say, A, B, and C, where A is a common 3rd-party library. A is used across projects.
  - Changing automatic upgrades of working copies from Subversion
  - Changing Bamboo database settings
  - Changing the JIRA-Bamboo integration cache settings in Bamboo
  - Changing the remote agent heartbeat interval
  - Cloning a bamboo instance
  - Connecting to (embedded) HSQL Database using Dbvisualiser
  - CVS Error logging in Bamboo
  - Deactivating a Bamboo user
  - Debugging Subversion connection in Bamboo
  - Do I have to upgrade all remote agents for Bamboo Release 2.1.2?
  - Enable User Management debug logging in Bamboo
  - Fixing failing Bamboo builds, with OutOfMemory errors
  - Fixing OutOfMemory Errors in Bamboo
  - Hibernate errors in logs after upgrading to Bamboo 2.0
  - How do I construct a cron expression in Bamboo?
  - How do I disable SSH access to my elastic instances?
  - How do I shut down my elastic instances if I have restarted my Bamboo server?
  - How do I stop Bamboo from shutting itself down and restarting?
  - How do I stop the Bamboo server from automatically configuring my remote agent's capabilities?
  - JUnit parsing in Bamboo
  - Known issues with CVS in Bamboo 2.0
  - Monitoring and Profiling Bamboo
  - Monitor Memory usage and Garbage Collection in Bamboo
  - Moving Bamboo-Home of an agent
  - Performing a thread dump.
  - Problems running Bamboo under Sun JDK 1.4
  - Removing Coverage plug-in data from the Bamboo database
  - Restoring passwords to recover admin users
  - Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files
  - Testing LDAP or Active Directory connectivity with Paddle
  - Troubleshooting Elastic Bamboo
  - Troubleshooting SQL Exceptions - Detailed Hibernate Logging
  - Troubleshooting Subversion connection
  - Using Bamboo For Release Management
  - Using Bamboo with Clover
  - Why can't I SSH to my elastic instance?
  - Working with Sun JAVA libraries
**Bamboo Evaluator’s FAQ**

If you are evaluating Bamboo, you may also wish to consult the Bamboo Evaluator’s FAQ:

- Can Bamboo be Extended or Integrated with Other Tools?
- Can Bamboo be Used for Release Management?
- Can I use Clover Code Coverage with Bamboo?
- How is Bamboo Licensed?
- What are Remote and Elastic Agents?
- What are the Hardware Requirements for Bamboo?
- What Build Tools can Bamboo Work With?
- What Environments are Supported?
- What is Continuous Integration?
- Who Broke The Build?

**Need more help?**

Do you have a question, or need help with Bamboo? Please [create a support request](#).

Browse our Development FAQ

You may also like to check out the forums:

- Bamboo Announcements
- Bamboo General Forum
- Bamboo Developers Forum

**Installation FAQ**

- Adding MIME types to Bamboo Standalone
- Bamboo 1.2 on Tomcat 5
- Can I have Bamboo.home pointing to a resource on a network share?
- Changing Bamboo’s port from the default 8085
- Changing the Root Context Path
- Configuring Bamboo on start-up
- Creating an external database for Bamboo
- Getting Bamboo Standalone to use the jetty.xml file
- Hardware sizing considerations
- Installation notes for Bamboo on JBoss 4.x
- Moving Bamboo between machines
- Running Bamboo as a Service on Windows
- Running Bamboo behind a firewall with Remote Agents outside the firewall
- Running Bamboo over HTTPS
- Running Bamboo service on Windows as the local user
- Setting up JNDI on Jetty
- Supported databases
- Upgrading to Bamboo 2.0 with a datasource configured

**Adding MIME types to Bamboo Standalone**

Bamboo Standalone ships with the Jetty application server (see Bamboo 'distributions'). To add additional MIME content types, edit the mimetypes.xml file in `../<Bamboo-install>/webapps/WEB-INF/mimetypes.xml` and insert your new mime-mapping.

For instance to add an additional text/plain mime type with txt2 extension, insert the following:

```
<mime-mapping>
  <extension>txt2</extension>
  <mime-type>text/plain</mime-type>
</mime-mapping>
```

Restart Bamboo, for the changes to take effect. If you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your...
Bamboo 1.2 on Tomcat 5

If you are upgrading to Bamboo 1.2.2 on Tomcat 5, the endorsed XML libraries (Specifically, ‘xml-apis.jar’ and ‘xercesImpl.jar’) get loaded before Bamboo starts-up, which results in the exception below, as Bamboo uses the wrong XML libraries.

javax.xml.transform.TransformerFactoryConfigurationError: Provider
org.apache.xalan.processor.TransformerFactoryImpl not found
at javax.xml.transform.TransformerFactory.newInstance(Unknown Source)
at com.opensymphony.xwork.util.DomHelper.<cinit>(DomHelper.java:168)

at com.opensymphony.xwork.util.DomHelper.parse(DomHelper.java:116)

at com.opensymphony.xwork.config.providers.XmlConfigurationProvider.init(XmlConfigurationProvider.java:91)

at com.opensymphony.xwork.config.impl.DefaultConfiguration.reload(DefaultConfiguration.java:86)

at com.opensymphony.xwork.config.ConfigurationManager.getConfiguration(ConfigurationManager.java:55)

at com.opensymphony.xwork.DefaultActionProxy.<init>(DefaultActionProxy.java:60)


at com.opensymphony.webwork.dispatcher.DispatcherUtils.serviceAction(DispatcherUtils.java:216)

at com.opensymphony.webwork.dispatcher.FilterDispatcher.doFilter(FilterDispatcher.java:202)

at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:52)

at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)


at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at com.atlassian.bamboo.filter.BambooAcegiProxyFilter.doFilter(BambooAcegiProxyFilter.java:21)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at com.atlassian.bamboo.filter.LicenseFilter.doFilter(LicenseFilter.java:72)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at com.atlassian.bamboo.filter.RequestCacheThreadLocalFilter.doFilter(RequestCacheThreadLocalFilter.java:26)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:157)

at org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter.java:75)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:186)

at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:186)

at org.apache.catalina.core.StandardFilter.doFilter(StandardFilter.java:121)

at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:214)

at org.apache.catalina.core.StandardContext.invokeNext(StandardContext.java:104)

at org.apache.catalina.core.StandardContext.invoke(StandardContext.java:452)

at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:152)

at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:198)

at org.apache.catalina.core.StandardRequestContext.init(StandardRequestContext.java:104)
2007-08-14 12:10:30 ApplicationDispatcher[/atlassian-bamboo-1.2.2] Servlet.service() for servlet action threw exception
java.lang.NullPointerException

javax.servlet.ServletException
at com.opensymphony.webwork.dispatcher.DispatcherUtils.serviceAction(DispatcherUtils.java:236)

To fix this issue:

1. Stop your Tomcat5 application server.
2. Remove the 'xml-apis.jar' and 'xercesImpl.jar' from "../<Tomcat_install>/common/endorsed/".
3. Start Tomcat5, and upgrade to 1.2.2.

The error above seems to be persistent with Bamboo 1.2.2, and doesn't affect any other versions.

Should you have problems upgrading, please raise a support request at https://support.atlassian.com/

Can I have Bamboo.home pointing to a resource on a network share?

Yes, it is possible to point Bamboo.home to a directory on your Windows network share. However, you need to specify the absolute path to the network share (e.g. \Network_computer\path_to_directory).

Note:

Please ensure that the user which Bamboo is running as has sufficient privileges to access the network resource.

Changing Bamboo’s port from the default 8085

To configure Bamboo to start on a port other than 8085, please see Configuring Bamboo on start-up.

Note: If you are running Bamboo on Linux, you need to start Bamboo on a privileged port (0-1024). You also need to run the Bamboo process as root.

Changing the Root Context Path

When running Bamboo behind a proxy, you might need to change the Root Context Path i.e. the host URL referenced while accessing Bamboo.
(e.g. http://localhost:8085/bamboo).

To change the context path from '/' to '/Your_Context_Path':

- **If you are using the bamboo.sh script to start Bamboo:**
  Change the following line in your bamboo.sh script:

  ```bash
  RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m -Djava.awt.headless=true
  -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true
  -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /*
  
  to:
  
  ```bash
  RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m -Djava.awt.headless=true
  -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true
  -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /Your_Context_Path"

- **Or, if you are using the wrapper to start Bamboo:**
  The wrapper reads the configuration information from the wrapper.conf file in the .//<Bamboo-Install>/conf/ folder. Find the following line:

  ```bash
  wrapper.app.parameter.4=/
  
  Replace it with the following line:
  
  ```bash
  wrapper.app.parameter.4=/Your_Context_Path

**Configuring Bamboo on start-up**

**Restarting Bamboo**

After editing the options below, Bamboo needs to be shut down and restarted for the changes to take effect. If you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server.

- Configuring Bamboo's start-up parameters under Linux
- Configuring Bamboo's start-up parameters under Windows

**Configuring Bamboo’s start-up parameters under Linux**

Bamboo on Linux/Unix can be started by either executing the bamboo.sh script or using the wrapper. Either way, the Bamboo server can be customised at start-up.

**Modifying the bamboo.sh script.**

The bamboo.sh script takes four parameters: start|stop|restart|status.

To customise these parameters at startup, edit the $RUN_CMD variable:

```bash
# This is how the Bamboo server will be started
#RUN_CMD="java -server -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH 
-Dirb.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /*"
```

- `java -Xms256m -Xmx512m` specifies the minimum and maximum Java Heap size.
- `-classpath $CLASSPATH` sets the class path at startup.
- `-Djetty.port=8085` specifies the port number for the Jetty server.
• **com.atlassian.bamboo.server.Server 8085 ./webapp** / is the main class that will be executed followed by the context path. In some cases it might be useful to increase the PermGen space. To do this, add the following parameter to the RUN_CMD variable: "XX:MaxPermSize=512m". This will set the PermGen space to 512mb next time Bamboo is run.

**Modifying the wrapper.**

The wrapper reads the configuration from wrapper.conf found in ../<BAMBOO_INSTALL>/conf. (The properties are documented inside the file.)

**Configuring Bamboo’s start-up parameters under Windows**

Bamboo can be started in Windows with the startup.bat file (from the command line) or as a Windows Service. Both use the wrapper to start Bamboo. As in Linux (see above), the wrapper reads the configuration from wrapper.conf. Please edit the .../wrapper(wrapper.conf file (situated in the root of your Bamboo_installation directory) as required.

**Creating an external database for Bamboo**

In case you wish to configure an external database you will need to create it before connecting bamboo to it.

**Mysql**

```sql
mysql> create database bamboodb character set utf8;
mysql> grant all privileges on bamboodb.* TO 'bamboouser'@'localhost' IDENTIFIED BY 'password';
mysql> flush privileges;
mysql> quit
```

**Postgres**

T.B.C.

**Oracle**

T.B.C.

**Getting Bamboo Standalone to use the jetty.xml file**

By default Bamboo doesn't use the jetty.xml file to configure itself. If you need to modify the jetty.xml for advanced configuration (such as JNDI or https), you will also need to tell Bamboo to use it.

The method for doing this depends on whether you are using the bamboo.sh startup script or the Java Service Wrapper.

**Step 1 - Instructing Bamboo to use jetty.xml**

*If you are using the bamboo.sh script to start Bamboo:*

The standard Bamboo startup script can be customised to use the jetty.xml file by modifying the following section in your bamboo.sh script (this section specifies how the Bamboo server will start):

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /",
```

Now, modify this startup script to read the jetty.xml file from webapp/WEB-INF/classes/jetty.xml by changing the RUN_CMD argument as follows:

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085 com.atlassian.bamboo.server.Server webapp/WEB-INF/classes/jetty.xml",
```

*If you are using the Java Service Wrapper to start Bamboo:*

When starting up Bamboo with the Java Service Wrapper, you'll need to modify the wrapper.conf file in the conf directory.

- You will need to replace the argument which specifies your port number "wrapper.app.parameter.2=8085" with "wrapper.app.parameter.2=../webapp/WEB-INF/classes/jetty.xml".
- You will need to comment out the other arguments: "wrapper.app.parameter.3=../webapp" and "wrapper.app.parameter.4=7"
This will make Bamboo start up using your jetty.xml configuration file instead of the default three arguments (port, web app directory, context path).

**Step 2 - Setting root context web application in jetty.xml**

**If you are using Bamboo 1.2.4 follow the instructions below:**

Edit the webapp root context in your jetty.xml file situated in `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml`. From

```
<Call name="addWebApplication">
  <Arg>/bamboo</Arg>
  <Arg>
    <SystemProperty name="bamboo.webapp" default="bamboo-web-app/src/main/webapp"/>
  </Arg>
</Call>
```

To

```
<Call name="addWebApplication">
  <Arg>/bamboo</Arg>
  <Arg>
    <SystemProperty name="bamboo.webapp" default="full/path/to/bamboo/install/directory/webapp"/>
  </Arg>
</Call>
```

**If you are using Bamboo 2.0 follow the instructions below:**

Replace your existing `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml` file with this jetty.xml file.

**Hardware sizing considerations**

Also see the Bamboo System Requirements.

For Bamboo, the minimum hardware requirements depend on the size and complexity of your plans. Considerations include:

1. Will your builds have functional tests as part of the plans?
2. Are your plans executed simultaneously? If so, how many plans will be running at any given time?
3. What are the requirements for your running builds, e.g. do they need large amounts of memory/disk/swap space?
4. How many users will be using Bamboo at any given time? Like any web application, the system resource needed is proportional to the load experienced by the server.

**Installation notes for Bamboo on JBoss 4.x**

This page is for people who are deploying the Bamboo EAR/WAR edition on the JBoss 4.x application server.

For full installation instructions please see the Bamboo EAR-WAR Installation Guide.

**File extraction notes**

To deploy Bamboo EAR-WAR onto your JBoss application server, copy the Bamboo WAR file to `../<JBoss-install>/server/default/deploy/atlassian-bamboo-1.1.2.war`.

By default the WAR file will extract to atlassian-bamboo-<version>. The name of the directory in the webapps folder will form the URL required to access Bamboo, e.g. `<JBoss-install>/server/default/deploy/atlassian-bamboo-1.1.2.war` will become `http://host:port/atlassian-bamboo-1.1.2`.

**How to set Java OPTs on JBoss 4.x**
• Windows:
  1. Find the run.bat file.
  2. Edit JAVA_OPTS to set the desired properties variable:

```
if exist "%JBOSS_HOME%\bin\native" set JAVA_OPTS=%JAVA_OPTS%
-Djava.library.path=%JBOSS_HOME%\bin\native
```

TO

```
if exist "%JBOSS_HOME%\bin\native" set JAVA_OPTS=%JAVA_OPTS%
-Djava.library.path=%JBOSS_HOME%\bin\native
-server -XX:MaxPermSize=256m
-Dbamboo.home=/opt/bamboo/bamboohome
-Xmx512m -Djava.awt.headless=true
-Dbamboo.home=Your_Path_To_Bamboo
```

• Linux-based systems:
  1. Find the run.sh file
  2. Edit JAVA_OPTS to set the desired properties variable:

```
# Setup JBoss specific properties
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
```

For further reference
Please visit the JBoss Wiki page on setting JavaOpts

Moving Bamboo between machines

To migrate Bamboo between machines,

1. Back up your current Bamboo instance as per our Bamboo backup guide
2. Login to the My Atlassian portal and retrieve your Bamboo license key.
3. Download a fresh distribution of Bamboo from the Bamboo download center to your new machine.
   - If you are using the WAR version, click the 'Show All' link on the Bamboo download center page to display the download link.
4. Unpack your Bamboo distribution on your new machine. (if you are using the WAR version, you need to deploy it to your application server) - See our installation guide.
5. Start your new Bamboo installation and run the Setup Wizard.
6. In step (3) of the Setup Wizard, import your old Bamboo instance from step (1).

Running Bamboo as a Service on Windows

Once you have installed Bamboo, you can choose to run Bamboo as service so that it starts up every time Windows restarts.

To do this,

1. Click on the Start menu in Windows.
2. Select Bamboo from the programs list.
3. Click on Install Service option to install Bamboo as a service in Windows.
4. Click Start Service to start the service.
Running Bamboo as local user

Bamboo service installs itself as the windows NT SYSTEM User, to run Bamboo as a local user follow this documentation.

Upgrading Bamboo server

If you have just upgraded your Bamboo server, you must re-install the Bamboo service. You can do this by removing the service and installing it again.

Running Bamboo behind a firewall with Remote Agents outside the firewall

Please note, this information does not apply to elastic agents.

The traffic between agent and server is bi-directional, however initial Requests are only sent from agent to server. Responses are sent from server to agent.

1. bamboo base url, usually http(s)://<host>:8085/<bamboo-app-context>
2. bamboo broker url, usually tcp://<host>:54663 if not defined otherwise

Both ports need to be open in the firewall. The direction of the initial request is from agent to server.

The remote agent has 2 arbitrary ports open to serve both connections.

Running Bamboo over HTTPS

This document is a guide to configuring Bamboo Standalone with basic HTTPS authentication. For further reference please visit the Jetty page on configuring SSL with Jetty.

1. Generating a certificate with the JDK keytool

The simplest way to generate keys and certificates is to use the keytool application that comes with the JDK, as it generates keys and certificates directly into the keystore.

The following command will generate a key pair and certificate directly into a keystore:

```bash
keytool -keystore keystore -alias jetty -genkey -keyalg RSA
```

This command will prompt for information about the certificate and for passwords to protect both the keystore and the keys within it. The only mandatory response is to provide the fully qualified host name of the server at the "first and last name" prompt.

Now, we need to configure an SSL listener.
2. Configuring Jetty

Using the Sun JVM, add the SunJsseListener as a HttpListeners. In the ../<Bamboo_Application_Directory>/webapp/WEB-INF/classes/jetty.xml file add the following lines.
This will make Bamboo accessible in port 8443 on https://localhost:8443/

**If you are using Bamboo 1.2.4 (or earlier)**

```xml
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
      <Set name="Port">8443</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>/keystore</Set>
      <Set name="Password">password</Set>
      <Set name="KeyPassword">password</Set>
    </New>
  </Arg>
</Call>
```

**If you are using Bamboo 2.0**

```xml
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.security.SslSocketConnector">
      <Set name="Port">8443</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>/keystore</Set>
      <Set name="Password">password</Set>
      <Set name="KeyPassword">password</Set>
    </New>
  </Arg>
</Call>
```

The keystore file in this example is given relative to the Bamboo Application Directory.

Please ensure that jcert.jar, jnet.jar and jsse.jar are on your classpath.

3. Getting Bamboo to use the jetty.xml file

Follow this Knowledge Base article, to instruct Bamboo to use the jetty.xml file configured in step 2.

**Running Bamboo service on Windows as the local user**

1. **Install Bamboo Application Server**
   1. Download Bamboo and run the Setup Wizard.
   2. Install Bamboo as Windows service, as described in the Bamboo Standalone Installation Guide (Windows).

2. **Edit the Bamboo service to run as the "local user"**
   1. Go to Start -> Run and enter 'services.msi'.
   2. The ‘Services’ window will display (see screenshot below). Double-click the 'Bamboo build server' row.
3. The 'Bamboo build server Properties' window will display (see screenshot above). Select the **This account** option and click the **OK** button to apply your changes.

3. **Give the local user access to "logon as a service"**

1. Go to Start -> Run and enter `secpol.msc`
2. The 'Local Security Settings' window will display. Expand the 'Local Policies' tree and click 'User Rights Assignment'.
3. Scroll down and find the 'Logon As a Service' Policy (see screenshot below). Double-click the 'Log on as a service' policy.

4. The properties window for the 'Log on as a service' policy will display (see screenshot below). Click the 'Add User or Group' button.
5. The ‘Select Users or Groups’ window will display (see screenshot above). Enter your local user and click ‘OK’ to allow your user to “logon as a service”.
6. Click ‘OK’ and close all open windows.

Bamboo will now start as service, under the local user.

**Setting up JNDI on Jetty**

The Bamboo start up script can be customised to setup JNDI resources

Follow this guide to setup Bamboo to use the jetty.xml file

You will also need to change the jetty.xml file under webapp/WEB-INF/classes by change the context path from /bamboo to /. Example of this is below:

**If you are using Bamboo 1.2.4 (or earlier):**

```xml
<!-- Add root context web applications. -->
<!-- Add root context web applications. -->
<Call name="addWebApplication">
  <Arg></Arg>  
  <Arg>
    <SystemProperty name="bamboo.webapp" default="webapp"/>
  </Arg>
</Call>
```

**If you are using Bamboo 2.0:**


To set up the JNDI mail session, you will also need to uncomment and modify the section of this jetty.xml shown below. You will need to replace the values inside the <Arg> tags with appropriate values (username, password, host, from address).

**In Bamboo 1.2.4 and earlier:**

```
<Call name="addHandler">
  <Arg>
    <New class="org.mortbay.jetty.webapp.WebAppContext">
      <Arg name="webApp">
        <SystemProperty name="bamboo.webapp" default="webapp"/>
      </Arg>
      <Arg name="contextPath"/>
    </New>
  </Arg>
</Call>
```

**In Bamboo 2.0:**

```
<!--
<Call name="addService">
  <Arg>...
</Arg>
</Call>
-->
```

If you are experiencing class loading problems with your mail session. Try uncommenting the following line in the web applications context (2.0 only):

```
<!--
<New id="resourceID" class="org.mortbay.jetty.plus.naming.Resource">
  <Arg>resourceName</Arg>
  <Arg>
    <New class="org.mortbay.naming.factories.MailSessionReference">
      ...
    </New>
  </Arg>
</New>-->
```

**Supported databases**

Bamboo is bundled with a pre-configured HSQL database for evaluation purposes only.

We recommend migrating to a external database, if Bamboo is to be used in production. For a list of supported databases, please see Connecting Bamboo to an external database.

**Upgrading to Bamboo 2.0 with a datasource configured**

The Bamboo upgrade tasks will fail if you are using a datasource to connect to your external database. To work around this issue please follow the procedure below:

1. Navigate to your Bamboo-Home root directory.
2. Execute the following command:
   ```
   cp bamboo.cfg.xml bamboo.cfg.xml.bak
   ```
3. Edit the `bamboo.cfg.xml` file in your root `Bamboo-Home` directory.
4. Find the following line:

   ```xml
   <property name="hibernate.connection.datasource">jdbc/bamboo</property>
   ```

5. Replace the line above with:

   ```xml
   <property name="hibernate.connection.datasource">jdbc/bamboo</property>
   <property name="hibernate.c3p0.acquire_increment">1</property>
   <property name="hibernate.c3p0.idle_test_period">100</property>
   <property name="hibernate.c3p0.max_size">15</property>
   <property name="hibernate.c3p0.max_statements">0</property>
   <property name="hibernate.c3p0.min_size">0</property>
   <property name="hibernate.c3p0.timeout">30</property>
   <property name="hibernate.connection.driver_class">YOUR_DRIVER_NAME</property>
   <property name="hibernate.connection.password">YOUR_DATABASE_PASSWORD</property>
   <property name="hibernate.connection.url">YOUR_CONNECTING_URL</property>
   <property name="hibernate.connection.username">YOUR_DATABASE_USERNAME</property>
   ```

**Note:** Replace `YOUR_DRIVER_NAME`, `YOUR_DATABASE_PASSWORD`, `YOUR_CONNECTING_URL` and `YOUR_DATABASE_USERNAME` with the driver, password, url and username of your database respectively.

6. Once you have successfully upgraded to Bamboo 2.0, remove your new `bamboo.cfg.xml` file and move your old `bamboo.cfg.xml.bak` from step (2) to `bamboo.cfg.xml`.

**Usage FAQ**

- How do I manually set the version of new Subversion workspaces?
- Securing your repository connection
- Troubleshooting remote agent connectivity problems
- Analysing Subversion Connectivity Issues
- Backing up Bamboo instances over 4GB
- Bamboo Database Schema
- Binding Bamboo to one IP address
- Can Bamboo build and test non-Java projects?
- Can multiple plans share a common 3rd-party directory?
- Changing automatic upgrades of working copies from Subversion
- Changing Bamboo database settings
- Changing the JIRA-Bamboo integration cache settings in Bamboo
- Changing the remote agent heartbeat interval
- Cloning a bamboo instance
- Connecting to (embedded) HSQL Database using Dbvisualiser
- CVS Error logging in Bamboo
- Deactivating a Bamboo user
- Debugging Subversion connection in Bamboo
- Do I have to upgrade all remote agents for Bamboo Release 2.1.2?
- Enable User Management debug logging in Bamboo
- Fixing failing Bamboo builds, with OutOfMemory errors
- Fixing OutOfMemory Errors in Bamboo
- Hibernate errors in logs after upgrading to Bamboo 2.0
- How do I construct a cron expression in Bamboo?
- How do I disable SSH access to my elastic instances?
- How do I shut down my elastic instances if I have restarted my Bamboo server?
- How do I stop Bamboo from shutting itself down and restarting?
- How do I stop the Bamboo server from automatically configuring my remote agent's capabilities?
- JUnit parsing in Bamboo
- Known issues with CVS in Bamboo 2.0
- Monitoring and Profiling Bamboo
- Monitor Memory usage and Garbage Collection in Bamboo
- Moving Bamboo-Home of an agent
- Performing a thread dump.
- Problems running Bamboo under Sun JDK 1.4
- Removing Coverage plug-in data from the Bamboo database
- Restoring passwords to recover admin users
- Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files
- Testing LDAP or Active Directory connectivity with Paddle
- Troubleshooting Elastic Bamboo
- Troubleshooting SQL Exceptions - Detailed Hibernate Logging
- Troubleshooting Subversion connection
- Using Bamboo For Release Management
- Using Bamboo with Clover
- Why can't I SSH to my elastic instance?
How do I manually set the version of new Subversion workspaces?

You can manually set the version of any new Subversion workspaces created by Bamboo on checkout. For example, if you use a pre-1.5 Subversion client to access code checked out by Bamboo (see this FAQ), you will need to force any new Subversion workspaces to be created with a version of 1.4.

To manually set the version of any new Subversion workspaces created by Bamboo, configure the following system parameter:

```
.bamboo.svn.wc.format
```

Valid values for this parameter are 1.3, 1.4, 1.5 and 1.6.

⚠️ Please note, setting this parameter will only affect any Subversion workspaces created after the parameter has been set. It will not change the version of any Subversion workspaces that have already been created.

To change this parameter for your Bamboo instance, you can follow either one of the methods below:

- Add the parameter with a `-D` prefix and appropriate value, in your command line when starting Bamboo. e.g. `-Dbamboo.svn.wc.format=1.4`, or
- Add the parameter as a property in your `<bamboo-home>/bamboo.cfg.xml` file. e.g. `<property name="bamboo.svn.wc.format">1.4</property>`

Securing your repository connection

About this page

This page shows how to secure your bamboo server to source repository connection.

Subversion

**svn+ssh**

In your build plan you must specify the absolute path to the repository when using `svn+ssh`, for example

```
svn+ssh://<svnhost>/absolute/path/to/repository/root/your/module
```

Using a key pair

They key pair is shared between your bamboo agent box (the bamboo server box in case of local agents) and the repository server box. Your repository configuration allows you to specify the location of a private key file that must be stored on the agent box.

The key pair has to be in PKCS12/OpenSSH format and the private key must be passphrase protected, otherwise a runtime exception is thrown by JDK security engine while opening the user key.

**Linux and related**

1. On the repository box generate the keypair

```
ssh-keygen -t rsa
```

2. add public key to `~/.ssh/authorized_keys`

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

3. copy the private key to all the agent boxes **into a directory that is common to all** agents (remote and local) e.g. `/var/keys/ssh/id_rsa`
For windows agents
Store the private key file in the same location on the drive that the agent is started from. For example you start your agent with

```
d:\bamboo-agent > java -jar atlassian-bamboo-agent-installer-xxx.jar ....
```

Then the key file must be in d:\var\keys\ssh\id_rsa

**Windows**

Private key should always be in OpenSSH format. On windows usually "putty" (plink) program is used that uses keys in its proprietary format (PPK - putty private key), this format is not supported by bamboo. The PuttyGen program may be used on Windows to convert key in PPK format to OpenSSH.

How to add the public key to the windows version of ~/.ssh/authorized_keys

**Troubleshooting remote agent connectivity problems**

This page intends to give tips on troubleshooting the most common connectivity problems with remote agents

**Configuration**

t.b.c.

**Remote Agent Registration**

A good way to start troubleshooting is to enable debug logging for the agent - server communication:

- Stop bamboo
- copy the attached log4j.properties file into <bamboo-install>/webapp/WEB-INF/classes
- restart bamboo and restart the remote agent

After restart useful debug traces are produced that show the agent-server communication in detail. Please use these logs to support your inquiries with Atlassian Support.

**Disconnect because of Timeouts**

t.b.c.

**Analysing Subversion Connectivity Issues**

Bamboo ships with a SVNkit java command-line client from Bamboo 2.2.4 onwards. The SVNkit java command-line client effectively functions like the Subversion command-line client except that it works with the Java libraries in Bamboo.

To run the SVNkit java command-line client, follow the steps below:

- **Linux/Unix:**
  1. Navigate to <BAMBOO_INSTALL>/scripts
  2. Update the permissions on the jsvn script by running the following command:
     ```
     chmod 777 jsvn
     ```
  3. Run the jsvn script. You can use ./jsvn --help or any commands you would use with the native svn client.

- **Windows:**
  1. Navigate to <BAMBOO_INSTALL>/scripts
  2. Run the jsvn.bat script.

For instructions on how to use the Subversion command-line client, please refer to the Subversion Command-Line Client documentation (v1.5).

**Back up Bamboo instances over 4GB**
Due to limitations of the original ZIP file format, and the TrueZIP library used to generate ZIP files, it is not possible to export a Bamboo instance when the resulting ZIP file, or the original size of any of its components, is larger than 4GB. Instead, you will need to backup Bamboo manually. We strongly recommend performing regular backups.

To backup Bamboo manually:

1. Shut down Bamboo.
2. Copy the contents of your Bamboo-Home directory.
3. If you are using an external database, use the database's native backup tool to backup your database (please consult your database documentation for further instructions). Alternatively, perform an SQL dump of your database.

To restore your Bamboo instance to a previous state:

1. Edit the ..../Bamboo-Install-Directory/webapps/WEB-INF/bamboo.init.properties file to point to your backed-up Bamboo-Home directory.
2. If you are using an external database, restore your database using the database's native backup tool.

Bamboo Database Schema

Bamboo 2.1 Database schema

Binding Bamboo to one IP address

These instructions apply to Bamboo Standalone, which ships with the Jetty application server.

If you have installed Bamboo on a machine with multiple interfaces, and need to bind Bamboo to a single IP address, follow these instructions.

Step 1 — Instruct Bamboo to read its configuration from the jetty.xml file

By default Bamboo doesn't use the jetty.xml file to configure itself. You will need to tell Bamboo to use it.
**Step 2 — Edit the Jetty.xml file**

Your `Jetty.xml` file is located in `<Bamboo_Install_directory>/webapp/WEB-INF/classes/jetty.xml`. Please note: YOUR_HOST_URL should be the same as the Bamboo base URL configured in Bamboo.

If you are using Bamboo 1.2.4:

Find the following section:

```xml
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">127.0.0.1</Set>
    </New>
  </Arg>
</Call>
```

Change the last line as follows:

```xml
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">YOUR_HOST_URL</Set>
    </New>
  </Arg>
</Call>
```

If you are using Bamboo 2.0

Find the following section:

```xml
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.bio.SocketConnector">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">127.0.0.1</Set>
    </New>
  </Arg>
</Call>
```

Uncomment the host property as follows:

```xml
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.bio.SocketConnector">
      <Set name="Port">
        <SystemProperty name="jetty.port" default="8085"/>
      </Set>
      <Set name="Host">YOUR_HOST_URL</Set>
    </New>
  </Arg>
</Call>
```

**Step 3 — Restart Bamboo**

If you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server.

Can Bamboo build and test non-Java projects?

Bamboo can be ported to be used on any architecture and can build projects in virtually any language/script (Java, C++, ruby, perl, VB.net, bash, make and C# to name a few of many projects currently built with Bamboo).

Bamboo can execute any script/build that has a return code after the build process is completed. Ideally, you would configure a build tool (such as Maven or Ant) to build your code. Bamboo will then call on the build tool to build your project (depending on how your build process is configured).
Regarding tests, Bamboo uses JUnit tests to integrate test results with Java and is capable of reading test results from any testing framework that outputs to a JUnit XML report.

Can multiple plans share a common 3rd-party directory?

For example, you might have three repository directories, say, A, B, and C, where A is a common 3rd-party library. A is used across projects. At this stage, Bamboo doesn't support having multiple checkout directories per build plan. However, you can work around this by setting these three directories up as separate Bamboo build plans - P_A, P_B and P_C.

To make this work, you will also need to specify as an argument to your build scripts for P_B and P_C the location of A, which will be something like this: `../Plan_key_for_A/`

Using a set up like this, your library module (A) should only be checked out once across the Bamboo instance.

See also:
Triggering a Build when another Build finishes

Changing automatic upgrades of working copies from Subversion

The SVNKit library in Bamboo was upgraded to version 1.2, in Bamboo 2.1.4.

This has two implications for source code checked out by Bamboo:

- any source code checked out by Bamboo will be automatically upgraded to be compatible with Subversion 1.5 and 1.6.
- (Bamboo 2.1.4 and earlier) any working copies already checked out are automatically upgraded to be compatible with Subversion 1.5.

This does not adversely affect any pre-1.5 Subversion servers. However, if you use a pre-1.5 Subversion client to access code checked out by Bamboo, then any Bamboo builds on that code may fail. Please refer to BAM-3241 for further details.

You can configure Bamboo to prevent it from automatically upgrading any source code checked out, by configuring the following parameter:

```
<property name="bamboo.svn.wc.format">1.4</property>
```

Valid values for this parameter are 1.3, 1.4, 1.5 and 1.6

Setting this parameter to 1.4 tells Bamboo to

- check out code to version 1.4 if no working copy exists
- not to automatically upgrade any already checked out code of an existing working copy to be compatible with Subversion 1.5.

To change this parameter for your Bamboo instance, you can follow either one of the methods below:

- Add the parameter with a `-D` prefix and appropriate value, in your command line when starting Bamboo. e.g.
  `-Dbamboo.svn.wc.format=1.4`
- Add the parameter as a property in your `<bamboo-home>/bamboo.cfg.xml` file.
  e.g. `<property name="bamboo.svn.wc.format">1.4</property>`

Changing Bamboo database settings

The Bamboo database configuration is persisted in the `<Bamboo-Home>/bamboo.cfg.xml` file. You can change the database settings by editing this file, as detailed in the instructions below:

Changing the Bamboo database username and password.

If you want to change the database username and password, edit the following line,

```
<property name="hibernate.connection.password">YOUR_PASSWORD</property>
<property name="hibernate.connection.username">YOUR_USERNAME</property>
```

Changing the Bamboo database URL

If you want to change the database URL, edit the following line,

```
<property name="hibernate.connection.url">DATABASE_URL</property>
```

⚠️ You need to restart the Bamboo application server for the changes to take effect. If you have any elastic agents running, ensure that they are
shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server.

Changing the JIRA-Bamboo integration cache settings in Bamboo

If you have integrated Bamboo with JIRA, you may wish to change the JIRA-Bamboo integration cache settings to tweak performance. Bamboo caches JIRA data when it retrieves issue data, to prevent repeat calls to JIRA from degrading performance.

Bamboo caches the following JIRA information:

- **Issue Details** — a summary representation of the JIRA information (including the Type, Status and Summary) for each issue is stored by Bamboo.
- **Issue Meta-Data** — the reference information (e.g. Status codes) used to translate the summary representation of JIRA information into display values is also cached by Bamboo.

The caching behaviour for the Issue Details and Issue Meta-Data is described below:

- **Issue Details** — Issue Details are usually cached by Bamboo when the issue is first accessed by Bamboo (although some background synchronisation of data does occur). The cached data is stored for 86,400,000 ms (i.e. 24 hours) before it is cleared.
- **Issue Meta-Data** — Issue Meta-Data is cached in the background by synchronisation tasks, rather than waiting until the information is first accessed. The cached data is stored for 864,000,000 ms (i.e. approximately 10 days).

This caching behaviour is designed to optimise system performance against the currency of information, but if you wish to change the default behaviour (e.g. you need to keep up to date with rapidly changing JIRA data and can tolerate reduced Bamboo performance) you can do so by modifying the following timeout values via the system properties:

- `bamboo.jira.issueCache.timeToLive`
- `bamboo.jira.metaDataCache.timeToLive`

You can do this by adding the timeout values as command line parameters when starting Bamboo.

E.g.
- `DBamboo.jira.issueCache.timeToLive=86400000`
- `DBamboo.jira.metaDataCache.timeToLive=864000000`

Changing the remote agent heartbeat interval

Remote agents periodically send a "heartbeat" signal to the Bamboo server. This is vital for tracking whether your remote agents are online or offline. The remote heartbeat is **asynchronous**, which means that if a remote agent goes offline and comes back online again it will reconnect instead of being shut down (as long as the same server is available).

However, you may wish to adjust the time parameters for the remote agent heartbeat, particularly if you have a lot of network activity already.

You need to be running **Bamboo 2.0.6** or above to adjust the following remote agent heartbeat parameters.

There are three configurable parameters on the bamboo server for the remote agent heartbeat:

- **bamboo.agent.heartbeatInterval** — This parameter governs the frequency of the heartbeat signal from the remote agents. This parameter is specified in seconds with the default being 5 **seconds**.
- **bamboo.agent.heartbeatTimeoutSeconds** — This parameter governs how long the Bamboo server will wait before it times out an agent that it hasn’t received a heartbeat signal from. A remote agent that has been timed out will be marked as ‘Offline’. Any builds being run by agents which have timed out will be abandoned. This parameter is specified in seconds with the default being 600 **seconds**.
- **bamboo.agent.heartbeatCheckInterval** — This parameter governs how often Bamboo checks for agents that have exceeded the heartbeat timeout specified in `bamboo.agent.heartbeatTimeoutSeconds`. This parameter is specified in seconds with the default being 30 **seconds**.

Please read the Configuring System Properties page for instructions on how to change a remote agent heartbeat parameter for your Bamboo server.

Cloning a bamboo instance

In case you need to clone your production instance to a test/staging instance in order to prepare migrating to another database or upgrading bamboo.

For example, you may want to transfer your current production snapshot to a test server as permitted in the license agreement.

We strongly recommend to duplicate bamboo first and then apply changes such as upgrade or migrating to another database.
License

Development licenses are available for any Commercial or Academic license. Create one or contact us for help.

Clone Production Instance - Standard

This is the simple and straightforward way to clone your instance

1. Export/Backup your current instance
2. Copy zip across to new server.
3. Install the same version of bamboo on new server
4. Point bamboo.home in "<bamboo-install>/webapp/WEB-INF/classes/bamboo-init.properties" to your new bamboo home directory
5. Start the new instance
6. Complete the setup wizard, choose 'import existing data'.

Clone Production Instance - Alternative

If your current instance has grown too large and export/import does not work you can still clone your instance using an alternative backup and restore strategy.
The purpose is to clone <bamboo-home> and make it available to the new test/clone instance.

1. shutdown production bamboo at a convenient time
2. Create a backup

<table>
<thead>
<tr>
<th>embedded DB</th>
<th>external DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• zip &lt;bamboo-production-home&gt; directory, the embedded database will be part of this zip.</td>
<td>zip &lt;bamboo-production-home&gt; directory, create a backup with the native tools provided by your DB.</td>
</tr>
</tbody>
</table>

3. (Optional) reduce the size of this zip by deleting the xml-data/build-dir - this directory only contains working copies of checked out sources.
4. restart production bamboo
5. transfer the home.zip to your cloned instance and unzip
6. edit <bamboo-clone-home>/bamboo.cfg.xml and <bamboo-clone-home>/xml-data/configuration/administration.xml and change the server names/ip addresses according to the new location.
7. point bamboo.home of your cloned instance to the unzipped <bamboo-clone-home> directory
8. start the bamboo clone

This should give you a perfectly cloned instance.

Your next steps

• If the new server has different locations for
  • JDKs
  • Ant
  • Maven
  • Perforce
  • Msbuild tools
    adjust the settings in the server capabilities settings to match the locations on the new machine.

• From here you can upgrade if desired.
• After the upgrade you should be able to export your instance without problems and then migrate to another database for instance.

Connecting to (embedded) HSQL Database using Dbvisualiser

The purpose of this guide is to walk you through connecting to Bamboo's embedded Hypersonic SQL Database using the Database Administration tool DBVisualizer.

The step by step instructions on how to Configure DBVisualizer and connect it to HSQLDB are below:

Prerequisites

1. Download and install the latest copy of DBVisualizer.
2. You will also need to download a copy (preferably the latest version) of HSQLDB.
3. Extract the contents of the HSQLDB archive.
4. Ensure that Bamboo is not running.

Connection Procedure
Please ensure that you read and follow the instructions below carefully.

⚠️ Remember to backup your `<bamboo-home>/database` folder before attempting any modifications.

1. Enter Connection Name

   ![Database Connections](image)

   1. Click on the icon highlighted in red.
   2. Enter an identifiable name for the connection, e.g. Bamboo_db.

2. Select JDBC Driver

   ![Select Database Driver](image)

   1. Select **HSQLDB embedded** from the drop down list.
   2. Click on **Load Driver Files**.
   3. Browse to directory where the `HSQLDB.jar` file is located.

3. Select Database Path
1. Browse to your `<Bamboo-Home>` directory.
2. Open the `Database` folder.
3. Select the `defaultdb.properties` file.

4. Enter Connection Details

![DbVisualizer Free - New Connection Wizard](image)

1. Remove the ".properties" (e.g. `<Bamboo-Home>/database/defaultdb`) from the end of `defaultdb`.
2. Type in `sa` for the username.
3. Leave the password field blank.

*(Refer to the example screenshot above)*.

5. Connect to the Embedded Database

1. Click 'Test Connection' to verify that the details are correct.
2. Click 'Finish' to complete the setup.
3. Select the connection from the list on the left hand side.
4. You can now click on "Connect" to connect to the embedded database.

*HSQL Database Manager*

Alternatively, you can use HSQLDB's database manager. Just copy the value of `hibernate.connection.url` in `bamboo.cfg.xml` as the URL and you're good to go.

**CVS Error logging in Bamboo**

Currently, if the server throws an error during a CVS build in Bamboo versions 2.0.x, the application will hang with no indication of any checkout/update problems. There is an open JIRA issue tracking this problem.

In order to further debug any CVS issues, you will need to turn up the CVS logging by passing in the `-DcvsClientLog=system` system argument to Bamboo. Please edit `<bamboo-install>/bamboo.sh` accordingly.

**Deactivating a Bamboo user**

To deactivate a Bamboo user account (rather than deleting it), change the password so that the user cannot login. To do this,

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Users' link in the left navigation column.
3. The 'Manage Users' screen will be displayed. Locate the relevant user in the list, and click the corresponding 'Edit' link in the **
4. The ‘User Details’ screen will be displayed. Type a new password in the ‘Password’ and ‘Confirm Password’ fields.

7. Click on the Save button.

Debugging Subversion connection in Bamboo

To debug your SVN connection, it is useful to enable logging. Modify the system property while launching the Java VM with:

```
$java -Djava.util.logging.config.file=path/to/logging.properties.disabled
```

To do this:

1) Download the `logging.properties.disabled` file attached to this document.

The `logging.properties.disabled` file contains a system property `svnkit.level` used to control the log level:

- **FINE** *(default)* — the level for non-detailed logging
- **FINER** — the level for more detailed logging
- **FINEST** — the level for full logging

2) Configure Bamboo to start with the `-Djava.util.logging.config.file=*path/to/*logging.properties.disabled` command, where "path/to/" refers to the absolute path to the `logging.properties.disabled` file from step (1).

The method of doing this depends on which operating system you are using:

- **If you are running Bamboo under Linux:**
  Modify the bamboo .sh script to pass in the logging parameter. To do this, find the following section:

  ```
  RUN_CMD="java -Xms256m -Xmx512m -XX:MaxPermSize=256m -Djava.awt.headless=true -classpath
  $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085
  com.atlassian.bamboo.server.Server 8085 ./webapp /"
  
  Change the RUN_CMD variable to:
  
  ```
  RUN_CMD="java -Xms256m -Xmx512m -XX:MaxPermSize=256m -Djava.awt.headless=true -classpath
  $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true
  -Djava.util.logging.config.file=*path/to/*logging.properties.disabled -Djetty.port=8085
  com.atlassian.bamboo.server.Server 8085 ./webapp /
  ```

- **If you are running Bamboo under Windows:**
  Modify the `wrapper.conf` file to add the logging parameter. To do this, find the following section:

  ```
  wrapper.java.classpath.1=../lib/*.jar
  wrapper.java.classpath.2=../lib
  wrapper.java.classpath.3=../webapp/WEB-INF/classes
  wrapper.java.classpath.4=../webapp/WEB-INF/lib/*.jar
  wrapper.java.library.path.1=../lib
  wrapper.java.additional.1=-Dorg.mortbay.xml.XmlParser.NotValidating=true
  wrapper.java.additional.2=-XX:MaxPermSize=256m
  wrapper.java.additional.3=-Djava.awt.headless=true
  ```

  Add the following line to the end of the section:

  ```
  wrapper.java.additional.4=-Djava.util.logging.config.file=*path/to/*logging.properties.disabled
  ```

3) Save your changes, and restart Bamboo for your changes to take effect.

4) You will find the `SVNKit log` file in `USER_HOME/svnkit.0.log` (where `USER_HOME` is the home directory of the user running Bamboo or logged in e.g: `/home/user/svnkit.0.log`).
Do I have to upgrade all remote agents for Bamboo Release 2.1.2?

We have improved the availability and reliability of remote agents in this release, by adding a failover to reconnect agents when the network drops out.

By default, remote agents now use ActiveMQ failover mechanism to reconnect.

It's not essential to upgrade the agent jar. The agent will automatically download the changed code from the server. The agent has a special classloader that actually downloads classes from the server. The JAR file on the agent only contains a handful of classes it needs to bootstrap itself.

Enable User Management debug logging in Bamboo

This page describes how to turn on user management debug logging.

Edit <bamboo-install>/webapp/WEB-INF/classes/log4j.properties and append the following lines:

```
log4j.logger.bucket.user=DEBUG
log4j.category.com.atlassian.user=DEBUG
log4j.category.com.atlassian.bamboo.user.BambooUserManagerImpl=DEBUG
```

Restart bamboo.

Fixing failing Bamboo builds, with OutOfMemory errors

If your Maven/Ant builds are failing with OutOfMemory errors, this means there isn't enough memory assigned to your build. To fix this error, you need to increase the memory allocated to your build(s). To do this, we need to edit the plans build configuration and increase the memory allocated for your build.

**If you are using Maven, to build your project.**

Edit your plan's build configuration, and add the following variable to the 'System Environment Variables' field in your plan's build configuration page:

```
MAVEN_OPTS=-Xmx512m
```

**If you are using Ant, to build your project.**

Edit your plan's build configuration, and add the following variable to the 'System Environment Variables' field in your plan's build configuration page:

```
ANT_OPTS=-Xmx512m
```

Note: You may wish to change the memory allocation from 512mb to fit your plan's requirements.

Fixing OutOfMemory Errors in Bamboo

**I am getting ‘Out of Memory’ errors. How can I allocate more memory to Bamboo?**

Since the default memory setting usually is 256MB in Bamboo, you might have to adjust the settings to run a bigger Bamboo instance with sufficient memory.

**On Linux:**

- In the unpacked Bamboo Standalone directory, edit the file `bamboo.sh`
- Edit the line beginning with `RUN_CMD=` substituting new values for `~Xms` (starting memory) and `~Xmx` (maximum memory)
- Leave the rest of the options in that line unchanged.

Here is an example of a minimal setting for a large system (maximum heap size is set to 768 megabytes):
On Windows:

Bamboo uses a wrapper to start, either as a service or in a console.

- The wrapper reads the configuration from the wrapper.conf file, which is found in BAMBOO_INSTALL/conf. (The parameters are documented inside the file.)
- Edit the -Xms value to increase your Bamboo memory.

Other Notes

Allocating too much memory to your JVM Heap can also cause OutOfMemory Errors.

The error java.lang.OutOfMemoryError: unable to create new native thread occurs when the operating system is unable to create new threads. This is due to the JVM Heap taking up the available RAM. Big heaps take away from the space that can be allocated for the stack of a new thread.

For Linux the maximum heap size of the JVM cannot be greater than 2GB. If you only have 2GB RAM in your server, it is not recommended to set the Max size of the JVM that high. The size of the stack per thread can also contribute to this problem. The stack size can reduce the number of threads that can be created.

To fix this problem, you should reduce the size of your JVM Heap and also the size of the stack per thread. The stack size can be changed with the following (example) parameter:

```
-Xss512k
```

Please refer to the following guide as a reference for JVM tuning: http://goobsoft.homeip.net/Wiki.jsp?page=JavaDebianTuning.

Allocating large maximum heap and low minimum heap can slow down Bamboo

If Xmx is much (much) larger than Xms Java will spend long periods of time in garbage collection to reach the Xms goal. This will slow down Bamboo performance.

If you need to increase Xmx also increase Xms as well to avoid that scenario.

Getting a memory dump on OOM errors

Passing in -XX:+HeapDumpOnOutOfMemoryError will make the JVM create a memory dump, when it runs out of memory. To do this:

On Linux:

- In the unpacked Bamboo Standalone directory, edit the file bamboo.sh
- Edit the line beginning with RUN_CMD= and add -XX:+HeapDumpOnOutOfMemoryError to the line
- Leave the rest of the options in that line unchanged.

For example

```
RUN_CMD="java -server -XX:+HeapDumpOnOutOfMemoryError -Xms512m -Xmx768m -XX:MaxPermSize=256m -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /
"
```

On Windows:

Bamboo uses a wrapper to start, either as a service or in a console.

- The wrapper reads the configuration from the wrapper.conf file, which is found in BAMBOO_INSTALL/conf. (The parameters are documented inside the file.)
- Add an additional -XX:+HeapDumpOnOutOfMemoryError parameter to the wrapper and save changes.

⚠ You need to restart the Bamboo server for changes to take affect. If you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server.

Permanent Generation Size
If you get the error message: java.lang.OutOfMemoryError: PermGen space this means that you have exceeded Java's fixed 64Mb block for loading class files. You will need to add the argument -XX:MaxPermSize and increase the memory.

- JDK 1.4 does not provide information as to why the OutOfMemory error occurred.
- JDK 1.5 and above are recommended as they provide a description of the error as in the above example.

Hibernate errors in logs after upgrading to Bamboo 2.0

If you are upgrading to Bamboo 2.0 from Bamboo 1.2.4 by pointing to your Bamboo-Home, you may see the following errors in your logs.

In Bamboo 2.0 we introduced compatibility with Oracle and MS SQL Server, this meant we the up had to make a few changes to the Bamboo Database schema and as a side-affect of these changes - you might notice hibernate warnings above in your atlassian-bamboo logs while Bamboo starts up. These warnings will not prevent your instance of Bamboo from working correctly, but will display every time Bamboo is started.

If you do wish to remove these warning messages, follow the steps below:

1. Perform an export from your current Bamboo 2.0 instance.
2. Re-install Bamboo 2.0 on a fresh instance.
3. Import the old exported data into your new Bamboo 2.0 instance.

This will remove the Hibernate Error messages while Bamboo starts up.

How do I construct a cron expression in Bamboo?

Cron is a time-based job scheduler used in Unix/Linux computer operating systems with a unique and powerful terminology. A number of scheduling features in Bamboo, such as build expiry and elastic instance scheduling, require you to specify your requirements as a cron-based expression. For example, a cron expression such as "0 0/30 9-19 ? * MON-FRI" signifies that a scheduled event will be triggered every half an hour from 9am to 7pm, Monday to Friday.

A cron expression comprises of 6 mandatory and one optional field to specify a schedule. The fields in sequential order are: seconds, minutes, hours, day-of-month, month, day-of-week and (optional) year, i.e.

```
<seconds> <minutes> <hours> <day-of-month> <month> <day-of-week> <year (optional)>
```

Each field can be expressed as an integer (e.g. 1, 2, 3, etc) and special characters can be used in most fields as well (i.e. ', - * / ? L W #').

Bamboo uses OpenSymphony's Quartz to schedule cron tasks. The syntax it accepts may vary from other cron implementations. Please refer to the Quartz documentation for further information on each of these parameters and more detailed examples.

How do I disable SSH access to my elastic instances?

By default, SSH (Secure Shell) access is enabled for elastic instances, the first time that you use Elastic Bamboo. Access rules for the Amazon Elastic Compute Cloud (EC2) are managed by 'security groups' in the Amazon Web Services Console. You can disable SSH access for your elastic instances by changing the EC2 access rules to remove the 'SSH' Connection Method from the 'elasticbamboo' security group.

For instructions on changing the EC2 access rules for Elastic Bamboo, please read the Elastic Bamboo Security document.

How do I shut down my elastic instances if I have restarted my Bamboo server?
If you restart your Bamboo server without shutting down your elastic instances first, your elastic instances will continue to run. Your elastic instances will also be orphaned from your Bamboo server, and you will not be able to shut them down via Bamboo after your Bamboo server has restarted. You will need to terminate them via the Amazon Web Services (AWS) Console.

To shut down an elastic instance via the AWS Console,

1. Log in to the AWS Console. The ‘Amazon EC2’ tab of the console should display.
2. Click the 'Instances' link under the 'Images & Instances' section of the left navigation column. Your EC2 instances should display.
3. Check the checkbox next to the instances that need to be terminated in the 'My Instances' panel. In most cases, it should be all instances unless you are running Elastic Bamboo on multiple Bamboo servers.
4. The buttons at the top of the 'My Instances' panel should become enabled. Click the 'Terminate' button to terminate your instances.

How do I stop Bamboo from shutting itself down and restarting?

If your Bamboo server is shutting itself down and restarting multiple times during the day, you may be experiencing problems with Bamboo's service wrapper.

Symptom

Bamboo server is restarting itself a couple of times per day. Wrapper reported in the log JVM seems to be hung and will be terminated.

e.g.

```
INFO | wrapper | 2009/01/28 15:24:34 | Wrapper Process has not received any CPU time for 11 seconds. Extending timeouts.
... 
[TransportConnector] Connector vm://bamboo Stopped
[BrokerService] ActiveMQ JMS Message Broker (bamboo, ID:BSYTEST5-4449-1233174172549-1:0) stopped
STATUS | wrapper | 2009/01/28 15:28:18 | <-- Wrapper Stopped
```

Explanation

Bamboo's service wrapper comes with a `timeout` that specifies the interval at which the JVM is pinged. A response is expected in that time. If the JVM is too busy, it will not respond to this ping in time.

`wrapper.ping.timeout` defines the timeout in seconds. 0 means that it will never time out. The default value of this setting is 30 seconds.

Solution

Increase the timeout in the wrapper's configuration. To do this,

1. Edit `<Bamboo-Install>/conf/wrapper.conf`.
2. Add the following line to the end of the wrapper.conf file:

```
wrapper.ping.timeout=90
```

3. Restart Bamboo. If you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server.

How do I stop the Bamboo server from automatically configuring my remote agent’s capabilities?

The Bamboo server automatically detects and populates the capabilities that a remote agent should be configured with upon agent start up. If you have modified the agent capabilities, they will be reset by the server’s automatic capability detection when the agent is next restarted.

You can override this by adding the following flag, "--DISABLE_AGENT_AUTO_CAPABILITY_DETECTION=true", to the Bamboo server.

JUnit parsing in Bamboo

Bamboo can parse any test output that conforms to standard JUnit XML format. The implementation of this is pretty simple — Bamboo looks for specific tags in the JUnit XML output.

A failed JUnit XML report, that is successfully parsed by Bamboo:

```
<testsuite errors="0" tests="3" time="0.391" failures="1"
 name="com.atlassian.bamboo.repository.perforce.PerforceSyncCommandTest">
 <properties>
  <property value="Java(TM) 2 Runtime Environment, Standard Edition" name="java.runtime.name"/>
  <property value="UnicodeBig" name="sun.io.unicode.encoding"/>
  ............
 </properties>
 <testcase time="0.001" name="testGeneratesCorrectP4CommandLine"/>
 <testcase time="0" name="testGettersReturnExpectedStuff"/>
 <testcase time="0.164" name="testUsingPerforceWhenNoFilesHaveChanged">
 <failure type="junit.framework.AssertionFailedError"
 message="Should not have any errors. [Perforce client error; Connect to server failed; ]
 expected:&lt;0&gt; but was:&lt;4&gt;"
 at junit.framework.Assert.fail(Assert.java:47)
 at junit.framework.Assert.failNotEquals(Assert.java:282)
 at junit.framework.Assert.assertEquals(Assert.java:64)
 at junit.framework.Assert.assertEquals(Assert.java:201)
 at com.atlassian.bamboo.repository.perforce.PerforceSyncCommandTest.testUsingPerforceWhenNoFilesHaveChanged
 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 junit.framework.TestCase.runTest(TestCase.java:154)
 at junit.framework.TestCase.runBare(TestCase.java:127)
 at junit.framework.TestResult.runProtected(TestResult.java:110)
 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 sun.reflect.GeneratedMethodAccessor17.invoke(Unknown Source)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
 at java.lang.reflect.Method.invoke(Method.java:585)
 at org.apache.maven.surefire.batteryJUnit.BatteryJUnit.executeJUnit(JUnitBattery.java:242)
 at org.apache.maven.surefire.batteryJUnit.BatteryJUnit.executeJUnit(JUnitBattery.java:216)
 at org.apache.maven.surefire.executableBattery(ExecutableBattery.java:215)
 at org.apache.maven.surefire.junitRunner.junitRunner.run(Surefire.java:163)
 at org.apache.maven.surefire.junitRunner.run(Surefire.java:87)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```
A passed JUnit XML report, that is successfully parsed by Bamboo.
Known issues with CVS in Bamboo 2.0

In Bamboo 2.0, we rewrote the CVS implementation and replaced the CVS log command with the CVS rlog command - this lets you perform a CVS update on your local working directory without checking out your project.
CVS Error logging in Bamboo

Currently, if the server throws an error during a CVS build in Bamboo versions 2.0.x, the application will hang with no indication of any checkout/update problems. There is an open JIRA issue tracking this problem.

In order to further debug any CVS issues, you will need to turn up the CVS logging by passing in the `-DcvsClientLog=system` system argument to Bamboo.

1) Incompatibility with CVS servers 1.11.1 and below

Support for the `rlog` command 1.11.1p and performing a CVS `rlog` command returns the following error:

```
-cvs [rlog aborted]: server does not support rlog
```

2) Incompatibility with CVS server version 1.11.x when using "." to denote the root module to be checked out.

The CVS `rlog` command fails if you are using CVS version 1.11.x, with the following error.

```
INFO   | jvm 1 | 2008/05/15 14:19:10 | E cvs: recurse.c:642: do_recursion: Assertion `strstr(repository, "/./") == ((void *)0)` failed.
INFO   | jvm 1 | 2008/05/15 14:19:10 | error
```

Please upgrade your CVS version to 1.12.x to get around this issue.

3) CVS Checkout format

Due to prior issues, Bamboo will checkout all files (including text files) from the CVS server as binary, however post Bamoo 2.1.2 this behaviour can be changed via a system parameter. To do this restart Bamboo with the following parameter (if you have any elastic agents running, ensure that they are shut down before you restart the Bamboo server. If you do not shut down your elastic instances before restarting, they will continue to run and become orphaned from your Bamboo server).

```
-DCVS_CHECKOUT_BINARY_FORMAT=false
```

Post 2.1.5 this has been replaced with a more flexible option

```
-DCVS_CHECKOUT_FORMAT=BINARY
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Command Options</th>
<th>Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY (Default)</td>
<td>-b</td>
<td>forces all files to be checked out in binary and won't convert any line endings</td>
</tr>
<tr>
<td>TEXT</td>
<td>-kv</td>
<td>forces all files to be checked out as text and converts all line endings (even Binary files)</td>
</tr>
<tr>
<td>NONE</td>
<td></td>
<td>lets CVS decide whether or not to convert line endings</td>
</tr>
</tbody>
</table>

For further reference, on configuring Bamboo start-up options see this document

Monitoring and Profiling Bamboo

This page helps you to set up profiling for bamboo. Profiling information can be useful to monitor bamboo's performance, memory consumption and the server's CPU load

On this page

- Profiling with JMX
  - Enabling the profiler
  - Monitoring and Controlling the Profiler with JConsole
- Profiling with Yourkit
  - Installing Yourkit
  - Enabling the profiler
Profiling with JMX

Enabling the profiler

JMX is Sun's native java platform monitor. JConsole can be used to visualise the profiling data.

To enable JMX add `-Dcom.sun.management.jmxremote (or -Dcom.sun.management.jmxremoteport=<portNum> for remote monitoring) as a command line argument when starting bamboo.

For Unix: add the parameter to the RUN_CMD line in bamboo.sh
For Windows: add the parameter to the .conf\wrapper.conf file as wrapper.java.additional.4=-D....

Restart Bamboo.

Does not work when bamboo is run as a windows service
At the moment profiling via JMX only works when bamboo is run as a console application on windows. Running bamboo as a windows service with profiling is currently not supported.

Monitoring and Controlling the Profiler with JConsole

Please refer to Sun's documentation

Profiling with Yourkit

Installing Yourkit

Yourkit is an alternative (commercial) java profiler.
Follow the installation instructions for your platform.
You do not need a license if you just run the profiling agent with bamboo.

Enabling the profiler

Add `-agentlib:yjpagent=onexit=snapshot` to the command line in bamboo.sh or as an additional parameter to wrapper.conf

Then follow the instructions on enabling the profiler manually for your platform.

Restart Bamboo.

Monitor Memory usage and Garbage Collection in Bamboo

A simple way to do this is to turn on garbage collection and heap dump on out of memory.

Please add the following parameters to the run command line in bamboo.sh

```
```

Then restart bamboo as ./bamboo.sh start.

The garbage collection traces and the heap dumps are in <bamboo-install>/gc.log.

Some helpful links:
http://java.sun.com/developer/technicalArticles/Programming/GCPortal/

Moving Bamboo-Home of an agent

To move an agent's Bamboo-Home -

1. Move the Bamboo-Home of the agent, to the intended location.
2. Edit the `<Bamboo-Agent-Home>/bamboo-agent.cfg.xml` file, find the following line -

```xml
```

3. Point the working directory and the artifact directory to the new Bamboo-Home.
4. Start your Agent with `-Dbamboo.home=your_new_agent_home` and point to your new Bamboo-Agent-Home.

**Performing a thread dump.**

If Bamboo stops responding, or is performing poorly, you should create a thread dump to help Atlassian determine the cause of the problem. This will show the state of each thread in the JVM, including a stack trace and information about what locks that thread is holding and waiting for.

**Windows Users**

To take a thread dump from Windows:

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 Bamboo process. Users running Bamboo Standalone, select the ‘Java (Jetty) …’ option. Users running Bamboo WAR should select their application server process.
6. Ensure that the “Thread dump” and “Keep Remote Thread Running” is selected.
7. Click OK to capture the thread dump.
8. Save the output to a file, eg ‘threaddump.log’
9. If you were asked by Atlassian technical support to create the thread dump, attach the logfile to the support ticket.

Alternatively, if you are not running Bamboo as a service, click on the console and press `<CTRL>+<BREAK>`

**Linux (and Solaris and other Unixes) Users**

Find the process id of the JVM and issue the command:

Use the `ps` command to get list of all processes.

```
kill -3 <pid>
```

**Note:** This will not kill your server (so long as you included the "-3" option, no space in between). The thread dump will be printed to Bamboo's standard output.

Please note that some application servers (like tomcat) redirect stdout (to catalina.out for instance).

**Thread Dump Tools**

- Samurai
- Thread Dump Analyzer TDA

**Problems running Bamboo under Sun JDK 1.4**

**Backing up Bamboo**

With Bamboo versions 1.2.1 and 1.2.2, exporting Bamboo when running under Sun JDK version 1.4 will fail due to incompatible with JAVA versions.
Exporting Bamboo

This issue was fixed in Bamboo 1.2.3.

Bamboo version 1.2 will not start under Sun JDK 1.4

It is recommended that users upgrade to Bamboo 1.2.1 or above as a fix for this issue.

Removing Coverage plug-in data from the Bamboo database

The third-party Coverage plug-in for Bamboo stores very large amounts of data in the Bamboo database. There are two consequences of this:

1. Using the Coverage plug-in with an embedded Bamboo database may result in poor performance, OutOfMemoryErrors, and/or Bamboo start-up failures; and
2. After installing the Coverage plug-in, you may encounter difficulties with Bamboo's import, export and backup features, such as OutOfMemoryErrors and corrupted export and backup files.

Precautionary Measures

To mitigate the risk of these problems, Atlassian makes the following recommendations to users of the Coverage plug-in:

1. Atlassian strongly recommends that you migrate to a supported external database before installing this plug-in; and
2. Once you have installed the Coverage plug-in, Atlassian strongly recommends that you regularly backup your bamboo-home and external database using external tools, as the plug-in may interfere with the reliability of Bamboo's built-in backup feature. We intend to address the underlying issue in a future release of Bamboo.

Recovery Procedure

If an instance of Bamboo is configured with an embedded database and the Coverage plug-in is failing for the reasons described above, this can be rectified by removing the Coverage plug-in's data from the database, using the following procedure.

On Linux, Mac OS X and other Unix-like platforms:

1. Shut down Bamboo.
2. Execute the following commands in a shell, substituting bamboo-home with the path to your Bamboo home directory:

   cd bamboo-home/database
   grep "^INSERT INTO BUILDRESULTSUMMARY_CUSTOMDATA .*'coverage\." defaultdb.script | gzip > coverage.sql.gz
   mv defaultdb.script defaultdb.script.backup_with_coverage
   gzip defaultdb.script.backup_with_coverage
   gunzip -c defaultdb.script.backup_with_coverage.gz | grep -v "^INSERT INTO BUILDRESULTSUMMARY_CUSTOMDATA .*'coverage\." > defaultdb.script

On Microsoft Windows:

For assistance, please raise a Bamboo support request.

Restoring passwords to recover admin users

Use this document if you are unable to login as administrator or have forgotten your password and do not have Mail Server configured, to manually replace administrator passwords.

Follow the instructions for either the Embedded Database or External Database. If you have not configured a database, use the Embedded instructions.
Embedded Database Instructions

Stage One - Identify Administrator

This guide assumes that the first user added was an administrator. If this is not the case, search for the admin username and find their user id number, then modify their password hash instead.

1. Shutdown Bamboo
2. In your Bamboo home directory, open \database\defaultdb.script file in a text editor
3. Search for the text:

```
INSERT INTO USERS VALUES(1
```

To find the administrator login entry:

```
INSERT INTO USERS VALUES(1,'USERNAME', 'PASSWORD_HASH')
```

Where the 1 is the user id number, and USERNAME and PASSWORD_HASH are actual values.

As an example, my table entry for user admin with password admin looks like this;

```
INSERT INTO USERS VALUES(1,'admin',
'x61Ey612K12gpFL56FT9weDnpSo4AV8j8+qx2AuThDrY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==',
'admin@admin.com', '2007-08-14 11:26:18.504000000', 'admin')
```

1. This step makes admin the administrator's password. Bamboo does not store passwords in plain text in the database, but uses hashes computed from the original password. The hash for the characters admin is below:

```
x61Ey612K12gpFL56FT9weDnpSo4AV8j8+qx2AuThDrY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==
```

Paste the admin password hash between the " characters of their existing PASSWORD_HASH. The new administrator login entry should look like:

```
INSERT INTO USERS VALUES(1,'USERNAME',
'x61Ey612K12gpFL56FT9weDnpSo4AV8j8+qx2AuThDrY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==',
'EMAIL', 'DATE_TIME', 'FULL_NAME')
```

Where USERNAME is the administrator username.

2. Save the file
3. Start up Bamboo
4. Login with the administrator username and password admin

External Database Instructions

Stage One - Identify User

The first user added is always an admin. To restore your password you simply need to update the password hash in the USERS table with the admin hash.

Connect to your database using a database admin tool such as DBVisualiser. Please download a database admin tool now if you do not have one installed already. Once installed, connect to your database and retrieve the list of administrator usernames with:

```
select * from USERS where ID=1
```

This command should list all users who belong to Bamboo-Admin user group.

Stage Two - Replace Administrator Password

Bamboo does not store passwords in plain text in the database, but uses hashes computed from the original password. You instead cut and a
paste a hash, rather than the plain password, over the existing password. Below is the hash for the password admin.

\[ x61Ey612Kl2gqFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qOyB+XURPWx10Nxp3Y3pB37A== \]

To change the password to admin for a given username:

1. Shutdown Bamboo
2. Connect to your database. run this SQL on your database: select * from USERS where NAME='admin'If you are using LDAP integration for user management (not only authentication) then your admin user will be in a different table. The SQL to run is:

```
update USERS set PASSWORD = 'x61Ey612Kl2gqFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qOyB+XURPWx10Nxp3Y3pB37A=='
where NAME = 'USER_NAME_FROM_STAGE_ONE'
```

3. Start Bamboo
4. Login with your username and your password is now admin

Send Errors to stderr - Script Builder in Visual Studio WinXP to build Solutions Files

To display an Error Summary for erroneous builds in bamboo build summary is not available for the Script Builder - going through the build logs seems tedious.

There is a section named “Error summary” which collects all errors during the build process that are printed to stderr. For example a build script would print this message into the build summary section. It is up to you to insert the appropriate messages into your build script.

**Problem**

The actual problem is devenv.com/msbuild not being very helpful: both build tools only append to stdout stream, even in the case of warnings/errors during the build.

**Solution**

I solved the issue by writing a simple Ruby script that invokes the build tool and filters the stdout stream for any warnings and errors via regexp; the matching warning/error lines are then echoed to stderr and Bamboo picks them up nicely.

```
#!/bin/bash
echo "ERROR build xyz failed" >&2
```

```
pipe = IO.popen("devenv.com #{$*}[0] /Rebuild ")
errors = 0
warnings = 0
while line = pipe.gets
  if line =~ /^.* : .* error .*$/
    $stderr.puts line
    errors += 1
  elsif line =~ /^.* : warning .*$/
    $stderr.puts line
    warnings += 1
  else
    $stdout.puts line
  end
end
exit errors > 0 ? 1 : 0
```

**Related Pages**

Knowledge Base - (BSP-1381) Script Builder Display build errors in Error Summary

Testing LDAP or Active Directory connectivity with Paddle

**Introduction**
Paddle is a tool that will test the LDAP or Active Directory settings in your atlassian-user.xml.

**Instructions for use**

You do not need to have Bamboo running to run this tool. The steps are:

1. Download into a directory where you have permissions to create files.
2. Copy your atlassian-user.xml into that directory - this is found in your <Bamboo-Install>/webapp/WEB-INF/classes/ directory.
3. Run `java -jar paddle-x.x.jar` (where x.x is the version of Paddle you downloaded).

**Parameters**

Paddle currently supports the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td><code>java -jar paddle-x.x.jar debug</code></td>
<td>Prints DEBUG messages to the console as well as paddle.log.</td>
</tr>
<tr>
<td>limit</td>
<td><code>java -jar paddle-x.x.jar</code> limit=100</td>
<td>Sets the limit on the number of results returned by user and group queries. Defaults to 10.</td>
</tr>
</tbody>
</table>

**Sample output**

This is an example of a successful run:

```
# LDAP Support Tool version 1.1
# Connection to LDAP/Active Directory Server at ldap://192.168.0.86:389 SUCCESSFUL.

TEST 1: Search and list 10 users

User: CN=Administrator
    Member of:
    (1) CN=Schema Admins
    (2) CN=Enterprise Admins
    (3) CN=Domain Admins
    (4) CN=Group Policy Creator Owners

User: CN=Guest
    Does not belong to any LDAP groups.

User: CN=SUPPORT_388945a0
    Member of:
    (1) CN=HelpServicesGroup

User: CN=IUSR_MALTSHOVEL
    Does not belong to any LDAP groups.

User: CN=IWAM_MALTSHOVEL
    Member of:
    (1) CN=IIS_WPG

User: CN=ASPNET
    Does not belong to any LDAP groups.

User: CN=krbtgt
    Does not belong to any LDAP groups.

User: CN=John\, Smith
    Member of:
    (1) CN=Domain Users
    (2) CN=Sales and Marketing

User: CN=Matt Ryall
    Member of:
    (1) CN=Enterprise Admins
    (2) CN=Domain Admins
```
Troubleshooting Elastic Bamboo

This page contains troubleshooting tips for configuring and using Elastic Bamboo.

EBS Volumes and Snapshots

Elastic Agent does not mount a Volume to my Elastic Instance

What happens

When an Elastic Bamboo is configured to use an EBS snapshot the elastic agent will mount an EBS-volume that has been created before to the elastic instance. This volume will be populated with the contents of a snapshot.

The relation is

snapshot 1 ---- * Volume
Your AWS account lets you limit the maximum number (N) of volumes available. If not enough volumes are available, i.e. are ready to mount, the elastic agent will not succeed to mount a volume to the elastic instance and will use the default bamboo agent home directory that will be empty after startup.

The instance will still work but you will experience longer build times...

Looking at your aws console will reveal some volumes that are linked to a snapshot but are not attached to an instance for example i-b626b6df:/dev/sdh (attached). You will have 'dangling' volumes associated with snapshots that are still 'in use' and cannot be used by other elastic instances. There are several reasons

- Bamboo has crashed or lost connection to the elastic instance
- Bamboo was shut down without all running instances shutting down.

**Diagnosis**

Access your elastic instance via ssh and view the /tmp/setupEbsSnapshot.log file for any details.

**Solution**

Free up your accounts capacity by deleting some volumes that are not mounted to any active instance. Remember to to shut down elastic instances via bamboo's Management Console.

**Troubleshooting SQL Exceptions - Detailed Hibernate Logging**

This document outlines the steps to take to increasing logging on your system.

1. Open <bamboo-install>/webapp/WEB-INF/classes/log4j.properties and add/change the following lines. The double ## lines are comments, leave them intact.

   ```
   ## log hibernate prepared statements/SQL queries (equivalent to setting 'hibernate.show_sql' to 'true')
   log4j.logger.net.sf.hibernate.SQL=DEBUG, confluencelog
   log4j.additivity.net.sf.hibernate.SQL=false
   
   ## log hibernate prepared statement parameter values
   log4j.logger.net.sf.hibernate.type=DEBUG, confluencelog
   log4j.additivity.net.sf.hibernate.type=false
   ```

   If you can not locate these lines in your log4j.properties file, please add them to the end of it.

2. Restart bamboo.
3. Redo the steps that led to the error.
4. Zip up your logs directory and attach it your support ticket.

**Troubleshooting Subversion connection**

**Note:**

- Bamboo 1.0.4 does not authenticate with Subversion. Please upgrade Bamboo to 1.0.5 or above (we recommend upgrading to the latest version of Bamboo).
- SSL authentication for Subversion is not available as a Bamboo feature prior to Bamboo 2.1.

**Cannot validate Subversion repository during plan configuration**

The most common repository error is:

```
This is not a valid Subversion Repository: svn: Authentication required for '<http://svnhost:80; My Subversion Repository>'
```

In all likelihood, the authentication failed due to invalid login credentials. Please double-check your username and password and try again.

**Using self signed SSL certificates**
If you are using a self signed certificate prior to Bamboo 2.1, you need to get subversion to cache your repository connection as there is no way to instruct Bamboo to accept self-signed SSL certificates.

On Linux

- Open up a command line (as the user running Bamboo) and checkout your repository before using Bamboo.

On Windows

- If you are running Bamboo as a service on Windows, then consult this document to run the service as the local user,
- Login as the user running Bamboo, and checkout your repository.

Troubleshooting SVN Connection

If you are still having trouble connecting to repository,

1. Try logging in outside of Bamboo, from the command line or from any other repository client. If this step fails, then it’s likely that your repository is not configured correctly.
2. Please the consult our JIRA issue tracker for known Bamboo repository issues.
3. If you still can’t connect to Subversion, please raise a support request and attach your atlassian-bamboo server logs to the support request.

Using Bamboo For Release Management

Suggested reading for users wishing to configure Bamboo for release management:

- Customer plan for release management with Atlassian Bamboo & JIRA
- Plugins for release management and deployment
- Forum thread on release management

Using Bamboo with Clover

Getting Started

To use Clover with Bamboo, you need to:

1. Integrate Clover with Bamboo with your build:
   - Clover-for-Ant Installation Guide
   - Clover-for-Maven 2 Installation Guide
2. And either:
   - call the Clover goal in your plan configuration (see Specifying a Plan's Builder);
   or:
   - add the maven-clover-plugin report to the reports section in your POM.
3. Ensure that there are tests present in your build plan that generate test results in JUnit test report format.
4. Ensure that your build creates a Clover report (that is, a clover.xml file). Bamboo will use this Clover report as source.
5. Set up Bamboo to read the Clover report (clover.xml file) generated by Clover. To do this:
   a. Ensure the 'Clover output will be produced' check-box is ticked in your plan's build configuration page.
   b. Instruct Bamboo on the location of your 'Clover XML Directory'—where Bamboo will look for the XML report output file from Clover. Please specify file path relative to your plan's root directory (e.g /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/), i.e. please do not specify an absolute path.

For further details, please see Specifying a Plan's Builder.

Common Problems

Q: I have managed to get Clover statistics displayed in numerical form for each build, but the graphs do not show a history of these statistics?
A: The history of Clover is displayed over time periods (e.g. a day, a week, a month), and the minimum data point is per day. The Clover coverage will not display data that is less than a day old.

Q: Will the Bamboo/Clover integration run on failed builds?
A: Before Bamboo version 1.2.1, Bamboo would only report Clover coverage for successful builds. As of Bamboo 1.2.1, Bamboo will report Clover coverage regardless of the build outcome.

Why can’t I SSH to my elastic instance?

SSH access is enabled by default for elastic instances when you first use Elastic Bamboo. If you are attempting to access your elastic instance via SSH and get bad permission errors (see below for an example error), there may be a permission issue with the Elastic Bamboo private key file (elasticbamboo.pk).
On certain Unix/Linux systems, the SSH command may fail if the private key file's permissions are too open. Changing the private key file's permissions to make it only readable by the owner may resolve this. You may change file permissions to

```
$ sudo chmod 0600 /home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk
```

Example error: Elastic Bamboo private key file permission

```
[user@atlassian ~]$ ssh -i /home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk root@ec2-174-129-192-22.compute-1.amazonaws.com
The authenticity of host 'ec2-174-129-192-22.compute-1.amazonaws.com (174.129.192.22)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-174-129-192-22.compute-1.amazonaws.com,174.129.192.22' (RSA) to the list of known hosts.

Permissions 0644 for '/home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk' are too open.
It is recommended that your private key files are NOT accessible by others. This private key will be ignored.

bad permissions: ignore key: /home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk
Enter passphrase for key '/home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk':

Permissions 0644 for '/home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk' are too open.
It is recommended that your private key files are NOT accessible by others. This private key will be ignored.

bad permissions: ignore key: /home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk
Enter passphrase for key '/home/bamboo/bamboo-home/xml-data/configuration/elasticbamboo.pk':
```

Working with Sun JAVA libraries

Due to licensing restrictions, we are not allowed to re-distribute native SUN libraries through our maven2 public repositories.

If you are developing plugins for Bamboo or building Bamboo from source, you might need `javax.mail` and `javax.transaction:jta:jar` for Bamboo to build successfully. The relevant POMs for this look something like this:

```
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.3.2</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>jta</groupId>
    <artifactId>jta</artifactId>
    <version>1.0.1</version>
    <scope>compile</scope>
</dependency>
```

Before building, please install the Sun JAR's into your local Maven2 repositories by following the instructions below.

To install the `javax.mail` JAR into your local Maven2 repository:

1. Download the `javax.mail` Jar from Sun's website.
2. Install it on your local machine by entering the following command in a terminal:

```
mvn install:install-file -DgroupId=javax.mail -DartifactId=mail -Dversion=1.3.3 -Dpackaging=jar -Dfile=YOUR/PATH/TO/FILE
```
To install `javax.transaction:jta:jar` JAR into your local Maven2 repository:

1. Download the `javax.transaction:jta:jar` Jar from Sun's website.
2. Install it on your local machine by entering the following command in a terminal:

   ```
   mvn install:install-file -DgroupId=javax.transaction -DartifactId=jta -Dversion=1.0.1B -Dpackaging=jar -Dfile=/path/to/file
   ```

Tips of the Trade

Below are some links to external blog posts and articles containing technical tips and instructions on setting up and using Bamboo. This page presents an opportunity for customers and community authors to share information and experiences.

The references here are technical 'how to' guides written by bloggers who use Bamboo. For feature tours, solution tours and other information about continuous integration, 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 Bamboo. 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 Bamboo. 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:

- Lightning fast notification
- Automating the staging and production deployments
- Continuous Integration for Ruby
- Continuous Integration Goodness for your Ruby Project
- Atlassian Bamboo and Perl Test Harness
- Bamboo JMeter Aggregator - Getting the most from performance builds
- Bamboo plugins for Git and GitHub

### Build Management

**Lightning fast notification**

- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: Using IM as a notification system to keep developers up to date on the new deployments for their various projects
- Date: 15 April 2009
- Related documentation: About Instant Messenger (IM) Notifications

**Automating the staging and production deployments**

- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: Moving your builds to general availability (GA) and production deployments with Maven, JIRA and Bamboo
- Date: 6 May 2009
- Related documentation: JiraVersions Plugin
### Non-Java Languages

#### Continuous Integration for Ruby
- By: John Ferguson Smart, on the 'Atlassian Blog'
- About: A Continuous Integration environment that runs Ruby builds and tests on Bamboo, and automates the deployment and installation on a remote test machine
- Date: 20 May 2009
- Related documentation: Can Bamboo build and test non-Java projects?

#### Continuous Integration Goodness for your Ruby Project
- By: Nick Sieger, on the 'Nick Sieger' blog
- About: Running Ruby builds and tests on Bamboo
- Date: 6 Jan 2007
- Related documentation: Can Bamboo build and test non-Java projects?

#### Atlassian Bamboo and Perl Test Harness
- By: Gary Richardson, on blog 'The Cult of Gary'
- About: Getting your Perl test cases to work with Bamboo, using TAP::Harness::JUnit
- Date: 7 November 2008
- Related documentation: Can Bamboo build and test non-Java projects?

### Performance Builds

#### Bamboo JMeter Aggregator - Getting the most from performance builds
- By: James Roper, on the 'Atlassian Blog'
- About: Using the Bamboo JMeter Aggregator plugin to manage the data produced by your performance builds
- Date: 21 May 2009
- Related documentation: Bamboo JMeter Aggregator Plugin

### Repositories

#### Bamboo plugins for Git and GitHub
- By: Ken Olofsen, on the 'Atlassian Blog'
- About: Using Bamboo with Git and GitHub
- Date: 2 May 2009
- Related documentation: Specifying a Plan's Source Repository

### Other Sources of Information

- Bamboo documentation
- Evaluator resources
- Atlassian website
- Atlassian forums
- Atlassian blog
- Bamboo plugins

### Glossary
activity log

Every plan has an activity log. An activity log is a temporary display of the latest output from the plan's most recent build log.

agent

An agent is a service that runs Bamboo builds. There are two types of agents:

- local agents run on the Bamboo server.
- remote agents run on computers other than the Bamboo server. An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2)

(Note: Local agents run in the server's process, i.e. in the same JVM as the server. Each remote agent runs in its own process, i.e. has its own JVM.)

Each agent has a defined set of capabilities. An agent can only run builds for plans whose requirements are met by the agent's capabilities.

See Configuring Agents and Capabilities and Specifying a Plan's Capability Requirements.

agent-specific capability

An agent-specific capability applies to one agent only. Note that the value of an agent-specific capability will override the value of a shared capability of the same name (if one exists).

See About Agents and Capabilities and Configuring Capabilities.

artifact

An artifact is something created by a build. There are two types of artifacts:

- User-defined artifacts (e.g. JAR files) are specified in the build's plan by a Bamboo administrator.
- Auto-generated artifacts are created automatically by Bamboo.
**author**

An *author* is any person who contributes to a *build* by checking-in code to a repository that is associated with a Bamboo *plan*. An author need not be a Bamboo user.

See Reporting on Author Trends.

**build**

A *build* is one execution of a *plan*.

Every build has a Build Number, which is appended to the relevant Plan Key to form the Build Key. For example, if a plan with the key "CRM-BRANCH" is executed for the seventeenth time, the build key will be "CRM-BRANCH-17".

**build activity**

*Build activity* is the number of builds that occur in a given period of time.

**build duration**

*Build duration* is the total time taken to execute a *build plan* — that is, the time taken to compile the code and run all of the plan's tests.

Variations in a plan's *build duration* can be monitored over time.

**builder**

A *builder* is a software compiler program external to Bamboo. Bamboo supports multiple builders. Once a builder is defined in the Bamboo system, it can then be specified in *build plans* by a Bamboo administrator.

See Configuring a new Builder.

**build log**

Every *build* has a *build log*. A *build log* is a permanent record of all the output generated by compiling the *plan*’s source-code and executing the tests.

**build plan**

See *plan*.

**build queue**

The Bamboo *build queue* controls the sequence of *builds*. When a plan submits a build to the build queue, the build will wait in the build queue until a suitable *agent* is available to run the build.

The build queue is displayed on the Dashboard.

**build result**

Every completed build has a *build result*:

- 'Successful' — the code compiled, with or without errors, and all tests completed successfully.
- 'Failed' — either the code did not compile, or at least one test failed.

Additionally,

- if the build result is 'Failed', and the previous build result was 'Successful', the build is said to be 'Broken'.
- if the build result is 'Successful', and the previous build result was 'Failed', the build is said to be 'Fixed'.
**build telemetry**

*Build telemetry* is the insight provided by Bamboo’s dynamic reports, charts and collation of build metrics. Build telemetry helps identify trends across build plans and across authors — not just focusing on the results of a single build.

**capability**

A *capability* is a feature of an *agent*. A capability can be a:

- builder (e.g. Maven)
- JDK
- custom capability (a key-value property which defines a particular characteristic of an agent, e.g. ‘operating.system=WindowsXP’ or ‘fast.builds=true’)
- Perforce (location of the P4 client application, if Perforce is being used as the source repository)

Capabilities can be defined specifically for an agent, or they can be shared between either all local agents or all remote agents. Note that the value of an agent-specific capability overrides the value of a shared capability of the same name (if one exists).

See [Configuring Capabilities](#).

**child**

A *child* is a plan which gets triggered when another plan completes a build. See [Setting up Build Dependencies](#).

**committer**

A *committer* is the Bamboo user(s) who committed code to a particular build (i.e. someone who committed code after the previous build was checked out by Bamboo).
Administrators can configure a plan’s notifications to be sent to the build’s committer(s).

**custom capability**

Custom capabilities can be used to control which build plans will be built by a particular agent. For example, if the builds for a particular plan should only run in a Windows environment, you could create a custom capability ‘operating.system=WindowsXP’ for the appropriate agent(s), and specify it as a requirement for this plan.

- To create a new custom capability in your Bamboo system, see [Configuring a new Custom Capability](#).
- To specify a build plan’s requirement for a custom capability, see [Specifying a Plan’s Capability Requirements](#).

**default image**

Atlassian maintains an *elastic image* in the *Amazon Simple Storage Service (S3)*. This publicly shared image is the default elastic image used with the *Elastic Bamboo* feature in Bamboo. The following packages (and related capabilities) are included in the default image:

- Fedora 8 (Linux) operating system
- JDK 6u7 (DLJ distribution)
- Maven 1.1
- Maven 2.0.9
- Ant 1.7.1
- Bamboo 2.2 elastic agent*

* The agent jar also contains the libraries required to connect to Subversion and CVS.

**elastic agent**

An *elastic agent* is a remote agent that runs in the *Amazon Elastic Compute Cloud (EC2)*. An elastic agent process runs in an instance of an Elastic Bamboo *elastic image*. An elastic agent inherits its capabilities from the *elastic image* that it was created from.
elastic bamboo

Elastic Bamboo is a feature in Bamboo that allows you to utilise computing resources from the Amazon Elastic Compute Cloud (EC2) to run builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Builds can be run on these 'elastic agents' in the same way that builds are run non-elastic agents.

elastic block store

Amazon Elastic Block Store (EBS) provides persistent storage volumes that can be attached to EC2 instances. Elastic Bamboo can use the EBS to store snapshots of relatively static build information, such as checkouts of source code and Maven repository data. You can choose a snapshot to create EBS volumes from and associate it with an elastic image configuration. These volumes can then be attached to your elastic instances when they are started from their elastic image.

elastic image

An elastic image is an Amazon Machine Image (AMI) that is stored in the Amazon Simple Storage Service (S3) for use with the Elastic Bamboo feature. An elastic image is used to create elastic instances, which in turn create elastic agents. Conceptually, the elastic image can be considered to be the boot hard disk that contains the operating system and software run on your elastic instances.

You can associate multiple elastic images with a Bamboo server. One default shared image is maintained by Atlassian in the Amazon S3, and is available to all Elastic Bamboo users. You also create your own custom elastic images.

elastic instance

An elastic instance is an instance of an elastic image. An elastic instance is created whenever an image is started. An image can be started multiple times, creating multiple instances. Each time an elastic instance is started, one elastic agent is created.

Conceptually, an elastic instance can be thought of as a computer. Elastic agent processes are run on this computer and the elastic image is the boot hard disk. Please note however, that elastic instances are temporary and transient. Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) are not persisted when the instance is shut down. Any customisations to the instance itself will also be lost when the it is shut down.

favourites

Each Bamboo user can nominate their favourite plans — that is, the plans they work with the most.

Each user's favourites are displayed on the 'My' page of the Dashboard. Bamboo administrators can also configure each plan to send build result notifications to users who have nominated the plan as one of their favourites (these users are known as the plan's 'watchers').

global permission

A global permission is the ability to perform a particular operation in relation to Bamboo as a whole.

See Granting Global Permissions to Users or Groups.

See also plan permission.

label

A label is a convenient way to tag and group build results that are logically related to each other. Labels can also be used to define RSS feeds and to control build expiry.

Labels can be applied to build results automatically, by specifying the label(s) in a build plan (note that only Bamboo administrators can do this). Labels can also be applied ad hoc to build results by Bamboo users.

local agent

See agent.
parent
A parent is a plan which triggers another plan to build whenever it completes a build. See Setting up Build Dependencies.

permission
See plan permission and global permission.

plan
A Bamboo plan (or build plan) is the "recipe" for a build.

A plan defines: what gets built (i.e. the source-code repository); how the build is triggered; which builder to use; which agent capabilities are required for the build; what artifacts the build will produce; what tests to run; who will be notified of the build result; any labels with which the build result or build artifacts will be tagged; and who has permission to view and perform various actions on a plan and its build results.

Every plan belongs to a project.

Each plan has a Plan Key, which is prefixed by the relevant Project Key. E.g. the "CRM" project could have plans "CRM-TRUNK" and "CRM-BRANCH".

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

plan permission
A plan permission is the ability to perform a particular operation in relation to a build plan. For each plan, different permissions can be granted to particular groups and/or users.

See Granting Plan Permissions to Users or Groups.

See also global permission.

project
A project is a collection of plans.

A project enables easy identification of plans that are logically related to each other, which is useful for instance when generating reports across multiple plans.

Each project has a Name (e.g. "CRM System") and a Key (e.g. "CRM"). The Project Key is prefixed to the relevant Plan Keys, e.g. the "CRM" project could have plans "CRM-TRUNK" and "CRM-BRANCH".

queue
See build queue.

reason
A build's reason is the way in which the build was triggered.

There are a number of ways in which a build can be triggered for a plan,

- **Build Strategy:**
  - Code updated — a build can be triggered whenever one or more authors checks-in code.
  - Scheduled build — a build can be scheduled to occur at regular intervals.
  - Manual build — a build can be triggered manually.
  - Initial clean build — a build will be triggered when a new plan is created.
- **Build Dependency:**
**Dependency** — a build can be triggered whenever a successful build occurs for another plan.

The way in which each build was triggered is listed in the 'Reason' column on the Dashboard.

Note that build triggering can only be configured by a Bamboo administrator. For more information please see About Build Triggering.

---

**remote agent**

See agent.

**remote agent supervisor**

A remote agent supervisor is an application that is installed alongside a Bamboo remote agent, by default. The remote agent supervisor is an implementation of the Java Service Wrapper.

The remote agent supervisor monitors remote agents on the machine that it is installed on. If any remote agent crashes, the remote agent supervisor will automatically attempt to restart it. If communications are lost with the Bamboo server, the remote agent will shut itself down and wait for the remote agent supervisor to restart it.

The remote agent supervisor will run on the following operating systems:

- **Linux:**
  - x86
  - x86_64
  - IA64
  - PPC 64 bit (but not 32 bit)
- **Mac OSX:**
  - all architectures
- **Solaris:**
  - x86
  - x86_64 (running in 32 bit mode)
  - IA64 (running in 32 bit mode)
  - SPARC (both 32 bit and 64 bit)
- **Windows:**
  - 32 bit (but not 64 bit)

**requirement**

A requirement is an agent capability required by a build plan.

Together, capabilities and requirements control which agents can execute builds for particular plans. Each plan can only be built by agents whose capabilities meet the plan's requirements. Matching can be specified as either a regular expression or an exact match. See Specifying a Plan's Capability Requirements.

**shared capability**

Shared capabilities are inherited by all applicable agents, that is, (shared) local server capabilities are inherited by all local agents, and shared remote capabilities are inherited by all remote agents. Note, however, that the value of a shared capability will be overridden by the value of an agent-specific capability of the same name (if one exists).

See About Agents and Capabilities and Configuring Capabilities.

**triggering**

There are a number of ways in which a build can be triggered for a plan,

- **Build Strategy:**
  - Code updated — a build can be triggered whenever one or more authors checks-in code.
  - Scheduled build — a build can be scheduled to occur at regular intervals.
  - Manual build — a build can be triggered manually.
  - Initial clean build — a build will be triggered when a new plan is created.
• Build Dependency:
  • Dependency — a build can be triggered whenever a successful build occurs for another plan.

The way in which each build was triggered is listed in the 'Reason' column on the Dashboard.

Note that build triggering can only be configured by a Bamboo administrator. For more information please see About Build Triggering.

watcher

A plan's watchers are the Bamboo users who have marked this plan as one of their favourites. Administrators can configure a plan's notifications to be sent to the plan's watchers.

Diagrams

How are builds distributed to agents?

An agent will consume a single plan at a time and block until that build is complete. If you'd like to build multiple plans concurrently on the Bamboo server then simply setup multiple local agents. If the agents are remote, then you'll need to install that number of agent instances on the machine. Separate installations are required because each remote agent will need its own home and log directories.
Plan ACME

Build no. ACME-123

Plan ACME submits build number 123 to the Build Queue

Build Queue

The Build Queue forwards build no. ACME-123 to the next available agent whose capabilities meet Plan ACME's requirements.

BAMBOO SERVER
Local Agent 1
Local Agent 2

Network

Remote Agent 1
Remote Agent 2
Remote Agent 3

How do capabilities affect the distribution of builds to agents?
Plan ACME

**Capability requirements**
- Builder = Maven
- JDK = 1.5
- Custom: fast.build = true

**Build no. ACME-123**
Plan ACME submits build number 123 to the Build Queue

**Build Queue**
The Build Queue forwards build no. ACME-123 to the next available agent whose capabilities meet Plan ACME's requirements. (Note: agent-specific capabilities override shared capabilities.)

**BAMBOO SERVER**
- Local Agent 1
- Local Agent 2

**Remote Agent 1**

**Remote Agent 2**
- Remote Agent 3

**Agent-specific capabilities**
- JDK = 1.5
- Custom: fast.build = true

**Shared local capabilities**
- Builder = Ant
- JDK = 1.5

**Shared remote capabilities**
- Builder = Maven
- JDK = 1.4

TreeNavigation
TreeNavigationVersions

Click for all versions

Bamboo 2.4.x
Bamboo 2.3.x
Bamboo 2.2.x
Bamboo 2.1.x
Bamboo 2.0.x
Bamboo 1.2.x
Bamboo 1.1.x
Bamboo 1.0.x