# Bamboo Documentation

## 2 Bamboo User's Guide

### 2.2 Getting Started

- **2.2.1 Using the Bamboo Dashboard** ................................................................. 15
- **2.2.2 Viewing Bamboo's Current Activity** ....................................................... 16
- **2.2.3 Viewing your Latest Build Results** ...................................................... 17
- **2.2.4 Working with Favours** ............................................................................. 20
- **2.2.4.1 Adding a Plan to your Favourties** ....................................................... 21
- **2.2.4.2 Removing a Plan from your Favourties** ............................................. 22
- **2.2.5 Displaying the Wallboard** ................................................................. 23
- **2.2.6 Viewing Bamboo's Agents** .............................................................. 24

### 2.2.2 Working with Projects, Stages, Plans and Jobs

- **2.2.2.1 Working with Plans** ............................................................................. 34
  - **2.2.2.1.1 Viewing a Plan's Details** ............................................................... 34
  - **2.2.2.1.2 Viewing the JIRA Issues linked to the Builds in a Build Plan** .......... 35
- **2.2.2.2 Working with Stages** ............................................................................. 36
- **2.2.2.3 Working with Jobs** ............................................................................. 37
  - **2.2.2.3.1 Viewing a Job** ............................................................................... 37
  - **2.2.2.3.2 Viewing a Job's Maven Dependencies** ......................................... 38
  - **2.2.2.3.3 Viewing the Clover Coverage Summary for a Job** .................... 39

### 2.2.3 Working with Build Results

- **2.2.3.1 Viewing a Build Result** ................................................................. 40
  - **2.2.3.2 Viewing the Code Changes that Triggered a Build** ..................... 41
  - **2.2.3.3 Viewing a Build's Artifacts** .......................................................... 42
  - **2.2.3.4 Viewing a Build Log** ....................................................................... 43
  - **2.2.3.5 Viewing the Metadata for a Build Result** .................................... 44
- **2.2.3.6 Viewing the Clover Code-Coverage for a Build Result** ............... 45
  - **2.2.3.7 Viewing the JIRA Issues for a Build Result** .................................... 46
  - **2.2.3.8 Linking JIRA Issues to a Build** .................................................... 47

### 2.2.4 Working with Tests

- **2.2.4.1 Viewing Test Results for a Build** ..................................................... 48
- **2.2.4.2 Viewing a Test's History** .............................................................. 49
  - **2.2.4.3 Viewing Test Statistics for a Job** .................................................. 50

### 2.2.5 Reporting on Plan Trends

- **2.2.5.1 Viewing Build Statistics for a Plan** .................................................. 51
- **2.2.5.2 Generating Reports across Multiple Plans** ..................................... 52
  - **2.2.5.2.1 'Build Activity' Report** ............................................................... 53

## Index

1. _books_ .......................... 2
2. Bamboo Documentation .......................... 8
  2.1 Bamboo 101 .......................... 8
  2.2 Bamboo User's Guide .......................... 15

## Contents

**Bamboo Documentation**

- **Getting Started** .......................... 15
  - **Using the Bamboo Dashboard** ................................................................. 15
  - **Viewing Bamboo's Current Activity** ....................................................... 16
  - **Viewing your Latest Build Results** ...................................................... 17
  - **Working with Favours** ............................................................................. 20
    - **Adding a Plan to your Favourite** ....................................................... 21
    - **Removing a Plan from your Favourite** ............................................. 22
  - **Displaying the Wallboard** ................................................................. 23
  - **Viewing Bamboo's Agents** .............................................................. 24

**Working with Projects, Stages, Plans and Jobs**

- **Working with Plans** ............................................................................. 34
  - **Viewing a Plan's Details** ............................................................... 34
  - **Viewing the JIRA Issues linked to the Builds in a Build Plan** .......... 35
- **Working with Stages** ............................................................................. 36
- **Working with Jobs** ............................................................................. 37
  - **Viewing a Job** ............................................................................... 37
  - **Viewing a Job's Maven Dependencies** ......................................... 38
  - **Viewing the Clover Coverage Summary for a Job** .................... 39

**Working with Build Results**

- **Viewing a Build Result** ................................................................. 40
  - **Viewing the Code Changes that Triggered a Build** ..................... 41
  - **Viewing a Build's Artifacts** .......................................................... 42
  - **Viewing a Build Log** ....................................................................... 43
  - **Viewing the Metadata for a Build Result** .................................... 44
- **Viewing the Clover Code-Coverage for a Build Result** ............... 45
  - **Viewing the JIRA Issues for a Build Result** .................................... 46
  - **Linking JIRA Issues to a Build** .................................................... 47

**Working with Tests**

- **Viewing Test Results for a Build** ..................................................... 48
- **Viewing a Test's History** .............................................................. 49
  - **Viewing Test Statistics for a Job** .................................................. 50

**Reporting on Plan Trends**

- **Viewing Build Statistics for a Plan** .................................................. 51
- **Generating Reports across Multiple Plans** ..................................... 52
  - **'Build Activity' Report** ............................................................... 53

---

**Bamboo Documentation**

- **Getting Started** .......................... 15
  - **Using the Bamboo Dashboard** ................................................................. 15
  - **Viewing Bamboo's Current Activity** ....................................................... 16
  - **Viewing your Latest Build Results** ...................................................... 17
  - **Working with Favours** ............................................................................. 20
    - **Adding a Plan to your Favourite** ....................................................... 21
    - **Removing a Plan from your Favourite** ............................................. 22
  - **Displaying the Wallboard** ................................................................. 23
  - **Viewing Bamboo's Agents** .............................................................. 24

**Working with Projects, Stages, Plans and Jobs**

- **Working with Plans** ............................................................................. 34
  - **Viewing a Plan's Details** ............................................................... 34
  - **Viewing the JIRA Issues linked to the Builds in a Build Plan** .......... 35
- **Working with Stages** ............................................................................. 36
- **Working with Jobs** ............................................................................. 37
  - **Viewing a Job** ............................................................................... 37
  - **Viewing a Job's Maven Dependencies** ......................................... 38
  - **Viewing the Clover Coverage Summary for a Job** .................... 39

**Working with Build Results**

- **Viewing a Build Result** ................................................................. 40
  - **Viewing the Code Changes that Triggered a Build** ..................... 41
  - **Viewing a Build's Artifacts** .......................................................... 42
  - **Viewing a Build Log** ....................................................................... 43
  - **Viewing the Metadata for a Build Result** .................................... 44
- **Viewing the Clover Code-Coverage for a Build Result** ............... 45
  - **Viewing the JIRA Issues for a Build Result** .................................... 46
  - **Linking JIRA Issues to a Build** .................................................... 47

**Working with Tests**

- **Viewing Test Results for a Build** ..................................................... 48
- **Viewing a Test's History** .............................................................. 49
  - **Viewing Test Statistics for a Job** .................................................. 50

**Reporting on Plan Trends**

- **Viewing Build Statistics for a Plan** .................................................. 51
- **Generating Reports across Multiple Plans** ..................................... 52
  - **'Build Activity' Report** ............................................................... 53

---

**Bamboo Documentation**

- **Getting Started** .......................... 15
  - **Using the Bamboo Dashboard** ................................................................. 15
  - **Viewing Bamboo's Current Activity** ....................................................... 16
  - **Viewing your Latest Build Results** ...................................................... 17
  - **Working with Favours** ............................................................................. 20
    - **Adding a Plan to your Favourite** ....................................................... 21
    - **Removing a Plan from your Favourite** ............................................. 22
  - **Displaying the Wallboard** ................................................................. 23
  - **Viewing Bamboo's Agents** .............................................................. 24

**Working with Projects, Stages, Plans and Jobs**

- **Working with Plans** ............................................................................. 34
  - **Viewing a Plan's Details** ............................................................... 34
  - **Viewing the JIRA Issues linked to the Builds in a Build Plan** .......... 35
- **Working with Stages** ............................................................................. 36
- **Working with Jobs** ............................................................................. 37
  - **Viewing a Job** ............................................................................... 37
  - **Viewing a Job's Maven Dependencies** ......................................... 38
  - **Viewing the Clover Coverage Summary for a Job** .................... 39

**Working with Build Results**

- **Viewing a Build Result** ................................................................. 40
  - **Viewing the Code Changes that Triggered a Build** ..................... 41
  - **Viewing a Build's Artifacts** .......................................................... 42
  - **Viewing a Build Log** ....................................................................... 43
  - **Viewing the Metadata for a Build Result** .................................... 44
- **Viewing the Clover Code-Coverage for a Build Result** ............... 45
  - **Viewing the JIRA Issues for a Build Result** .................................... 46
  - **Linking JIRA Issues to a Build** .................................................... 47

**Working with Tests**

- **Viewing Test Results for a Build** ..................................................... 48
- **Viewing a Test's History** .............................................................. 49
  - **Viewing Test Statistics for a Job** .................................................. 50

**Reporting on Plan Trends**

- **Viewing Build Statistics for a Plan** .................................................. 51
- **Generating Reports across Multiple Plans** ..................................... 52
  - **'Build Activity' Report** ............................................................... 53
2.2.6 Reporting on Author Trends ........................................... 64
2.2.6.1 Viewing Build Statistics for all Authors ................. 65
2.2.6.2 Viewing Build Results for an Author ....................... 65
2.2.6.3 Generating Reports on Selected Authors ................. 66
2.2.6.3.1 'Build Activity per Author' Report ..................... 67
2.2.6.3.2 'Number of Build Failures per Author' Report .... 68
2.2.6.3.3 'Number of Builds Broken per Author' Report ..... 68
2.2.6.3.4 'Number of Builds Fixed per Author' Report .... 69
2.2.6.3.5 'Percentage of Successful Builds per Author' Report 69
2.2.7 Working with Comments ............................................. 70
2.2.7.1 Commenting about a Build Result ......................... 70
2.2.7.2 Viewing Comments about a Build Result ................. 71
2.2.7.3 Viewing Code Check-in Comments ............................ 72
2.2.8 Working with Labels .................................................. 73
2.2.8.1 Labelling a Build Result ................................. 73
2.2.8.2 Removing a Label from a Build Result .................... 74
2.2.8.3 Viewing Labelled Build Results ......................... 74
2.2.8.4 Viewing Popular Labels .................................... 75
2.2.9 Subscribing to RSS Feeds ........................................... 75
2.2.9.1 Subscribing to an RSS Feed for All Build Results for All Plans 76
2.2.9.2 Subscribing to an RSS Feed for All Build Results for a Particular Plan . 76
2.2.9.3 Subscribing to an RSS Feed for Failed Builds for All Plans ................. 76
2.2.9.4 Subscribing to an RSS Feed for Failed Builds for a Particular Plan ................. 76
2.2.9.5 Subscribing to an RSS Feed for Labelled Build Results .... 76
2.2.10 Working with Instant Messenger (IM) Notifications ....... 77
2.2.10.1 Labelling a Build Result via IM ......................... 77
2.2.10.2 Commenting about a Build Result via IM ............... 78
2.2.11 Managing your User Profile ..................................... 79
2.2.11.1 Changing your Password ............................... 79
2.2.11.2 Changing your Notification Preferences ................. 80
2.2.11.3 Associating your Author Name with your User Profile .... 80
2.2.11.4 Changing your Personal Details ......................... 81
2.2.11.5 Viewing your Notifications .............................. 81
2.3 Bamboo Administrator's Guide ...................................... 82
2.3.1 Configuring Projects, Plans, Stages and Jobs ................. 82
2.3.1.1 Configuring a Plan ......................................... 83
2.3.1.1.1 Creating a Plan ........................................ 84
2.3.1.1.2 Editing a Plan ......................................... 92
2.3.1.1.3 Specifying the Source Repository for a Plan .... 100
2.3.1.1.4 Disabling or Deleting a Plan .......................... 140
2.3.1.1.5 Configuring Concurrent Builds ....................... 142
2.3.1.1.6 Modifying Multiple (Bulk) Plans ................. 143
2.3.1.1.7 Moving Plans to a Different Project ............... 145
2.3.1.1.8 Setting up Build Dependencies .................... 148
2.3.1.2 Configuring a Stage ......................................... 150
2.3.1.2.1 Creating a Stage ..................................... 151
2.3.1.2.2 Deleting a Stage ..................................... 152
2.3.1.2.3 Editing a Stage ....................................... 153
2.3.1.3 Configuring a Job ........................................... 154
2.3.1.3.1 Creating a Job ....................................... 154
2.3.1.3.2 Editing a Job ........................................ 158
2.3.1.3.3 Disabling or Deleting a Job ......................... 168
2.3.1.3.4 Configuring Tasks .................................. 170
2.3.1.3.5 Specifying the Source Repository for a Job .... 190
2.3.2 Configuring Agents and Capabilities .......................... 191
2.3.2.1 Configuring Agents ......................................... 194
2.3.2.1.1 Creating a Local Agent ................................ 195
2.3.2.1.2 Creating a Remote Agent ............................ 196
2.3.2.1.3 Editing an Agent's Details ......................... 200
2.3.2.1.4 Disabling or Deleting an Agent .................... 201
2.3.2.1.5 Viewing an Agent .................................. 202
2.3.2.1.6 Monitoring Agent Status ............................. 210
2.3.2.2 Configuring Capabilities .................................... 213
2.3.2.2.1 Configuring a new Executable ....................... 213
2.3.2.2.2 Configuring a new JDK ................................ 220
2.3.2.2.3 Configuring a new Custom Capability ............ 224
2.3.2.2.4 Configuring a new Perforce Capability ............ 225
2.3.2.2.5 Configuring a new Mercurial Capability ........... 226
2.3.2.2.6 Configuring a Capability ............................ 227
2.3.2.2.7 Renaming a Capability ................................ 228
2.3.2.2.8 Deleting a Capability ............................... 229
2.3.2.2.9 Viewing a Capability's Agents and Jobs ............ 230
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>353</td>
<td>2.3.10.4 Configuring Plugins</td>
</tr>
<tr>
<td>353</td>
<td>2.3.10.4.1 Enabling the Auto-Favourite Plugin</td>
</tr>
<tr>
<td>354</td>
<td>2.3.10.4.2 Enabling the Clover Plugin</td>
</tr>
<tr>
<td>355</td>
<td>2.3.10.5 Installing a Plugin</td>
</tr>
<tr>
<td>356</td>
<td>2.3.10.6 Disabling or Enabling a Plugin</td>
</tr>
<tr>
<td>357</td>
<td>2.3.10.7 Viewing the Plugin Audit Log</td>
</tr>
<tr>
<td>358</td>
<td>2.3.10.8 Viewing your Installed Plugins</td>
</tr>
<tr>
<td>359</td>
<td>2.3.10.9 Plugin Blacklist</td>
</tr>
<tr>
<td>359</td>
<td>2.3.11 Using Bamboo with Other Applications</td>
</tr>
<tr>
<td>359</td>
<td>2.3.11.1 Embedding Bamboo into Other Applications</td>
</tr>
<tr>
<td>359</td>
<td>2.3.11.1.1 Javascript Widgets</td>
</tr>
<tr>
<td>363</td>
<td>2.3.11.2 Integrating Bamboo with Other Applications</td>
</tr>
<tr>
<td>363</td>
<td>2.3.11.2.1 Integrating Bamboo with JIRA</td>
</tr>
<tr>
<td>365</td>
<td>2.4 Bamboo Installation and Upgrade Guide</td>
</tr>
<tr>
<td>366</td>
<td>2.4.1 Bamboo Installation Guide</td>
</tr>
<tr>
<td>367</td>
<td>2.4.1.1 Bamboo Remote Agent Installation Guide</td>
</tr>
<tr>
<td>370</td>
<td>2.4.1.1.1 Configuring Remote Agent Capabilities via bamboo-capabilities.properties</td>
</tr>
<tr>
<td>371</td>
<td>2.4.1.1.2 Legacy Remote Agent Installation Guide</td>
</tr>
<tr>
<td>373</td>
<td>2.4.1.2 Bamboo EAR-WAR Installation Guide</td>
</tr>
<tr>
<td>374</td>
<td>2.4.1.2.1 Configuring Tomcat to Use HtpOnly Session ID Cookies</td>
</tr>
<tr>
<td>375</td>
<td>2.4.1.3 Bamboo Standalone Installation Guide (Linux)</td>
</tr>
<tr>
<td>376</td>
<td>2.4.1.4 Bamboo Standalone Installation Guide (Mac)</td>
</tr>
<tr>
<td>377</td>
<td>2.4.1.5 Bamboo Standalone Installation Guide (Windows)</td>
</tr>
<tr>
<td>379</td>
<td>2.4.1.6 Running the Setup Wizard</td>
</tr>
<tr>
<td>382</td>
<td>2.4.1.7 Using Bamboo's embedded hSQL database</td>
</tr>
<tr>
<td>382</td>
<td>2.4.1.7.1 PostgreSQL 8.2</td>
</tr>
<tr>
<td>384</td>
<td>2.4.1.7.2 MySQL 5.x</td>
</tr>
<tr>
<td>387</td>
<td>2.4.1.7.3 Oracle 10g and 11g</td>
</tr>
<tr>
<td>388</td>
<td>2.4.1.7.4 Microsoft SQL Server 2005 and 2008</td>
</tr>
<tr>
<td>392</td>
<td>2.4.1.7.5 How do I connect Bamboo to an unsupported database?</td>
</tr>
<tr>
<td>394</td>
<td>2.4.1.8 Hardware sizing considerations</td>
</tr>
<tr>
<td>394</td>
<td>2.4.2 Bamboo Release Notes</td>
</tr>
<tr>
<td>396</td>
<td>2.4.2.1 Bamboo Release Summary</td>
</tr>
<tr>
<td>398</td>
<td>2.4.2.2 Bamboo 3.1 Release Notes</td>
</tr>
<tr>
<td>405</td>
<td>2.4.2.2.1 Bamboo 3.1 Upgrade Guide</td>
</tr>
<tr>
<td>406</td>
<td>2.4.2.3 Bamboo 3.0 Release Notes</td>
</tr>
<tr>
<td>412</td>
<td>2.4.2.3.1 Bamboo 3.0.1 Release Notes</td>
</tr>
<tr>
<td>412</td>
<td>2.4.2.3.2 Bamboo 3.0.2 Release Notes</td>
</tr>
<tr>
<td>413</td>
<td>2.4.2.3.3 Bamboo 3.0.3 Release Notes</td>
</tr>
<tr>
<td>414</td>
<td>2.4.2.3.4 Bamboo 3.0 Upgrade Guide</td>
</tr>
<tr>
<td>416</td>
<td>2.4.2.4 Bamboo 2.7 Release Notes</td>
</tr>
<tr>
<td>426</td>
<td>2.4.2.4.1 Bamboo 2.7 Upgrade Guide</td>
</tr>
<tr>
<td>428</td>
<td>2.4.2.4.2 Bamboo 2.7.2 Release Notes</td>
</tr>
<tr>
<td>429</td>
<td>2.4.2.4.3 Bamboo 2.7.1 Release Notes</td>
</tr>
<tr>
<td>430</td>
<td>2.4.2.4.4 Bamboo 2.7.3 Release Notes</td>
</tr>
<tr>
<td>431</td>
<td>2.4.2.4.5 Bamboo 2.7.4 Release Notes</td>
</tr>
<tr>
<td>432</td>
<td>2.4.2.5 Bamboo 2.6 Release Notes</td>
</tr>
<tr>
<td>444</td>
<td>2.4.2.5.1 Bamboo 2.6 Upgrade Guide</td>
</tr>
<tr>
<td>446</td>
<td>2.4.2.5.2 Bamboo 2.6.3 Release Notes</td>
</tr>
<tr>
<td>447</td>
<td>2.4.2.5.3 Bamboo 2.6.2 Release Notes</td>
</tr>
<tr>
<td>448</td>
<td>2.4.2.5.4 Bamboo 2.6.1 Release Notes</td>
</tr>
<tr>
<td>449</td>
<td>2.4.2.6 Bamboo 2.5 Release Notes</td>
</tr>
<tr>
<td>455</td>
<td>2.4.2.6.1 Bamboo 2.5 Upgrade Guide</td>
</tr>
<tr>
<td>456</td>
<td>2.4.2.6.2 Bamboo 2.5.5 Release Notes</td>
</tr>
<tr>
<td>457</td>
<td>2.4.2.6.3 Bamboo 2.5.2 Release Notes</td>
</tr>
<tr>
<td>458</td>
<td>2.4.2.6.4 Bamboo 2.5.1 Release Notes</td>
</tr>
<tr>
<td>459</td>
<td>2.4.2.6.5 Bamboo 2.5.3 Release Notes</td>
</tr>
<tr>
<td>461</td>
<td>2.4.2.7 Bamboo 2.4 Release Notes</td>
</tr>
<tr>
<td>465</td>
<td>2.4.2.7.1 Bamboo 2.4 Upgrade Guide</td>
</tr>
<tr>
<td>466</td>
<td>2.4.2.7.2 Bamboo 2.4.3 Release Notes</td>
</tr>
<tr>
<td>466</td>
<td>2.4.2.7.3 Bamboo 2.4.2 Release Notes</td>
</tr>
<tr>
<td>467</td>
<td>2.4.2.7.4 Bamboo 2.4.1 Release Notes</td>
</tr>
<tr>
<td>468</td>
<td>2.4.2.8 Bamboo 2.3 Release Notes</td>
</tr>
<tr>
<td>476</td>
<td>2.4.2.8.1 Bamboo 2.3 Upgrade Guide</td>
</tr>
<tr>
<td>477</td>
<td>2.4.2.8.2 Bamboo 2.3.1 Release Notes</td>
</tr>
<tr>
<td>477</td>
<td>2.4.2.9 Bamboo 2.2 Release Notes</td>
</tr>
<tr>
<td>486</td>
<td>2.4.2.9.1 Bamboo 2.2.2 Upgrade Guide</td>
</tr>
<tr>
<td>486</td>
<td>2.4.2.9.2 Bamboo 2.2.4 Release Notes</td>
</tr>
<tr>
<td>487</td>
<td>2.4.2.9.3 Bamboo 2.2.3 Release Notes</td>
</tr>
<tr>
<td>489</td>
<td>2.4.2.9.4 Bamboo 2.2.2 Release Notes</td>
</tr>
<tr>
<td>490</td>
<td>2.4.2.9.5 Bamboo 2.2.1 Release Notes</td>
</tr>
<tr>
<td>491</td>
<td>2.4.2.10 Bamboo 2.1 Release Notes</td>
</tr>
<tr>
<td>496</td>
<td>2.4.2.10.1 Bamboo 2.1 Upgrade Guide</td>
</tr>
<tr>
<td>497</td>
<td>2.4.2.10.2 Bamboo 2.1.5 Release Notes</td>
</tr>
<tr>
<td>498</td>
<td>2.4.2.10.3 Bamboo 2.1.4 Release Notes</td>
</tr>
<tr>
<td>500</td>
<td>2.4.2.10.4 Bamboo 2.1.3 Release Notes</td>
</tr>
<tr>
<td>501</td>
<td>2.4.2.10.5 Bamboo 2.1.2 Release Notes</td>
</tr>
<tr>
<td>503</td>
<td>2.4.2.10.6 Bamboo 2.1.1 Release Notes</td>
</tr>
<tr>
<td>504</td>
<td>2.4.2.11 Bamboo 2.0 Release Notes</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.4.2.11.1 Bamboo 2.0 Upgrade Guide</td>
<td>507</td>
</tr>
<tr>
<td>2.4.2.11.2 Bamboo 2.0.6 Release Notes</td>
<td>510</td>
</tr>
<tr>
<td>2.4.2.11.3 Bamboo 2.0.5 Release Notes</td>
<td>512</td>
</tr>
<tr>
<td>2.4.2.11.4 Bamboo 2.0.4 Release Notes</td>
<td>512</td>
</tr>
<tr>
<td>2.4.2.11.5 Bamboo 2.0.3 Release Notes</td>
<td>514</td>
</tr>
<tr>
<td>2.4.2.11.6 Bamboo 2.0.2 Release Notes</td>
<td>515</td>
</tr>
<tr>
<td>2.4.2.11.7 Bamboo 2.0.1 Release Notes</td>
<td>516</td>
</tr>
<tr>
<td>2.4.2.12 Bamboo 2.0 Beta Release Notes</td>
<td>517</td>
</tr>
<tr>
<td>2.4.2.12.1 Bamboo 2.0 Beta Upgrade Guide</td>
<td>521</td>
</tr>
<tr>
<td>2.4.2.12.2 Bamboo 2.0 Beta 9 Release Notes</td>
<td>524</td>
</tr>
<tr>
<td>2.4.2.12.3 Bamboo 2.0 Beta 8 Release Notes</td>
<td>525</td>
</tr>
<tr>
<td>2.4.2.12.4 Bamboo 2.0 Beta 6 Release Notes</td>
<td>527</td>
</tr>
<tr>
<td>2.4.2.12.5 Bamboo 2.0 Beta 5 Release Notes</td>
<td>529</td>
</tr>
<tr>
<td>2.4.2.12.6 Bamboo 2.0 Beta 4 Release Notes</td>
<td>530</td>
</tr>
<tr>
<td>2.4.2.12.7 Bamboo 2.0 Beta 3 Release Notes</td>
<td>532</td>
</tr>
<tr>
<td>2.4.2.12.8 Bamboo 2.0 Beta 2 Release Notes</td>
<td>534</td>
</tr>
<tr>
<td>2.4.2.12.9 Bamboo 2.0 Beta 1 Release Notes</td>
<td>536</td>
</tr>
<tr>
<td>2.4.2.13 Bamboo 1.2 Release Notes</td>
<td>546</td>
</tr>
<tr>
<td>2.4.2.13.1 Bamboo 1.2 Upgrade Guide</td>
<td>549</td>
</tr>
<tr>
<td>2.4.2.13.2 Bamboo 1.2 Plugin Interface Changes</td>
<td>549</td>
</tr>
<tr>
<td>2.4.2.13.3 Bamboo 1.2.4 Release Notes</td>
<td>549</td>
</tr>
<tr>
<td>2.4.2.13.4 Bamboo 1.2.3 Release Notes</td>
<td>551</td>
</tr>
<tr>
<td>2.4.2.13.5 Bamboo 1.2.2 Release Notes</td>
<td>553</td>
</tr>
<tr>
<td>2.4.2.13.6 Bamboo 1.2.1 Release Notes</td>
<td>554</td>
</tr>
<tr>
<td>2.4.2.14 Bamboo 1.1 Release Notes</td>
<td>555</td>
</tr>
<tr>
<td>2.4.2.14.1 Bamboo 1.1 Upgrade Guide</td>
<td>560</td>
</tr>
<tr>
<td>2.4.2.14.2 Bamboo 1.1.2 Release Notes</td>
<td>560</td>
</tr>
<tr>
<td>2.4.2.14.3 Bamboo 1.1.1 Release Notes</td>
<td>561</td>
</tr>
<tr>
<td>2.4.2.15 Bamboo 1.0 Release Notes</td>
<td>562</td>
</tr>
<tr>
<td>2.4.2.15.1 Bamboo 1.0 Upgrade Guide</td>
<td>564</td>
</tr>
<tr>
<td>2.4.2.15.2 Bamboo 1.0.5 Release Notes</td>
<td>564</td>
</tr>
<tr>
<td>2.4.2.15.3 Bamboo 1.0.4 Release Notes</td>
<td>565</td>
</tr>
<tr>
<td>2.4.2.15.4 Bamboo 1.0.3 Release Notes</td>
<td>566</td>
</tr>
<tr>
<td>2.4.2.15.5 Bamboo 1.0.2 Release Notes</td>
<td>567</td>
</tr>
<tr>
<td>2.4.2.15.6 Bamboo 1.0.1 Release Notes</td>
<td>568</td>
</tr>
<tr>
<td>2.4.2.16 Bamboo 1.0-Beta Release Notes</td>
<td>570</td>
</tr>
<tr>
<td>2.4.2.16.1 Bamboo 1.0-Beta Upgrade Guide</td>
<td>571</td>
</tr>
<tr>
<td>2.4.3 Bamboo Upgrade Guides</td>
<td>571</td>
</tr>
<tr>
<td>2.4.3.1 Bamboo Generic Upgrade Guide</td>
<td>573</td>
</tr>
<tr>
<td>2.4.4 Bamboo Security Advisories</td>
<td>574</td>
</tr>
<tr>
<td>2.4.4.1 Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta)</td>
<td>575</td>
</tr>
<tr>
<td>2.4.4.1.1 Security your Remote Agents</td>
<td>576</td>
</tr>
<tr>
<td>2.4.4.2 Bamboo Security Advisory 2009-03-09</td>
<td>578</td>
</tr>
<tr>
<td>2.4.4.3 Bamboo Security Advisory 2010-05-04</td>
<td>580</td>
</tr>
<tr>
<td>2.4.4.4 Bamboo Security Advisory 2011-03-29</td>
<td>582</td>
</tr>
<tr>
<td>2.4.5 Supported Platforms</td>
<td>583</td>
</tr>
<tr>
<td>2.4.5.1 End of Support Announcements for Bamboo</td>
<td>584</td>
</tr>
<tr>
<td>2.5 Bamboo FAQ</td>
<td>585</td>
</tr>
<tr>
<td>2.5.1 Installation FAQ</td>
<td>587</td>
</tr>
<tr>
<td>2.5.1.1 Changing the Root Context Path</td>
<td>587</td>
</tr>
<tr>
<td>2.5.1.2 Configuring Bamboo on start-up</td>
<td>587</td>
</tr>
<tr>
<td>2.5.1.3 Setting Bamboo standalone to use the jetty.xml file</td>
<td>588</td>
</tr>
<tr>
<td>2.5.1.4 Installation notes for Bamboo on JBoss 4.x</td>
<td>590</td>
</tr>
<tr>
<td>2.5.1.5 Running Bamboo as a Service on Windows</td>
<td>591</td>
</tr>
<tr>
<td>2.5.1.6 Running Bamboo over HTTPS</td>
<td>592</td>
</tr>
<tr>
<td>2.5.1.7 Running Bamboo service on Windows as the local user</td>
<td>593</td>
</tr>
<tr>
<td>2.5.1.8 Setting up JNDI mail on JBoss 4.2.2</td>
<td>594</td>
</tr>
<tr>
<td>2.5.1.9 Setting up JNDI on jetty</td>
<td>595</td>
</tr>
<tr>
<td>2.5.2 Support Policies</td>
<td>596</td>
</tr>
<tr>
<td>2.5.2.1 Bamboo Support Policy</td>
<td>597</td>
</tr>
<tr>
<td>2.5.2.2 Bug Fixing Policy</td>
<td>597</td>
</tr>
<tr>
<td>2.5.2.3 Deploying Multiple Atlassian Applications in a Single Tomcat Container</td>
<td>598</td>
</tr>
<tr>
<td>2.5.2.4 How to Report a Security Issue</td>
<td>598</td>
</tr>
<tr>
<td>2.5.2.5 New Features Policy</td>
<td>599</td>
</tr>
<tr>
<td>2.5.2.6 Patch Policy</td>
<td>599</td>
</tr>
<tr>
<td>2.5.2.7 Security Advisory Publishing Policy</td>
<td>600</td>
</tr>
<tr>
<td>2.5.2.8 Security Patch Policy</td>
<td>600</td>
</tr>
<tr>
<td>2.5.2.9 Severity Levels for Security Issues</td>
<td>601</td>
</tr>
<tr>
<td>2.5.3 Usage FAQ</td>
<td>602</td>
</tr>
<tr>
<td>2.5.3.1 Backing up Bamboo instances over 4GB</td>
<td>602</td>
</tr>
<tr>
<td>2.5.3.2 Bamboo Database Schema</td>
<td>603</td>
</tr>
<tr>
<td>2.5.3.3 Binding Bamboo to one IP address</td>
<td>603</td>
</tr>
<tr>
<td>2.5.3.4 Can Bamboo build and test non-Java projects?</td>
<td>604</td>
</tr>
<tr>
<td>2.5.3.5 Can multiple plans share a common 3rd-party directory?</td>
<td>604</td>
</tr>
<tr>
<td>2.5.3.6 Changing Bamboo database settings</td>
<td>605</td>
</tr>
<tr>
<td>2.5.3.7 Changing the JIRA-Bamboo integration cache settings in Bamboo</td>
<td>605</td>
</tr>
<tr>
<td>2.5.3.8 Changing the remote agent heartbeat interval</td>
<td>605</td>
</tr>
<tr>
<td>2.5.3.9 Cloning a bamboo instance</td>
<td>606</td>
</tr>
<tr>
<td>2.5.3.10 Connecting to (embedded) HSQL Database using Visualliser</td>
<td>607</td>
</tr>
<tr>
<td>2.5.3.11 CVS Error logging in Bamboo</td>
<td>609</td>
</tr>
</tbody>
</table>
2.8.1.1.4 System Plugin Modules .................................................. 688
2.8.1.1.5 Bamboo Event Listeners ........................................... 699
2.8.1.2 Bamboo’s Build Process .................................................. 701
2.8.1.3 Accessing Bamboo Components From Plugin Modules .................. 703
2.8.1.4 Common Bamboo Classes .............................................. 704
2.8.1.5 Bamboo Persistence using Bandana ...................................... 706
2.8.1.6 downloadable Plugin Resources ........................................ 707
2.8.1.7 Web Resources .............................................................. 708
2.8.1.8 Differences between Plugins1 and Plugins2 .................................. 709
2.8.1.9 Plugin Internationalisation .............................................. 710
2.8.1.10 Using Version 1 Plugins .................................................. 710
2.8.2 Bamboo REST APIs .............................................................. 712
  2.8.2.1 Bamboo REST Resources .............................................. 712
  2.8.2.2 Using the Bamboo REST APIs ........................................ 730
  2.8.2.3 Comparing the Bamboo Remote API to the Bamboo REST APIs .......... 732
2.8.3 Bamboo Remote API ............................................................ 733
  2.8.3.1 Build Results Filters ...................................................... 734
  2.8.3.2 Other Services .............................................................. 738
  2.8.3.3 Authentication Services ............................................... 739
  2.8.3.4 Elastic Bamboo Services .............................................. 740
  2.8.3.5 Build Results Services ................................................ 742
  2.8.3.6 Build Services .............................................................. 744
2.8.4 Bamboo Developer FAQ ......................................................... 747
  2.8.4.1 How do I inject managers into my plugin? ............................ 747
  2.8.4.2 How do I search for previous build result? .......................... 747
  2.8.4.3 How do I start a build programatically? ............................. 747
  2.8.4.4 How do I trigger off a build from my action? ....................... 748
  2.8.4.5 Note to Atlassian staff on the Bamboo Developer FAQ ................. 749
2.8.5 Bamboo Plugin Tutorial .................................................... 749
  2.8.5.1 Tutorial 1 - Getting Started with a Simple Post Build Labeller ........ 751
  2.8.5.2 Tutorial 2 - Configurable Regex Labeller ............................ 755
2.8.6 Changes by Version ............................................................. 760
  2.8.6.1 Changes for Bamboo 3.1 ................................................. 760
  2.8.6.2 Changes for Bamboo 3.0 ................................................ 762
  2.8.6.3 Changes for Bamboo 2.7 ................................................ 767
  2.8.6.4 Changes for Bamboo 2.6 ................................................ 775
  2.8.6.5 Changes for Bamboo 2.3 ................................................ 776
  2.8.6.6 Changes for Bamboo 2.2 ................................................ 777
  2.8.6.7 Changes for Bamboo 2.1 ................................................ 778
  2.8.6.7.1 Changes for Bamboo 2.1.5 ......................................... 778
  2.8.6.8 Changes for Bamboo 2.0 ................................................ 779
  2.8.6.8.1 Build Process for 2.0 ............................................... 779
  2.8.6.8.2 Repository plugin changes in 2.0 .................................. 780
  2.8.6.8.3 Updates to the build processing plugins ............................ 780
  2.8.6.8.4 Changes to Bamboo’s Configuration UI require the ConfigurablePlugin .................................................. 781
  2.8.6.8.5 Post-Build Processing in Bamboo 2.0 ................................ 781
  2.8.6.9 Changes for Bamboo 2.4 ................................................ 781
  2.8.6.10 Changes for Bamboo 2.5 ................................................ 782
2.8.7 Building a Bamboo War Distribution From Source ....................... 782
2.8.8 Setting up Bamboo Development Environment in IDEA ................. 782
  2.8.8.1 Building Bamboo In IDEA - Troubleshooting ........................... 785
2.9 Contributing to the Bamboo Documentation .................................. 787
  2.9.1 Tips of the Trade ............................................................. 788
2.9.2 Bamboo Documentation in Other Languages .................................. 790
3. TreeNavigation .............................................................. 791
4. TreeNavigationVersions ...................................................... 791
Bamboo 3.1 Documentation

The 15 most recent bookmarks in Bamboo 3.1
There are no bookmarks to display.

Bamboo Documentation

Bamboo 3.1.x

User's Guide

The Bamboo User's Guide is for project managers, developers, testers – anyone who uses Bamboo. New to Bamboo? Start by exploring the Bamboo dashboard and learning about projects and plans and build results. Try viewing a Plan's details and a Job's details, viewing the build result, the code changes that triggered the build and the build artifacts.

Administrator's Guide

The Bamboo Administrator's Guide is for people with Bamboo administration rights. It will help you set up users and groups, projects and plans, agents and capabilities and security. You may want to configure email and IM notifications and extend Bamboo's functionality by adding plugins. Admin tasks such as backup are also covered. You may also find the Knowledge Base, FAQ and Bamboo Forum useful.

Installation Guide

The Bamboo Installation Guide is for people who are installing Bamboo for the first time. Check the requirements in the Installation Guide, then download and install Bamboo. Where to next? The Bamboo 101 will help you get started. If you are using other Atlassian products, take a look at the Integration Guide.

Upgrade Guide

The Bamboo Upgrade Guide is for people who are upgrading their instance of Bamboo. Start by reading the latest Release Notes and version-specific Upgrade Guide for the version to which you are upgrading, then download Bamboo and follow the main Upgrade Guide.

Developer Resources

These resources are for software developers who want to create their own plugins for Bamboo. Take a look at the Development Hub, including the REST API documentation. You may also find the Bamboo Developers Forum useful (click here to subscribe).

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 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.
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 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
5. Please ensure that your Bamboo home directory is not located inside the Bamboo installation directory.
6. Start the Bamboo server by running ./bamboo.sh console in the installation root directory.
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 Executable 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. This opens the Create Plan introductory page.
2. Click the large 'Create a New Plan' button and commence creating your new Plan as described in the Creating a Plan topic.

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 Triggering a Plan Build 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 Executable 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.

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.
   - An elastic agent process runs in an elastic instance and will automatically start when an instance is started.
   - Enter ‘1’ in the 'Number of instances' field.
   - Select the 'Default' elastic instance 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 instances 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 author.
   - '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
     - '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
     - REV30 - Generating Reports across Multiple Plans
   - '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.
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' — a convenient summary of information that is relevant to you (only appears if you have logged in to Bamboo):
  - 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 statistics (only appears if your Bamboo User Profile has been associated with your Author Name).

On this page:

- Using the Dashboard
- Notes

Using the Dashboard

The Dashboard is your Bamboo 'home' page. The Dashboard contains three tabs:
Screenshot above: Bamboo Dashboard-'All Plans' tab

You can:

- click the **project name** (e.g. ‘Bamboo Testing’) to view the Plans in the project.
- click the **plan name** (e.g. ‘Acceptance Test JDK 1.6’) to **view the Plan details**.
- click the **build number** (e.g. ‘7823’) 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).

The icon next to a build number 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 queuing a build for this plan in the Build Queue.
- 🟢 Bamboo is currently executing a build for this plan.
- ✟ This plan has been disabled.

**Notes**

- You can return to the Dashboard from anywhere in Bamboo by clicking the ‘**Home**’ link in the top navigation bar.

**Related Topics**

- Working with Projects, Stages, Plans and Jobs
- Working with Build Results

**Viewing Bamboo’s Current Activity**

On this page:

- Using the Current Activity page
  - Bamboo Users
  - Bamboo Administrators

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:
   - All plans which are currently being built by an online agent in the 'Building' section at the top-left of the page,
   - Plans which are waiting to be built in the 'Queue' section below, and
   - A list of completed builds in the 'Recently Built' section at the right of the page.

Screenshot: Bamboo Dashboard-'Current Activity' tab
The 'Building' section shows the plans which are currently being built by an online agent. An agent can only run one plan at a time. Once an agent has completed this process, it can then commence running another plan waiting in the 'Queue' section (provided that that plan's requirements meet the agent's capabilities).

The Bamboo system in the screenshot above indicates 11 online agents currently in the process of building a plan, where:

- 'bamboo-agent-76.private.atlassian.com' is currently building a plan called 'Confluence Trunk - Main Build with Java 5'.
- 'wxp3x632msie6 (2038)' is currently building a plan called 'Confluence Trunk Selenium - Selenium, IE8 Batch 2', and so on.

The 'Queue' section shows the list of plans which are next in line to be built on the next appropriate online agent (shown in the 'Building' section) that becomes free. Plans at the top of the 'Queue' list are next in line to be built on the next appropriate free agent.

Once a plan in the 'Building' section has completed, it is moved to the 'Recently Built' section, which shows a list of the recently completed builds in reverse chronological order.

### Using the Current Activity page

In the 'Recently Built' section of this page, you can:

- Click a plan name (e.g. 'Bamboo - Export-import functional tests') to view the plan's summary page.
- Click a build number (e.g. 'BAM-EXPIMP-163') to view the build result.
- Click a 'Reason' (e.g. 'Updated by...') to view the code changes that triggered the build.
- Click the comment icon to add a comment to a build, so that you and others can keep track of any outcomes from this build.

**Screenshot: Commenting on a build**
If you are logged in as an ordinary Bamboo User or if you have anonymous access to a Bamboo site, you can do the following on this page:

- Click on a plan name in either the 'Building' or 'Queue' section of this page (e.g. 'Bamboo - Acceptance Test JDK 1.6') to view that plan's activity log.
- Click the 'X of Y online agents are building' link in the 'Building' section of the page to view the list of all available Bamboo agents. 'X' is the number of online agents that are currently building Bamboo plans and 'Y' is the total number of available online agents. For more information about the distinction between 'online' and 'offline' agents, refer to Monitoring Agent Status.

If you are logged in as a Bamboo Administrator, additional features on this page become available, such as the ability to:

- Reorder plans in the 'Queue'. To do this, move your mouse over the icon, click on it to 'grab' the plan, move the plan to its new position in the queue and then release the mouse button.
- Enable or disable agents. To do this, move your mouse over the 'X of Y online agents are building' link and in the resulting 'Online...
Agents' popup balloon, click the 'Enable' or 'Disable' buttons next to the agents you wish to enable or disable. You can also enable or disable all online agents by clicking the appropriate ‘...all’ buttons in the popup balloon. For more information about managing agents, please refer to the Configuring Agents and Monitoring Agent Status sections.

*Screenshot: Enabling or Disabling Online Agents*

- View individual agents details. For more information, please refer to Viewing Bamboo's Agents.

**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.

Screenshot: 'My Bamboo--My Favourite Plans'
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 the Wallboard

A development team can benefit from setting up a dedicated monitor to display Bamboo’s latest build results using Bamboo’s wallboard feature, 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 you have permission to see on the dashboard, as a wallboard,

1. Log into Bamboo.
   - If your Bamboo administrator has allowed anonymous access, this step is optional.
2. Do either of the following:
   - On the Bamboo Dashboard, click the ‘Wallboard’ dropdown and select ‘All Plans’.
   - OR
   - Type the following URL into your browser, but replace ‘bambooserver’ with the real name of your Bamboo server:
     
     http://bambooserver:8080/bamboo/telemetry.action

To display Bamboo’s latest build results, for your favourite plans only, as a wallboard,

1. Log into Bamboo.
2. Do either of the following:
   - On the Bamboo Dashboard, click the ‘Wallboard’ dropdown and select ‘Favourite Plans’.
   - OR
   - Type the following URL into your browser, but replace ‘bambooserver’ with the real name of your Bamboo server:
     

Hints and Tips:

- If your wallboard is displayed on a touchscreen (such as an iPad) or its content can be accessed with a ‘human interface device’, such as a mouse, then touching or clicking a build result on the wallboard shows more information about that build.
Once you are viewing the wallboard in your browser window, set your browser to 'full screen' mode to make the wallboard fill your entire screen.

- You can set your browser to full screen mode by pressing the **F11** key for most common browsers on Windows and UNIX/Linux-based systems and the **Shift + Command + F** key combination for Firefox on Mac OS X.
- If you are going to display the wallboard 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

On this page:

- Viewing All Available Bamboo Agents
- Viewing a Specific Bamboo Agent as a Bamboo User
- Viewing a Specific Bamboo Agent as a Bamboo Administrator

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

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

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.

### Viewing All Available Bamboo Agents

To view all of Bamboo's available agents,
1. Click the ‘Home’ link in the top navigation bar. This opens the Dashboard.
3. Click the ‘X of Y online agents are building’ link in the ‘Building’ section of the page. ’X’ is the number of online agents that are currently building Bamboo builds and ‘Y’ is the total number of available online agents.

Screenshot: Accessing the List of All Bamboo Agents

![Building Tab](image)

4. A list of all agents in your Bamboo system will be displayed (see screenshot below).

Screenshot: View Agents

<table>
<thead>
<tr>
<th>Local Agents</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Status</td>
</tr>
<tr>
<td>Core 1</td>
<td>✔️ Building - JIRA-FUNC-JDK1.6-4572</td>
</tr>
<tr>
<td>Core 2</td>
<td>✔️ Building - JIRA-FUNC-JDK1.6BATCH9-9220</td>
</tr>
<tr>
<td>Core 3</td>
<td>✔️ Building - JIRA-FUNC-JDK1.6BATCH9-2148</td>
</tr>
<tr>
<td>Unit Test Core</td>
<td>✔️ Building - JIRA-FUNC-JDK1.6-7254</td>
</tr>
</tbody>
</table>

Remote Agents

Remote agents run on computers other than the Bamboo server.

There are currently 35 remote agents online (20 non-elastic and 15 elastic). A maximum of 100 agents are supported by your license.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>attassign15-Agent 1</td>
<td>✔️ Building - RAMIN-COMPINST-220</td>
</tr>
<tr>
<td>attassign15-Agent 2</td>
<td>✔️ Building - RAMIN-COMPSTD-208</td>
</tr>
<tr>
<td>attassign15-Agent 3</td>
<td>✔️ Building - SEL-ENTFDGMBATCH9-968</td>
</tr>
</tbody>
</table>

Viewing a Specific Bamboo Agent as a Bamboo User

To view a specific agent as a standard Bamboo user,

1. Click the ‘Home’ link in the top navigation bar. This opens the Dashboard.
3. Click the name of the agent you wish to view in the ‘Building’ section of the page. To access the name of the agent, you can either:

- Click its name in the list of builds as shown in the following screenshot:
  
  ![Screenshot: Accessing the Agent Name in the List of Builds]

- Click the ‘X of Y online agents are building’ link in the ‘Building’ section of the page to view the list of all Bamboo Agents.
  
  ![Screenshot: Choosing an Agent on the List of All Bamboo Agents Page]

  * ‘X’ is the number of online agents that are currently building Bamboo builds and ‘Y’ is the total number of available online agents.

- Click the name of an agent in the ‘Local Agents’ section of this page (see screenshot below).

  ![Local Agents]

  You can view agents listed in the ‘Remote Agents’ or ‘Elastic Images’ sections of this page by clicking their names too. However, you will require Bamboo administrator permissions to access these details. Upon clicking one of these agent’s names, you may be prompted to log in to Bamboo.

  ![Remote Agents]

  There are currently 28 remote agents online (14 non-elastic and 15 elastic). A maximum of
4. The details of the selected agent will be displayed. If you have not changed the default view, the ‘Recent Builds’ view will be shown. If not, click the ‘**Recent Builds**’ tab.

*Screenshot: View Agent*

<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>PLUGINS-SUBPLUGHEAD-3173</td>
<td>Dependant of BASIC-JDK16-2236</td>
<td>53 minutes ago</td>
<td>2 minutes</td>
<td>279 passed</td>
</tr>
<tr>
<td>✔️</td>
<td>REL_ENTTFOOTTRIGGER-1113</td>
<td>Dependant of BASIC-JDK16-9236</td>
<td>50 minutes ago</td>
<td>10 seconds</td>
<td>No tests found</td>
</tr>
<tr>
<td>✔️</td>
<td>DEPLOY.TRUNK-15</td>
<td>Manual build by Andreas Knecht</td>
<td>58 minutes ago</td>
<td>7 minutes</td>
<td>No tests found</td>
</tr>
<tr>
<td>✔️</td>
<td>BASIC-JDK16-9236</td>
<td>Updated by Scott Hammond</td>
<td>1 hour ago</td>
<td>6 minutes</td>
<td>12111 passed</td>
</tr>
<tr>
<td>✔️</td>
<td>JIRA/PRO/TEST-SUB/REVBRANCH-492</td>
<td>Scheduled build</td>
<td>4 hours ago</td>
<td>6 minutes</td>
<td>No tests found</td>
</tr>
<tr>
<td>✔️</td>
<td>JIRA/PRO/TEST-UWEBLOGIC/JSPHEAD-531</td>
<td>Scheduled build</td>
<td>5 hours ago</td>
<td>8 minutes</td>
<td>No tests found</td>
</tr>
<tr>
<td>✔️</td>
<td>GH-DEV-732</td>
<td>Updated by Jean Christophe Hurt</td>
<td>7 hours ago</td>
<td>1 minute</td>
<td>42 passed</td>
</tr>
</tbody>
</table>
• You can click on the ‘Executable Plans’ tab to view the plans that this agent is capable of building.  
*Screenshot: View Agent - Executable Plans*

**Executable Plans**

Plans whose requirements match the capabilities

1. Deploy JIRA to Maven - Crowd Embedded - JDK 1.6
2. Deploy JIRA to Maven - JDK 1.6 Branch
3. Deploy JIRA to Maven - JDK 1.6 Trunk
4. Failing Build Debug - Unit Test Clone 2
5. GreenHopper - Default JDK 1.5
6. JIRA Code Coverage - Functional - Unit Test Coverage JDK 1.6
7. JIRA Code Coverage - Unit Test Coverage JDK 1.6
8. JIRA Crowd Embedded - JIRA 1.6 Func - bundles plugins disabled
9. JIRA Crowd Embedded - JDK 1.6 Func Batch 01
10. JIRA Crowd Embedded - JDK 1.6 Func Batch 02
11. JIRA Crowd Embedded - JDK 1.6 Func Batch 03
12. JIRA Crowd Embedded - JDK 1.6 Func Batch 04

• You can also 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>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>false</td>
<td>View</td>
</tr>
<tr>
<td>elastic</td>
<td>false</td>
<td>View</td>
</tr>
<tr>
<td>func.test runner</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>local</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>maven1.doyer2.license</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>maven1.doyer2.plugins</td>
<td>false</td>
<td>View</td>
</tr>
<tr>
<td>resinx3.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>unit.test.runner</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>weblogic2.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>websphere81.installation</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>Builder Label</th>
<th>Path</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash (Command)</td>
<td>/bin/bash</td>
<td>View</td>
</tr>
<tr>
<td>Maven 1 (Maven)</td>
<td>amportools/maven/maven-1.0.2</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.x)</td>
<td>amportools/maven/current</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.1.0 (Maven 2.x)</td>
<td>amportools/maven/apache-maven-2.1.0</td>
<td>View</td>
</tr>
<tr>
<td>Script (Script)</td>
<td>amportools/maven/apache-maven-2.1.0</td>
<td>View</td>
</tr>
</tbody>
</table>

**JDK**

JDK capabilities define the JDKs which are available to your build plans.

<table>
<thead>
<tr>
<th>JDK Label</th>
<th>Java Home</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK 1.4</td>
<td>jdk-1.4.2</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6</td>
<td>jdk-1.6.2</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6</td>
<td>jdk-1.6.2</td>
<td>View</td>
</tr>
<tr>
<td>Special JDK 1.6</td>
<td>homajira2-bamboo-j2ae/astxdrjdk-1.6</td>
<td>View</td>
</tr>
</tbody>
</table>
Viewing a Specific Bamboo Agent as a Bamboo Administrator

To view a specific agent as a Bamboo administrator,

1. Click the 'Home' link in the top navigation bar. This opens the Dashboard.
2. Navigate to the 'Current Activity' tab of the Dashboard.
3. Click the name of the agent you wish to view in the 'Building' section of the page. To access the name of the agent, you can either:
   - Click its name in the list of builds as shown in the following screenshot:
     Screenshot: Accessing the Agent Name in the List of Builds
   - Hover your mouse pointer over the 'X of Y online agents are building' link and in the resulting popup, click on the agent's name as shown in the following screenshot:
     Screenshot: Accessing the Agent Name via the Popup
4. The details of the selected agent will be displayed in the Bamboo administration console, where you can further configure this agent and its capabilities. By default, the ‘Capabilities’ view will be shown. If not, click the ‘Capabilities’ tab.

**Screenshot: View Agent**

### Agent-Specific Capabilities

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom.

The following capabilities are specific to this agent:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>jdk.test.runner</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

### Custom

‘Custom’ capabilities are key-value pairs that define particular characteristics of an agent (e.g. operating system=Windows XP, 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>container</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>elastic</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>local</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>maven1.clover.license</td>
<td>importtools/clover/license</td>
<td>View</td>
</tr>
<tr>
<td>maven1.clover2.plugin</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>resin3x.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>unit.test.runner</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>weblogic92.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>websphere61.installation</td>
<td>true</td>
<td>View</td>
</tr>
</tbody>
</table>

### Shared Capabilities

The following ‘shared’ capabilities are inherited by this agent. Note that the value of a shared capability is overridden by the value of an agent-specific capability with the same key/label (if one exists).

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>elastic</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>local</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>maven1.clover.license</td>
<td>importtools/clover/license</td>
<td>View</td>
</tr>
<tr>
<td>maven1.clover2.plugin</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>resin3x.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>unit.test.runner</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>weblogic92.installation</td>
<td>true</td>
<td>View</td>
</tr>
<tr>
<td>websphere61.installation</td>
<td>true</td>
<td>View</td>
</tr>
</tbody>
</table>
- You can click on the 'Executable Plans' tab to view the plans that this agent is capable of building.

_Screenshot: View Agent - Executable Plans_

### Executable Plans

Plans whose requirements match the capabilities.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Last built</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy JIRA to Maven - Crowd Embedded JDK1.6</td>
<td><strong>DEPLOY-CROWDEMJBDED-37</strong> 2 weeks ago</td>
</tr>
<tr>
<td>Deploy JIRA to Maven - JDK1.6 Branch</td>
<td><strong>DEPLOY-BRANCH-24</strong> 1 month ago</td>
</tr>
<tr>
<td>Deploy JIRA to Maven - JDK1.6 Trunk</td>
<td><strong>DEPLOY-TRUNK-15</strong> 1 hour ago</td>
</tr>
<tr>
<td>Failing Build Debug - Unit Test Clone 2</td>
<td></td>
</tr>
<tr>
<td>GreenHopper - Default JDK1.6</td>
<td></td>
</tr>
<tr>
<td>JIRA Code Coverage - Functional - Unit Test Coverage JDK1.6</td>
<td><strong>COV-UNIT-13</strong> 9 hours ago</td>
</tr>
<tr>
<td>JIRA Code Coverage - Unit Test Coverage JDK1.6</td>
<td></td>
</tr>
<tr>
<td>JIRA Crowd Embedded - JDK1.6 Func - bundled plugins disabled</td>
<td><strong>JIRA CROWD-FUNC-150</strong> 19 hours ago (Last successful build JIRA CROWD-FUNC-150) 3 days ago)</td>
</tr>
<tr>
<td>JIRA Crowd Embedded - JDK1.6 Func Batch 01</td>
<td>Never built</td>
</tr>
<tr>
<td>JIRA Crowd Embedded - JDK1.6 Func Batch 02</td>
<td>Never built</td>
</tr>
<tr>
<td>JIRA Crowd Embedded - JDK1.6 Func Batch 03</td>
<td>Never built</td>
</tr>
<tr>
<td>JIRA Crowd Embedded - JDK1.6 Func Batch 04</td>
<td><strong>JIRA CROWD-FB04-32</strong> 3 weeks ago (Never built successfully)</td>
</tr>
</tbody>
</table>
Working with Projects, Stages, Plans and Jobs

Project

A project is a collection of Plans. Projects enable you to easily group and identify plans which are logically related to each other. They are especially useful when generating reports across multiple Plans.

Plans

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Stages

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

## Jobs

A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job's Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

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.

### Structure of a Plan
Related Topics

- Working with Plans
- Working with Stages
- Working with Jobs

Working with Plans

About Plans

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

Navigating to a Plan
To navigate to a Plan,

1. Click **Home** to go to the Dashboard and click the **All Plans** tab.
2. In the list of Plans, click the name of the desired Plan. The Plan’s **Plan Summary** page will be displayed.

**Related Topics**

For more information on working with Plans, see the following topics:

**Viewing a Plan’s Details**

**Viewing a Plan’s Details**

A **Plan** defines everything about your entire build process. Plans:

- consist of one or more **Jobs**, which are organised into one or more **Stages**;
- contains a single ‘Default Job’ in a single Stage, after **creating a new Plan**;
- define default settings for what gets built by Jobs in the Plan (i.e. the ‘default source repository’);
- define how the Plan’s build is triggered;
- who will be **notified** of the Job’s **build result**;
- define who has **permission** to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a **Project**.

Projects and plans can only be configured by Bamboo administrators (see **Creating a Plan**).

**On this page:**

- Viewing a Plan’s Details
- Notes

**Viewing a Plan’s Details**

**To view a plan’s details:**

1. Navigate to the desired Plan, as follows:
   - If you are viewing the Dashboard, locate and click the Plan’s name in the list, or
   - If you are viewing a Job or **Build Result**, click the Plan name in the breadcrumb links at the top of the screen.
2. Click a tab to view information about the Plan. The tabs are described below:
   - **‘Plan Summary’** — View key information about the Plan, as shown in the diagram below.
   - **‘Recent Failures’** — View information about recent failures of the Plan, including the builds that failed, links to the build results, time taken to fix, etc.
   - **‘History’** — View the full history of builds of the Plan.
   - **‘Issues’** — (This tab only displays if your administrator has **integrated Bamboo with JIRA**). You can view JIRA issues linked to builds for your Plan on this tab.
   - **‘Configuration’** — (This tab only displays if you are an **administrator** for the Plan). You can view and edit the Plan’s configuration on this tab. For details please see the Bamboo Administrator’s Guide.
   - **‘Showing Last 25 Builds’** menu — (This menu only displays if you are viewing the ‘Plan Summary’, ‘Recent Failures’ or ‘Issues’ tabs). The data (e.g. percentages, graphs, issues list, etc) on these tabs will change depending on the value selected.
   - **‘Actions’** menu — (This menu only displays if you are an **administrator** for the Plan). Access common functions for the Plan via this menu, e.g. run Plan, disable Plan, configure Plan, etc.

**Diagram above: Plan Summary (annotated)**

**Notes**

**Related Topics**

**Using the Bamboo Dashboard**
**Viewing a Build Result**
**Integrating Bamboo with JIRA**
**Configuring a Plan’s Permissions**

**Viewing the JIRA Issues linked to the Builds in a Build Plan**

If you use 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.
Viewing the JIRA Issues linked to Builds in a Build Plan

To view the JIRA issues linked to builds in a build plan:

1. Navigate to the desired Plan, as described on Working with Plans.
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 above: JIRA Issues linked to Builds in a Build Plan

Notes

Related Topics

Viewing the JIRA Issues for a Build Result
Linking JIRA Issues to a Build

Working with Stages

About Stages

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

Viewing the Stages of a Plan

To view the Stages of a Plan,

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. In the list of Plans, click the name of the desired Plan. The Plan's Plan Summary page will be displayed.
   - The Plan Navigator on the left will display the Stages of the Plan. The Stages will be shown with a grey background.
     - Hover your mouse over the Stage to display a description of the Stage.
A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job's Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

Navigating to a Job

To navigate to a Job in a Plan,

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. In the list of Plans, click the name of the desired Plan. The Plan's Plan Summary page will be displayed.
3. In the Plan Navigator on the left, click the name of the desired Job.

Related Topics

For more information on working with Jobs, see the following topics:

Viewing a Job Viewing a Job's Maven Dependencies Viewing the Clover Coverage Summary for a Job
Viewing a Job

To navigate to a Job:

1. Click ‘Home’ to go to the Dashboard and click the ‘All Plans’ tab.
2. In the list of Plans, click the name of the desired Plan. The Plan’s Plan Summary page will be displayed.
3. In the ‘Plan Navigator’ on the left, click the name of the desired Job. The Job will be displayed (see diagram below)

Notes

Working with Jobs

Viewing a Job’s Maven Dependencies

If you have configured a Job to use a Maven builder (Maven 2 or later), you can choose to have dependencies generated from your Maven pom.xml (read the documentation for setting up Maven as a builder for instructions). After the initial build, Maven will parse the pom.xml file, determine the artifacts produced by the build and generate the dependencies. You can view these dependencies in two places:

- On the ‘Dependencies’ tab when configuring your Plan, as described in Setting up Build Dependencies.
- On the ‘Artifacts’ tab when viewing a Job’s build result, as described below.

Viewing the Maven Dependencies for a Job

Before you begin:

- The Maven dependencies for a build will only become known to Bamboo after a build. If you cannot see the Maven dependencies for a build, try running it first without triggering any other dependencies. See Modifying Multiple (Bulk) Plans if you want to run multiple builds.

To view the Maven dependencies for a Job’s build result:

1. Navigate to the desired Job, as described on Working with Jobs.
2. Click the desired build result number in the ‘Recent History’ of the Job Summary.
3. Click the ‘Artifacts’ tab for the build results. The produced Maven artifacts and Maven artifact dependencies will be listed.
Viewing the Clover Coverage Summary for a Job

If you use Atlassian's Clover and your Job 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 Job.

To view the Clover coverage summary for a Job:

1. Navigate to the desired Job, as described on Working with Jobs.
2. Click the 'Clover' tab. Graphs of the coverage history and lines of code history will display.
About Builds

A build is the execution of either a Plan or a Job. The execution of a Plan is referred to as a 'Plan build' and that of a Job is a 'Job build'.

About Build Results

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.
- 'Incomplete' — the build was not completed, e.g. it may have been stopped manually.

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.

Related Topics

For more information on working with build results, see the following topics:

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

Viewing a Build Result

The instructions on this page describe how to view the build results for a Plan.

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.
- *Incomplete* — the build was not completed, e.g. it may have been stopped manually.

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*.

On this page:

- Viewing the Most Recent Job Build Result of a Plan
- Viewing all Build Results for a Plan
- Viewing all Build Results for a Job
- Notes

Viewing the Most Recent Job Build Result of a Plan

To view the most recent Job build result of a Plan:

1. Go to the **Dashboard**.
2. Locate the plan in the list, then click the Build Number.

Viewing all Build Results for a Plan

To view all build results for a plan:

1. Navigate to the desired Plan, as described on Working with Plans. The ten most recent builds will be displayed in the ‘Recent History’ section on the ‘Plan Summary’ tab.
2. Click the ‘History’ tab to view all builds for the Plan.

Viewing all Build Results for a Job

To view all build results for a Job:

1. Navigate to the desired Job, as described on Working with Jobs. The ten most recent builds will be displayed in the ‘Recent History’ section on the ‘Job Summary’ tab.
2. Click the ‘History’ tab to view all builds for the Job.

A build result looks like this:
The 'Summary' tab (shown in the screenshot above) displays a snapshot of the build result. You can also click the 'Comment' button to add a comment.

- The 🔄 icon represents a successful build.
- The 🗡️ icon represents a build that was not completed. For example, it may have been stopped manually.
- The 🙅‍♂️ icon represents a failed build.
- 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 '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 'Issues' tab to view any JIRA issues that relate to this build result (if applicable).
  - Click the 'Clover' tab to view the Clover code-coverage that relate to this build result (if applicable).

Notes

Related Topics

Viewing the Code Changes that Triggered a Build
Viewing a Build's Artifacts
Viewing a Build Log
Viewing the Metadata for a Build Result
Stopping an Active Job Build
Stopping an Active Plan Build

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. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired 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 the Source Repository for a Plan in the Bamboo Administrator’s Guide.
Bamboo 3.1 Documentation

**Code Changes**

This build was triggered at the scheduled time of 12:00 AM

**Przemysław Braski**

BAM-8202 - Builds seem to get picked off the queue in random order


**Piotr Stefaniak**

CR-BAM-2004 BAM-8202 Mercurial. Cloning from cache to working directory should be instant (performance) code review fixes

- /atlassian/bamboo/trunk/components/bamboo-plugins/bamboo-plugin-mercurial/src/main/java/com/atlassian/bamboo/plugins/hg/HgRepositoryAccessImpl.java (version 142382) (diffs)

**Krystian Brazulewicz**

BAM-7876 Replace com.google.collections.google-collections with com.google.guava.guava

- /atlassian/bamboo/trunk/components/bamboo-plugins/bamboo-plugin-restbom.xml (version 142383) (diffs)

**Slawek Ginter**

BAM-7200 - single cache delete schedules remote agent action

- /atlassian/bamboo/trunk/components/bamboo-plugins/bamboo-plugin-mercurial/src/test/java/com/atlassian/bamboo/plugins/hg/HgCacheHandlerTest.java (version 142385) (diffs)
- /atlassian/bamboo/trunk/components/bamboo-plugins/bamboo-plugin-mercurial/src/main/resources/com/atlassian/bamboo/plugins/hg/18n.properties (version 142385) (diffs)
- /atlassian/bamboo/trunk/components/bamboo-plugins/bamboo-plugin-mercurial/src/test/java/com/atlassian/bamboo/plugins/hg/DeleteHgCacheDirectoryTest.java (version 142385) (diffs)

**Screenshot above: Code Changes**

**Notes**

- **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.

**Related Documentation**

- Triggering a Plan Build
- Specifying the Source Repository for a Plan

**Viewing a Build’s Artifacts**

After a build has run, you can view the artifacts that were produced by all of the Jobs in the plan. You can also view the latest version of an artifact from the most recent build.

Artifacts are files created by a Job build (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual Jobs.

**On this page:**

- Viewing the Artifacts for a Build
- View the Latest Version of an Artifact from the Most Recent Build

**Viewing the Artifacts for a Build**

To view a build’s artifacts:
1. Go to the build result. See Viewing a Build Result for instructions.
2. Click the 'Artifacts' tab. The artifacts produced by the Jobs in the plan will be displayed. The artifact definitions for a Job determine which artifacts are kept and which artifacts are shared with other Jobs in the plan.
   - The artifacts that are marked as shared (in the artifact definitions) are listed under 'Shared Artifacts'.
   - The artifacts that are not marked as shared (in the artifact definitions) are listed under 'Job Artifacts'.

**Viewing the Latest Version of an Artifact from the Most Recent Build**

If you would like to view the latest version of an artifact from the most recent build, you can manually edit the build artifact URL to retrieve it.

*To view the latest version of an artifact from the most recent 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'.
   - If you need to log in to view the artifacts, you can append os_username and os_password parameters to the URL to access the files.

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

---

**Screenshot above: Build Artifacts**

**Related Topics**

- Configuring a Job's Build Artifacts
- Configuring Artifact Sharing between Jobs

**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 Job's source-code and executing the tests.

**On this page:**

- Viewing a Build Log
- Notes

**Viewing a Build Log**

To view a build log:

1. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. Click the 'Logs' tab.
   - Click ‘View’ for the desired log, to view the it in Bamboo.
   - Click 'Download' to download a text file of the log.
1. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. Click the 'Metadata' tab.
1. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. Click the ‘Clover’ tab.
1. Bamboo also provides data on code-coverage trends for a plan over a period of time. For details see: 'Clover Code Coverage' Report.

Related Topics
Working with Build Results
'Clover Code Coverage' Report
'Clover Lines of Code' Report

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.

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.

Viewing the JIRA Issues for a Build Result

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. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. The 'JIRA Issues' section will display two of the issues linked to the build. See 'Screenshot: JIRA Issues for a Build Result — Summary tab' below.
3. 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 above: JIRA Issues for a Build Result — Summary tab
1. Bamboo 3.1 Documentation

2. Screenshot above: JIRA Issues for a Build Result — Issues tab

3. Related Topics

4. Viewing the JIRA Issues linked to the Builds in a Build Plan

5. Linking JIRA Issues to a Build

6. 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.

7. On this page:
   - Editing Issues Links for your Build
   - Manually Adding New Issue Links to a Build
   - Notes

8. Editing Issues Links for your Build

Issue links that have been created automatically or manually, can be edited or removed from the build.

9. 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.
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.

![Screenshot above: Adding new JIRA Issue Links to a Build](image)

**Notes**

**Related Topics**

- Viewing the JIRA Issues for a Build Result
- Viewing the JIRA Issues linked to the Builds in a Build Plan

**Working with Tests**

**Related Topics**

For information on working with tests, see the following topics:

- Viewing Test Results for a Build
Viewing a Test’s History

Viewing Test Statistics for a Job

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.

On this page:
- Viewing the Tests for a Build
- Notes

Viewing the Tests for a Build

To view the tests for a particular build:

1. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. Click the ‘Tests’ tab.
   - Click the test name to see a particular test’s results for other builds.

Screenshot above: Test Results for a Build

Notes

- 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.

Related Topics
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. Navigate to the build results for the Plan/Job, as described in Viewing a Build Result, and click the desired build result.
2. Click the 'Tests' tab.
3. Click the name of the test in which you are interested.
4. 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.

Screenshot above: Test History

Notes

Related Topics

Working with Tests
Viewing Test Statistics for a Job

Bamboo provides a summary of test results across all of a Job’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?

### On this page:
- Viewing Test Statistics for a Job’s Builds
- Notes

### Viewing Test Statistics for a Job’s Builds

**To view the test statistics for all of a Job’s builds:**

1. Navigate to the desired Job, as described in Working with Jobs.
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).

   - To view a test’s history, click the test name.

---

**Screenshot above: Test Statistics for a Job**
1. Most Failing  
2. Longest to Fix  
4. Number of Tests

_Screenshots above: Test Statistics for a Job - Tabs_

**Notes**

**Related Topics**

To compare the number of tests over time across _multiple plans_, view the 'Number of Tests' Report.

**Reporting on Plan Trends**

**Related Topics**

For information on reporting on Plan trends, see the following topics:

- *Viewing Build Statistics for a Plan*
- *Generating Reports across Multiple Plans*

**Viewing Build Statistics for a Plan**

You can view build statistics for a Plan on the Plan's summary.

On this page:
- Viewing Build Statistics for a Plan on the Plan Summary

**Viewing Build Statistics for a Plan on the Plan Summary**

A _Plan_ defines everything about your entire build process. Plans:

- consist of one or more _Jobs_, which are organised into one or more _Stages_;  
- contain a single 'Default Job' in a single Stage, after creating a new Plan;  
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');  
- define how the _Plan's build_ is _triggered_;  
- who will be _notified_ of the Job's _build result_;  
- define who has _permission_ to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a _Project_.

Projects and plans can only be configured by Bamboo administrators (see *Creating a Plan*).

On this page:
- Viewing a Plan's Details  
- Notes

**Viewing a Plan’s Details**

To view a plan’s details:

1. Navigate to the desired Plan, as follows:  
   - If you are viewing the _Dashboard_, locate and click the Plan’s name in the list, or  
   - If you are viewing a Job or Build Result, click the Plan name in the breadcrumb links at the top of the screen.
2. Click a tab to view information about the Plan. The tabs are described below:
   - 'Plan Summary' — View key information about the Plan, as shown in the diagram below.
• 'Recent Failures' — View information about recent failures of the Plan, including the builds that failed, links to the build results, time taken to fix, etc.
• 'History' — View the full history of builds of the Plan.
• 'Issues' — (This tab only displays if your administrator has integrated Bamboo with JIRA). You can view JIRA issues linked to builds for your Plan on this tab.
• 'Configuration' — (This tab only displays if you are an administrator for the Plan). You can view and edit the Plan’s configuration on this tab. For details please see the Bamboo Administrator’s Guide.
• 'Showing Last 25 Builds' menu — (This menu only displays if you are viewing the 'Plan Summary', 'Recent Failures' or 'Issues' tabs). The data (e.g. percentages, graphs, issues list, etc) on these tabs will change depending on the value selected.
• 'Actions' menu — (This menu only displays if you are an administrator for the Plan). Access common functions for the Plan via this menu, e.g. run Plan, disable Plan, configure Plan, etc.

Diagram above: Plan Summary (annotated)

Notes

Related Topics
Using the Bamboo Dashboard
Viewing a Build Result
Integrating Bamboo with JIRA
Configuring a Plan’s Permissions

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.

Generating Reports across Multiple Plans

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
   - '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
   - REV30 - Generating Reports across Multiple Plans

Additionally, your Bamboo administrator may configure custom reports by using plugins. For details please see the Bamboo Administrator’s Guide.

3. 'Build plans' — choose the plan(s) on which you want to report. You can use the <Ctrl> key to select multiple plans.
   (Project names are shown in italics, e.g. 'Geronimo SVN'. Plan names are shown in non-italics, e.g. 'Main Build'.
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.
‘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.
'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.
'Build Duration' Report

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.

Note, 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.

'Screenshot above: Sample report 'Build Duration per Plan'

'Clover Code Coverage' Report

You can choose the plan(s) and time period on which you want to report.

Note, 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.
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.

'S Clover Lines of Code' Report

You can choose the plan(s) and time period on which you want to report.

Note, 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.
'Clover Lines of Code' Report

Provides an indication of the size of the codebase for the build.

Screenshot above: 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.

'Screenshot above: 'Number of Build Failures' Report'
Screenshot above: Sample report ‘Number of Build Failures per Plan’

'Percentage of Successful Builds' Report

You can choose the plan(s) and time period on which you want to report.

<table>
<thead>
<tr>
<th>Chart</th>
<th>Data Table</th>
<th>Builds</th>
</tr>
</thead>
</table>

'Screenshot above: 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.
Time to Fix

How long does it take on average to fix problems? This provides an indication of how quickly breakages are resolved for the plan.

Screenshot above: Sample report 'Time to Fix per Plan'

'Number of Tests' Report

You can choose the plan(s) and time period on which you want to report.
REV30 - 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.

Generating Reports across Multiple Plans

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:
   - '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).
3. 'Build plans' — choose the plan(s) on which you want to report. You can use the <Ctrl> key to select multiple plans.
   - Project names are shown in italics, e.g. 'Geronimo SVN'. Plan names are shown in non-italics, e.g. 'Main Build'.
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.
Custom Reports and Statistics

Compare trends between different plans. You can choose the different reports and the plans you wish to compare in the form below.

**Report Parameters**

<table>
<thead>
<tr>
<th>Report</th>
<th>Build Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many builds are triggered in a given time period? This indicates the level of activity for the plan.</td>
<td></td>
</tr>
</tbody>
</table>

**Build Plans**

- **Artifact Sharing Dogfooding**
  - Artifact sharing
  - CI Tests
  - Maven Sharing CI Tests
  - Maven Sharing CI Tests with Maven 3

- **Bamboo Database Test**
  - PostgreSQL
  - PostgreSQL 8.4

- **Bamboo Main**
  - CI Tests
  - Extras
  - Stable CI Tests
  - Stable Extras

- **Bamboo Plugins**
  - Atlassian Command Line Interface for Bamboo
  - Bamboo ClearCase Plugin
  - Bamboo FindBugs Plugin
  - Bamboo JIRA Releases Plugin
  - Bamboo JMeter Aggregator Plugin
  - Bamboo Sandbox Plugin

**Group By**

| Group By | Auto |

**Date Filter**

| Date Filter | Last 90 Days |

Submit

*Screenshot above: Report Parameters--Build Plans*

**Notes**

**Related Topics**

**Reporting on Plan Trends**

**About 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.

**Related Topics**

For information on reporting on author trends, see the following topics:

*Viewing Build Statistics for all Authors*
Viewing Build Results for an Author

Generating Reports on Selected Authors

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.

On this page:
- Viewing a Summary of All Author Statistics
- Notes

Viewing a Summary of All Author Statistics

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).
   - Click any author’s name to see their recent build results.

![Screenshot above: Authors Summary](image)

Notes

Related Topics

Reporting on Author Trends

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.
Viewing an Author's Build Results

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.
   
   If your Bamboo User Profile has not yet been associated with your Author Name, there will be no 'User Details' tab.

4. 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.

Screenshots above: Viewing an author's build results (click to view larger images)

Notes

Related Topics

Reporting on Author Trends

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.
Generating a Report on Selected Authors

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 above: 'Report Parameters--Authors'

Notes

Related Topics

'Report Parameters 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' report

'Number of Build Failures per Author' Report

You can select the author(s) on whom you want to report.

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 Fixed per Author' Report

You can select the author(s) on whom you want to report.

Sample Report: 'Number of Builds Fixed per Author'

'Percentage of Successful Builds per Author' Report
You can select the author(s) on whom you want to report.

**Percentage of Successful Builds**

![Chart showing successful builds over time]

Sample Report: ‘Percentage of Successful Builds per Author’

**Working with Comments**

**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**.

**Related Topics**

For information on working with comments, see the following topics:

*Commenting about a Build Result*

*Viewing Comments about a Build Result*

*Viewing Code Check-in Comments*

**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.

---

**On this page:**

- Commenting on a Build Result
- Notes

**Commenting on a Build Result**

Before you begin:
You must be logged in to Bamboo before you can comment on a build result.

To comment on a build result:

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. From within the 'Build Result' screen, click the 'Comment' button.
3. Type your comment into the 'Comment' box, then click the 'Add' button.
Viewing Comments About a Particular Build Result

To view comments about a particular build result:

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. A list of all comments about this build result will be displayed in the ‘Summary’ tab, including author and timestamp. Comments added to the Job that produced the build result will also be displayed.

![Screenshot above: Comments about a build result and related Job](image)

Viewing Comments About Build Results on Plan/Job Summary

To view comments about a build result a Plan summary or Job summary:

1. Navigate to the desired Plan or Job, as described on Working with Plans and Working with Jobs.
2. The Plan's/Job's build results will be displayed in the ‘Recent History’ section of the Plan/Job Summary. The message icon (bambooicon_comment.gif) indicates that there are one or more comments about a particular build result. Hover your mouse over the icon to see the comment(s).

![Screenshot above: Viewing comments on a Plan/Job summary](image)

Related Topics

Working with Comments
Commenting about a Build Result
Viewing a Build Result

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.

### Viewing the Code Check-in Comments for a Particular Build Result

To view the code check-in comments for a particular build result:

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. The build's commit comment will be shown to the right of the screen, under the heading 'Code Changes'.

### Notes

- A note about build triggering* — There are a number of methods by which Bamboo can 'trigger' (i.e. execute) a Plan's build,

  - **Build Strategy:**
    - **Code updated** — a Plan's build is triggered whenever one or more authors checks in code.
    - **Scheduled build** — a Plan's build is triggered at scheduled times or specified time intervals.
    - **Manual build** — a Plan's build is only ever triggered manually.
    - **Initial clean build** — a Plan's build is triggered immediately after a new Plan has been created.

  - **Build Dependency:**
    - **Dependency** — a build can be triggered whenever another Plan's build is successfully completed.

The trigger method for a Plan's build is listed in the 'Reason' column on the Dashboard.

### Working with 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.

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.

### Related Topics

For information on working with labels, see the following topics:

- Labelling a Build Result
- Removing a Label from a Build Result
- Viewing Labelled Build Results
- Viewing Popular Labels

### 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.
Labelling a Build Result

Before you begin:

- You must be logged in to Bamboo before you can label a build result.

To label a build result:

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. Click the pencil icon (edit), next to the ‘Labels’ link in the ‘Details’ section.
3. Type the relevant label (or multiple labels, separated by commas or spaces). Note that the label will be saved in lowercase characters.
4. Click ‘Close’.

Notes

- You can view a list of existing labels by clicking the ‘Labels’ link.
- You can also label a build result via Instant Messaging (IM).

Related Topics
Working with Labels

Removing a Label from a Build Result

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.

The instructions on this page describe how to remove a label from a build result.

Removing a Label from a Build Result

Before you begin:

- You must be logged in to Bamboo before you can label a build result.

To remove a label from a build result:

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. Click the pencil icon (edit), next to the ‘Labels’ link in the ‘Details’ section.
3. Click the ‘X’ at the right of the label you want to remove.
4. Click the ‘Close’ button.

Notes

Related Topics
Working with Labels

Viewing Labelled 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.

The instructions on this page describe how to view all build results with a particular label.

### On this page:
- Viewing Labelled Build Results
- Notes

#### Viewing Labelled Build Results

**To view all build results which have a particular label:**

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. Click the *Labels* link in the *Details* section of the *Build Summary* tab. The list of all labels that are used in Bamboo will be displayed.
3. Click the label of interest. The list of all build results which have that label will be displayed.

#### Notes

**Related Topics**

- Working with Labels

### Viewing Popular 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.

When *labelling a build result*, it can be useful to see which labels are most popular, that is, most frequently used by your colleagues.

### On this page:
- Viewing Popular Labels
- Notes

#### Viewing Popular Labels

**To view the most popular labels:**

1. Navigate to the desired build result, as described on Viewing a Build Result.
2. Click the *Labels* link in the *Details* section.
3. Click the link *See also labels in All Projects*.
4. Click the link *By Popularity*.
5. This will display a list all labels that are used in Bamboo, by popularity. You can click any label to see a list of all build results which have that label.

#### Notes

**Related Topics**

- Working with Labels

### Subscribing to RSS Feeds

**About Bamboo RSS Feeds**
Bamboo aggregates key information about your builds into RSS feeds. You can subscribe to these feeds using any feed reader.

**Related Topics**

For information on subscribing to RSS Feeds, see the following topics:

- [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: ![Feed for all builds](#) or just the [failed builds](#).
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: ![Feed for all builds](#) or just the [failed builds](#).
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: ![Feed for all builds](#) or just the [failed builds](#).
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: ![Feed for all builds](#) or just the [failed builds](#).
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

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

Related Topics

For information on working with instant messenger (IM) notifications, see the following topics:

Labelling a Build Result via IM

Commenting 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.

On this page:
- Labelling a Build Result via IM
- Notes

Labelling a Build Result via IM

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.
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.

**On this page:**
- Commenting about a Build Result via IM
- Notes

**Commenting about a Build Result via IM**

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

About your User Profile

You can manage your user details, password, notifications preferences and a number of other preferences via your user profile.

Related Topics

For information on managing your user profile, see the following topics:

Changing your Password

Changing your Notification Preferences

Associating your Author Name with your User Profile

Changing your Personal Details

Viewing 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.

Before you begin:

- You must have the 'Edit' permission for a plan to add or remove notifications for it.

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 that displays.
   - '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.

Screenshot above: User Profile

Notes

Related Topics

Managing your User Profile

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.
Associating your Author Name with your User Profile

Before you begin:

- 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.

Screenshot above: Bamboo User Profile

Notes

Related Topics

Managing your User Profile

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.
Viewing your Notifications

To view your notifications:

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. The notifications that are applicable to you will be listed in the 'My Notifications section', with the following information:
   - 'Plan' — the build plan that this notification is set up for.
   - 'Event' — the build event that triggers this notification.
   - 'Notification Recipient' — user (i.e. your username), group or specific email/IM address that this notification is specified for.

   Please note, if the plan administrator has specified your email/IM address (of your user profile) for a notification, you will receive the notification, regardless of your personal notification preferences. You will need to contact the plan administrator, if you wish to opt out of the notification.

Notes

Related Topics

Managing your User Profile

Bamboo Administrator's Guide

Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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/failed 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.

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

Configuring Projects, Plans, Stages and Jobs

Projects

A project is a collection of Plans. Projects enable you to easily group and identify plans which are logically related to each other. They are especially useful when generating reports across multiple Plans.
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.

**Plans**

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Plan's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

**Stages**

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

**Jobs**

A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job's Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

**Tasks**

A Task is an operation that is run on a Bamboo working directory using an executable. Tasks are run sequentially within a single Job. Once a Task is defined in the Bamboo system, it can then be specified in Jobs by a Bamboo administrator. A Job can be configured to execute a number of Tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers or execute a Script.

**Builds**

A build is the execution of either a Plan or a Job. The execution of a Plan is referred to as a 'Plan build' and that of a Job is a 'Job build'.

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

**Related Topics**

- Configuring a Plan
- Configuring a Stage
- Configuring a Job

**Configuring a Plan**

**About Plans**

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

Navigating to a Plan

To navigate to a Plan,

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. In the list of Plans, click the name of the desired Plan. The Plan's Plan Summary page will be displayed.

Creating a Plan

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contain a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

Creating a Plan

Before you begin:

- The 'Create Plan' or 'Admin' global permission is required to create new Plans.

To create a Plan in Bamboo:

1. Click the Create Plan link in the top navigation bar. The Create Plan introductory page is displayed with the following options:
• 'Create a New Plan'  
• 'Clone an Existing Plan' — This option only appears if you have the 'Clone' and/or 'Admin' plan permission for at least one Plan on the Bamboo server.  
• 'Import a Maven 2 Project' — This option only appears if Maven 2 has been installed.

2. Click one of the options to start creating your Plan. Each of these options is described in the pages linked below:
   • Cloning an Existing Plan  
   • Creating a New Plan  
   • Importing a Plan from Maven 2

Screenshot above: Create Plan Introductory Page

Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>The repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Notes

Related Topics

Creating a New Plan  
Cloning an Existing Plan  
Importing a Plan from Maven 2
Cloning an Existing Plan

The instructions on this page describe how to clone an existing Plan to create a new Plan.

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

On this page:
- Cloning an Existing Plan
- Appendix - Build Strategies
- Notes

Cloning an Existing Plan

When you clone an existing Plan, you make a copy of that Plan and its entire configuration.

On the 'Clone an Existing Plan' page, complete the options in the following sections to clone an existing Plan:

1. Start the Create Plan wizard as described on Creating a Plan and click 'Clone an Existing Plan'.
2. In the 'Plan to clone' section and dropdown, select the Plan you wish to clone. Plans are listed in the format 'Project Name - Plan Name' and are organised by Project. Only Plans for which you have the 'Clone' and/or 'Admin' plan permission are shown.
3. In the 'Plan Details' section, enter or select the relevant items for the following fields:
   - **Project** — Select an existing project to which this Plan will belong. If this field is not available (because no projects have yet been defined on your Bamboo server) or you select 'New Project', complete the following fields:
     - **Project Name** — Enter a name for a new project or select an existing project. The plan which you create on this page will be associated with this project.
     - **Project Key** — Enter a unique key for this project or select an existing project key.
   - **Plan Name** — Enter a name for the new Plan.
   - **Plan Key** — Enter a unique key for the new Plan.
   - **Plan Description** — Enter a short sentence to describe this Plan.
4. In the Enable this Plan section, choose whether or not to enable this Plan. Enabling the Plan instructs Bamboo to commence executing builds of the Plan based on the Plan's Build Strategy (defined above). To enable this Plan, select the 'Yes please!' check box.
5. Click the 'Create' button to create the Plan and the Plan's Plan Summary page will be displayed.

When you next return to the Dashboard (by clicking 'Home' in the top left of the page), your new Plan (and new Project, if applicable) will be displayed in the 'All Projects' list. Bamboo will also automatically run an initial build for your new plan, unless the Plan you cloned uses the Manual & dependent builds only Build Strategy.

If you wish to configure more Plan options, please refer to Editing a Plan.
Screenshots above: Cloning an existing Plan

Appendix - Build Strategies

This table lists Bamboo’s available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will ‘poll’ the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>The repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Notes

- You can configure Bamboo to automatically start an initial build for a new Plan, even if the Plan you cloned uses the Manual & dependent builds only Build Strategy. To do so, add the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.

Related Topics

Creating a Plan
Creating a New Plan
Importing a Plan from Maven 2
Creating a New Plan

The instructions on this page describe how to create a completely new Plan, i.e. not cloning an existing Plan or importing a Plan from Maven 2.

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contain a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

On this page:

- Step 1. Create a New Plan
- Step 2. Configure Tasks for the New Plan
- Appendix - Build Strategies
- Notes

Step 1. Create a New Plan

When you create a new Plan, you can define everything about your build process, including what gets built, how the Plan's Build is triggered and what Jobs are executed.

On the 'Create a New Plan' page, complete the options in the following sections to create your Plan:

1. Start the Create Plan wizard as described on Creating a Plan and click 'Create a New Plan'.
2. In the Plan Details section, enter or select the relevant items for the following fields:
   - Project — Select an existing project to which this Plan will belong. If this field is not available (because no projects have yet been defined on your Bamboo server) or you select 'New Project', complete the following fields:
     - Project Name — Enter a name for a new or select an existing project. The plan which you create on this page will be associated with this project.
     - Project Key — Enter a unique key for this project or select an existing project key.
   - Plan Name — Enter a name for the new Plan.
   - Plan Key — Enter a unique key for the new Plan.
   - Plan Description — Enter a short sentence to describe this Plan.
3. In the Source Repository section, specify a source repository configuration that will initially be used for building this Plan's Default Job and also defines the default source repository (or Plan's repository), which Jobs in this Plan can inherit.
   For details, refer to the following source repository configuration procedures, since many of these fields and options change depending on the Repository you select:
   - Bitbucket - Mercurial
   - CVS
   - Git
   - GitHub
   - Mercurial
   - Perforce
   - Subversion
4. In the Build Strategy section, select or enter the relevant items for the following fields:
   - Build Strategy — Choose one of the Build Strategy options (listed below), which will be used for triggering the execution of this plan.
   - You may need to configure other fields specific to your chosen build strategy.
5. Click 'Configure Tasks' to go to the second step of creating your Plan.

Step 2. Configure Tasks for the New Plan

When a new Plan is created, a Default Job is also created as part of the Plan. Before you create your Plan, you need to configure the Tasks for the Default Job. You can add more Tasks to the Default Job after the Plan is created, as well as create new Jobs.

1. On the 'Configure Tasks' screen, click 'Add Task' to add a task to the new Plan.
2. Click the desired Task type in the 'Task Types' window.
3. Fill out the details for the Task. The fields and options will be different depending on the executable that you chose in the Task Types window. See the following documentation for specific instructions on each executable:
   - Configuring a Builder Task
   - Configuring a Test Task
   - Using Global, Plan or Build-specific Variables
4. In the Enable this plan section, choose whether or not to enable this Plan. Enabling the Plan instructs Bamboo to commence
executing builds of the Plan based on the Plan's Build Strategy (defined above). To enable this Plan, select the 'Yes please!' check box.

5. Click the 'Create' button to create the Plan and the Plan's Plan Summary page will be displayed.

   When you next return to the Dashboard (by clicking 'Home' in the top left of the page), your new Plan (and new Project, if applicable) will be displayed in the 'All Projects' list. Bamboo will also automatically run an initial build for your new Plan, unless you have opted for the Manual & dependent builds only Build Strategy.

If you wish to configure more Plan options, please refer to Editing a Plan.

---

### Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>The repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

**Notes**

- You can configure Bamboo to automatically start an initial build for a new Plan, even if you have selected the Manual & dependent builds only Build Strategy. To do so, add the `fire.initial.build.for.manual.strategy` to your bamboo.cfg.xml file as described in Configuring System Properties.

**Related Topics**

- Creating a Plan
- Cloning an Existing Plan
**Importing a Plan from Maven 2**

The instructions on this page describe how to importing a Plan from a Maven 2 project.

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

---

**On this page:**
- Import a Maven 2 Project
- Appendix - Build Strategies
- Notes

---

**Import a Maven 2 Project**

If you have the details for a Plan set up in your Maven 2 project, Bamboo can check out and parse the pom.xml from this project and create a new Plan using the details from it. This feature _will not work_ with Maven 1 projects.

*To create a new Plan from a Maven 2 pom.xml file:*

1. Start the Create Plan wizard as described on Creating a Plan and click 'Import a Maven 2 Project'.
2. Specify a source repository configuration that will initially be used for building this Plan's Default Job and also defines the 'default source repository' (or Plan's repository), which Jobs in this Plan can inherit. For details, refer to the following source repository configuration procedures (from step 4), since many of these fields and options change depending on the 'Repository' you select:
   - CVS
   - Subversion
   - Perforce
   - Mercurial

   If you specify a CVS or Subversion source code management system, then the location you configure in the respective CVS Root or Repository URL fields should contain your Maven 2 project's pom.xml file.
3. Click the 'Import' button to display the Confirm Plan Details page.
4. On the Confirm Plan Details page, some of the fields should already have been populated with information from your pom.xml file. Confirm the following (making any necessary modifications):
   - Plan Details in the top section of the page — for more information about these fields, refer to the Plan Details section in the Create a New Plan procedure above.
   - Source Repository — for more information about these fields, refer to step 1 of this procedure.
5. In the Enable this plan section, choose whether or not to enable this Plan. Enabling the Plan instructs Bamboo to commence executing builds of the Plan based on the Plan's Build Strategy (defined above). To enable this Plan, select the 'Yes please!' check box.
6. Click the 'Confirm' button to save your Plan configuration so far and the Builder tab of the Plan's Default Job is displayed.
7. Select the type of 'Builder' that will be used for building this Default Job.
   
   In practice, you should leave the Builder option set to your 'Maven 2' builder and configure the options on this page as described in Bamboo's Maven configuration documentation. You can select another builder here but this would only be useful if your Maven 2 project's pom.xml depended on a non-Maven 2 builder. If you do need to specify a non-Maven 2 builder, refer to the Builder Configuration section in the Create a New Plan procedure above.

If you wish to configure more Plan options, please refer to Editing a Plan.
## Appendix - Build Strategies

This table lists Bamboo’s available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will ‘poll’ the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>Repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

### Notes

- Security Manager settings too strict — Please note, during the Maven import process, the project file (pom.xml or other specified by the user) is analysed using the maven-embedder library. The parent POM files will be examined in the local repository or downloaded from network repositories. In order to reuse Maven’s credentials for the network repositories, Bamboo requires internal access to the maven-embedder. If your JVM’s or web container’s SecurityManager settings are too strict, this process will fail and you will see an error similar to the following in your Bamboo logs:

  "Problem while initializing Maven Embedder. Probably Security Manager settings are too strict, refer to http://tomcat.apache.org/tomcat-6.0-doc/security-manager-howto.html"

  If your imported Maven project file uses repositories that require credentials (i.e. not public or local), you will need to adjust the SecurityManager settings for your JVM or web container appropriately for the import to work.

- By default, all Plans created by importing a Maven 2 project use the **Polling the Repository for changes** Build Strategy, below, which you can later change when editing this Plan. You can configure Bamboo to automatically start an initial build for a new Plan,
even if you have selected the Manual & dependent builds only Build Strategy. To do so, add the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.

Related Topics
Creating a Plan
Creating a New Plan
Cloning an Existing Plan

Editing a Plan

A Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contains a single ‘Default Job’ in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the ‘default source repository’);
- define how the Plan’s build is triggered;
- who will be notified of the Job’s build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

On this page:
- Editing a Plan
- Notes

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 to display the Plan’s Configuration pages (see below).
4. Click the appropriate tab to edit that aspect of your Plan:
   - ‘Plan Details’ — see Renaming a Plan, its Project or changing the Plan’s Description.
   - ‘Repository’ — see Specifying the Source Repository for a Plan.
   - ‘Stages’ — create new Stages and Jobs, rearrange the order of Stages and move Jobs to different Stages. See Configuring a Stage.
   - ‘Dependencies’ — see Setting up Build Dependencies.
   - ‘Permissions’ — see Configuring a Plan’s Permissions.
   - ‘Notifications’ — see Configuring a Plan’s Notifications.
   - ‘Variables’ — see Defining Plan Variables.
   - ‘Miscellaneous’ — see Configuring Expiry of a Plan’s Job Build Results.

Screenshot above: Plan’s Configuration Pages

Notes

Related Topics
Creating a Plan
Renaming a Plan, its Project or changing the Plan's Description
Specifying the Source Repository for a Plan
Setting up Build Dependencies
Configuring a Plan's Notifications
Configuring Expiry of a Plan's Job Build Results
Configuring a Plan's Permissions

Configuring a Plan's Notifications

Notifications in Bamboo are triggered by a range of events involving a Plan and its Jobs, 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.

For each Plan or Job, you can specify different recipients for each type of event notification. Also be aware that these recipients do not require Bamboo user accounts.

![On this page:
- Adding Notifications for a Plan or Job
- Notification Events
- Removing Notifications from a Plan or Job
- Notes](image)

Adding Notifications for a Plan or Job

Before you begin:

- You must have the 'Edit' permission for a Plan, to add or remove notifications for it.
- You need to configure Bamboo's SMTP email and/or instant messaging capabilities before Bamboo can send notifications. If you have not configured either or both of these, a note will display on the page prompting you to set up the appropriate server(s):
  - To configure an email server for Bamboo, click the 'Add an Email Server' link in the note and enter the email server details in the window that displays. See Configuring Bamboo to send SMTP Email for more information.
  - To configure an instant messaging server for Bamboo, click the 'Add an Instant Messaging Server' link in the note and enter the instant messaging server details in the window that displays. See Configuring Bamboo to use Instant Messaging for more information.

To add notifications for a Plan and/or its Jobs:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Notifications' tab to display the Plan's current Notifications settings.
3. 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 type of 'Event' you want to be notified about. Refer to the list of events (above) for details.
   - Specify the 'Recipient':
     - 'User' — Type the username of the appropriate Bamboo user; or click the following icon to select from a list of users:
     - 'Group' — Type the name of the appropriate Bamboo group(s).
     - '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.
     - 'IM Address' — This is useful if you need to send Instant Messenger (IM) notifications to a person who is not a Bamboo user. Type the appropriate IM address.
     - If you specify a broadcast address (eg. 'project-x@broadcast.chat.mycompany.com'), Bamboo will not know the context of related IM responses.
     - 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.
     - '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.
4. Click the 'Add' button.
5. Repeat steps 5 and 6 until you have added all the build notifications that you wish to enable for this plan.
6. Click the 'Save' button to save your changes.
### Notification Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plans</strong></td>
<td></td>
</tr>
<tr>
<td>All Builds Completed</td>
<td>Bamboo will send a notification whenever the Plan build finishes, regardless of the Plan build’s result. This notification is recommended for any Plans whose latest build activity is critical for people to be informed about. ✔️ This is a good Plan-based notification to use if you are new to Bamboo. You can change it to a less obtrusive notification option as you become more confident with continuous integration and Bamboo’s build processes.</td>
</tr>
<tr>
<td>Failed Builds And First Successful</td>
<td>Bamboo will send a notification whenever: - a build of this Plan fails. - the Plan is ‘fixed’ (that is, the Plan’s latest build is successful and the previous Plan build failed). ✔️ This notification option is generally suitable for the majority of Plans.</td>
</tr>
<tr>
<td>Change of Build Status</td>
<td>Bamboo will send a notification only when there has been a change in status of the Plan’s build activity over consecutive Plan builds — for example, only whenever a Plan’s latest build changes from successful to failed or vice versa (i.e. ‘fixed’). ✔️ This notification option is less obtrusive than the other Plan notifications mentioned above.</td>
</tr>
<tr>
<td>After X Failed Builds</td>
<td>This notification allows you to specify the ‘Number Of Failures’ (i.e. number of failed builds of this Plan), after which Bamboo will send a notification. ✔️ This notification option minimises the number of messages sent by Bamboo if the Plan’s builds fail on a frequent basis. You can also use this event to escalate Plan build problems, for example, to notify a manager when a Plan build fails five times.</td>
</tr>
<tr>
<td>Comment Added</td>
<td>Bamboo will send a notification whenever a comment is posted against a Plan build result. The email notification will contain all comments against the Plan build, whereas IM notifications will only contain the comment that triggered this notification event. ✔️ This notification can help improve collaboration between team members. Be aware that you will not receive notifications for any comments which you post yourself.</td>
</tr>
<tr>
<td><strong>Jobs</strong></td>
<td></td>
</tr>
<tr>
<td>All Jobs Completed</td>
<td>Bamboo will send a notification whenever a Job build of the Plan finishes, regardless of the Job build’s result. This notification is recommended if the latest build activity of all Jobs in this Plan are critical for people to be informed about. ✔️ This is a good Job-based notification to use if you are new to Bamboo. You can change it to a less obtrusive notification option as you become more confident with continuous integration and Bamboo’s build processes.</td>
</tr>
<tr>
<td>Failed Jobs And First Successful</td>
<td>Bamboo will send a notification whenever: - a build of this Job fails. - the Job is ‘fixed’ (that is, the Job’s latest build is successful and the previous Job build failed).</td>
</tr>
</tbody>
</table>
| 'Change of Job Status' | Bamboo will send a notification only when there has been a change in build activity status of the Jobs within this Plan over consecutive Plan builds — for example, only whenever the latest build of any Job in this Plan changes from successful to failed or vice versa (i.e. 'fixed').

- This notification option is less obtrusive than the other Job notifications mentioned above.

| 'First Failed Job For Plan' | If multiple Jobs fail in a Plan, Bamboo will only send a notification for the first failing Job detected by the Bamboo system.

- This is a less obtrusive notification option that informs about a failing Job (and hence, Plan) in the shortest possible time.

| 'Job Error' | Bamboo will send a notification whenever an error occurs in one of the Plan's Job build processes (i.e. the activities that Bamboo performs to run a Job build). This event is not related to failures of the actual build itself (see the 'Failed Jobs And First Successful' and 'Failed Builds And First Successful' events above). For example, a notification will be sent if Bamboo encounters an error when connecting to the repository, or detecting changes.

| 'Job Hung' | Bamboo will send a notification whenever it determines that one of the Plan's Job builds has hung, according to the hung Job build criteria (read more about configuring your hung Job build settings).

- Use this notification to ensure that the relevant people are informed when a Job build becomes unresponsive.

| 'Job Queue Timeout' | Bamboo will send a notification whenever one of the Plan's Job builds has been waiting in the queue for longer than the build queue timeout criteria (read more about configuring your Job's Build Queue Timeout settings).

- Use this notification to ensure that the relevant people are informed when a Job build is stuck in the build queue for too long.

| 'Job Queued Without Capable Agents' | Bamboo will send a notification whenever one of the Plan's Job builds is queued and there are no agents capable of building it.

- Use this notification to ensure that people are notified when changes to agents adversely affect your Job's builds.

### Defining Plan Variables

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

**Plan variables** are one type of variable that is available to you. A Plan variable is defined for one specific Plan, and has the same value every time that Plan is built. If you want to define a variable across all Plans rather than a single Plan, define a global variable as described in Defining Global Variables.

Plan variables can be accessed by using `{$bamboo.varName}`. Plan variables can also be overridden at runtime when running a manual build. For more information, see Triggering a Plan Build Manually.

---

**On this page:**

- Defining a Plan Variable
- Notes

---

### Defining a Plan Variable

Before you begin:

- Note, Plan variables override global variables with the same name.

To define a Plan Variable,

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Variables' tab.
3. Add, update or delete the Plan variables, as desired:
   - Click 'Add' to add new variables once you have entered the key and value for it.
   - Updates to existing rows will be saved as you move between cells in the table.
   - Click the trash can to delete a variable.
Granting Plan Permissions

To grant Plan permissions:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Permissions' tab to display the Plan's current Permissions settings.
3. Grant who has permission to access this Plan as described:
   a. In the 'Grant permission to' list at the bottom of the screen, select 'User'.
   b. Type the username into the box, or click the icon to select from a list of users in the resulting Select User window.
   c. 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').
   d. Click the 'Add' button. The user will be added to the list of users on the 'Permissions' tab.
   e. Select the check boxes of each permission you wish to grant to this user.
   f. To grant Plan permissions to a group:
      a. In the 'Grant permission to' list at the bottom of the screen, select 'Group'.
      b. Type the group name into the box.
      c. 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').
      d. Click the 'Add' button. The group will be added to the list of groups on the 'Permissions' tab.
      e. Select the check boxes of each permission that you wish to grant to this group.
   g. To grant Plan permissions to all Bamboo users:
      a. Locate 'Logged in users' (under 'Other') and select the check boxes of each permission that you wish to grant to all Bamboo users.
   h. To grant Plan permissions to anonymous users:
      a. Locate 'Anonymous users' (under 'Other') and select the check boxes of each permission that you wish to grant to
Revoking Plan Permissions

To revoke Plan permissions:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the ‘Permissions’ tab (see screenshot below).
3. Locate the relevant User / Group / Other (group of) users.
4. Clear the check box of each permission that you wish to revoke from these (group of) users.
   - If you clear all permissions for a user or group, that user or group will be removed from the ‘Permissions’ page.
5. Click the ‘Save’ button.

**Screenshot above: Plan Permissions**

Plan Permissions Summary

Please note, anyone with the ‘Admin’ global permission automatically has all plan permissions for every plan. 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  
  - People who don’t have the ‘View’ permission will not know that the plan exists. | - 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: | - a particular user - a particular group |
| | • edit this plan's permissions. | |
| | • delete this plan's build results and working files. | |

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.

Notes
Related Topics
Editing a Plan
Granting Plan Permissions to Users or Groups

Configuring Expiry of a Plan’s Job Build Results

By enabling **build expiry** for a Plan, you can choose what Job build result data* will be kept for a Plan and for how long this data will be kept (e.g. for reporting purposes), before Bamboo automatically deletes it.

* 'Job build result data' refers to all Job builds of a given Plan’s build.

If you disable build expiry, your Plan's Job build result data will never be automatically deleted from your Bamboo server.

You can enable/disable build expiry for:

- **all plans** (see Configuring Global Expiry of Job Build Results). This is generally the easiest way to manage build expiry in Bamboo.
- Your settings will apply to all Plans that do not have individual build 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 Job build result data.

You can also delete the results of a Plan build manually — see Deleting the Results of a Plan Build.

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 refer to Configuring Global Expiry of Job Build Results.

On this page:
- Enabling the Expiry of Job Build Results
- Disabling the Expiry of Job Build Results
- Notes

Enabling the Expiry of Job Build Results

Before you begin:

- Ensure that you back up your build results data before its expiry date is reached.

To enable and configure the expiry of Job build result data for a Plan:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Miscellaneous' tab to display the Plan's current build expiry settings.
3. Select the 'Override global build expiry configuration' check box. The following fields will be displayed:
   a. 'What should be expired?' — select:
      - 'Build result' — if you want to delete all build results data (including artifacts and build logs). If you choose this option, the following two options are automatically selected.
      - 'Artifacts' — if you want to selectively delete all user-defined artifacts of Jobs in this Plan but keep all other Job build result data (such as build logs).
      - 'Build Logs' — if you want to selectively delete all build logs of Jobs in this Plan but keep all other Job build result data (such as artifacts).
   b. In the Expiry criteria section, use one of the follow three criteria to determine how much Job build result data to keep, by modifying the 'Expiry period' and 'Minimum builds to keep' fields accordingly:
      - In this section, 'Job build results' could refer to artifacts, build logs or both (depending on what you specified in the previous substep).
      - To keep all Job build results of this Plan up to a certain age,
      - 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 Job build results of this Plan. E.g. specify '24 months' to keep all Job build results for the last two years.
      - In the 'Minimum builds to keep' field, specify '0'.

98
To keep a specified (or minimum) number of Job build results of this Plan,
- In the 'Expiry period' field, specify '0'.
- In the 'Minimum builds to keep' field, specify the number of Job build results of this Plan you want to keep. E.g. specify '50' to keep the latest 50 Job build results.
- To keep all Job build results of this Plan up to a certain age and a minimum number of these Job build results,
  - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your Job build results of this Plan. E.g. specify '24 months' to keep all Job build results for the last two years.
  - In the 'Minimum builds to keep' field, specify the number of Job build results of this Plan you want to keep. E.g. specify '50' to keep the latest 50 Job build results.
  (Hence, even if all Job builds of this Plan are over two years old, the last 50 Job build results will not be deleted.)
- If you wish to keep Job build results with particular labels, select the 'Don't expire builds with certain labels' check box. The 'Labels to keep' field will be displayed:
  - In the 'Labels to keep' field, specify any labels which match the labels applied to Job builds you want to keep. (If you want to specify more than one label, use spaces to separate them.) For any label(s) specified, all Job builds of this Plan which have a matching label will never be deleted, regardless of the Minimum builds to keep and Labels to keep settings.

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 Configuring Automatic Labelling of Job Build Results in the Bamboo Administrator's Guide.

4. Click the 'Save' button to save your changes.

Disabling the Expiry of Job Build Results

To disable expiry of the Job build result data for a Plan:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Miscellaneous' tab to display the Plan's current build expiry settings.
3. Select the 'Override global build expiry configuration' check box.
4. 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.
5. Click the 'Save' button to save your changes.
### Renaming a Plan, its Project or changing the Plan's Description

To rename a Plan, its Project or change the Plan's description:

1. Navigate to the configuration for the desired Plan, as described in Editing a Plan.
2. Click the 'Plan Details' tab to display the Plan Details.
3. Modify the following fields as required:
   - 'Project Name' — enter a new name for the Plan's Project.
   - 'Plan Name' — enter a new name for the Plan.
   - 'Plan Description' — enter a new description for the Plan.
4. If you want, you can enable or disable this Plan by selecting or clearing the 'Plan Enabled' check box, respectively.
5. Click the 'Save' button.

A Plan's Project Key and Plan Key are not editable, but can be changed as described in Moving Plans to a Different Project.

### Specifying the Source Repository for a Plan

Screenshot above: Edit Plan Details
When creating a new Plan, you must define a ‘default source repository’. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The ‘default source repository’ is initially used by the Plan’s ‘Default Job’ and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

For details, please refer to the following pages (see Notes below):

- Bitbucket - Mercurial
- CVS
- Git
- GitHub
- Mercurial
- Perforce
- Subversion

### Navigating to the Source Repository Settings for a Plan

To navigate to the source repository settings 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 its icon. The Plan’s Configuration pages will be displayed.
4. Click the ‘Repository’ tab to display the repository settings for your Plan (see screenshot below).
Notes

- A number of source repositories are supported 'out of the box', as described on the Supported Platforms page. If you need to use a type of repository that is not supported, 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.

Bitbucket - Mercurial

The instructions on this page describe how to configure a Bitbucket Mercurial source repository for either a Plan or a Job.

When creating a new Plan, you must define a ‘default source repository’. This includes specifying the source repository settings (including the repository's location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The ‘default source repository’ is initially used by the Plan's 'Default Job' and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).
Configuring a Bitbucket Mercurial Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

To configure a Bitbucket Mercurial source repository:

1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
2. Enter the details for the BitBucket repository as described below:
   - Use Plan’s Repository (Only available when configuring a Job) — Select this option if you wish to use the ‘default source repository’ of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.
   - Repository — Select ‘Bitbucket’.
   - Username — Type the username required to access the Bitbucket account. It will be also used to access chosen repository.
   - Password — Type the password you want to access the Bitbucket account. It will be also used to access chosen repository.
   - Repository — Select ‘My Repositories’ to let Bamboo look up your repositories automatically, or select ‘URL’ to manually specify the repository URL.
     - If ‘My Repositories’ is selected, click ‘Load Repositories’ to load the repositories for the username entered above, then select the ‘Repository’ and ‘Branch’ that you want to check out code from.
     - If ‘URL’ is selected, enter the ‘Repository URL’ — This is the full path to your Bitbucket repository (e.g. http://bitbucket.org/sinbad/ogre). If you want to choose a branch to work on, other than the default branch, click ‘Load Branches’ and select the desired branch.
   - Command timeout (Optional) — Type the number of minutes bamboo should wait for Bitbucket commands to finish. This is useful to cut hung Bitbucket processes. On the slower networks, you may consider increasing the default timeout to allow Bamboo to make an initial clone of mercurial repository.
   - ‘Verbose logs’ — Turns on --verbose and --debug options in hg commands and pass the output to build logs. Use that option if you encounter problems in Bamboo Mercurial behaviour.
3. Common Repository Configuration (Only available when configuring an existing Plan/Job)
   - Force Clean Build: (Optional) — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
   - Clean working directory after each build: (Optional) — You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.
   - Include/Exclude Files: (Only available when configuring a Plan) — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than ‘None’, the following field will appear:
     - File Pattern: — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.
   - Web Repository: — Select the type of web repository (‘None’, ‘Generic Web Repository’, ‘Mercurial Web Repository’, ‘FishEye’) to be associated with this build. You will be able to view code changes related to your build via the build results. Only a subset of Web Repository options are available for your chosen repository type.
     - If ‘Generic Web Repository’ is selected:
       - Web Repository URL — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the ‘Code Changes’ section of a build result.
       - Web Repository Module — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.
     - If ‘Mercurial Web Repository’ is selected:
       - Mercurial Web Repository Viewer Scheme — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial’s own default web server (Default Web Repository Scheme (hgserve)).
     - If ‘FishEye’ is selected:
       - FishEye URL — The URL of your FishEye repository (e.g. https://atlas-eye.atlassian.com/).
       - Repository Name — The name of your FishEye repository (e.g. ‘Bamboo’). This is effectively the alias for your repository path.
       - Repository Path — The path for your FishEye repository (e.g. ‘/atlassian/bamboo’).
How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/.

If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

- **Build Strategy** *(Only available when configuring a Plan)* — Choose one of the build strategy options (listed below), which will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.

  - You may need to configure other options specific to your chosen build strategy.

  - *If you select Manual & dependent builds only when creating a new Plan, an initial build will not automatically be run. You can force an initial build to be executed automatically by adding the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.*

4. Click the 'Save' button to save your changes.
Source Repository

Source Control: Bitbucket

Mercurial is currently set to use hg as the Mercurial Executable on the Bamboo Server. To run a Mercurial build remotely please ensure that the Mercurial Executable capability is defined for the agents.

Username: alui
The Bitbucket user required to access the repositories.

Password: ------------------------------
The password required by the Bitbucket username.

Repository: My Repositories
Select the repository you want to use for your Plan.

Branch: default
Choose a branch you want to checkout your code from.

Command timeout (minutes): 180
Specifies how many minutes are given for hg commands to finish. Default is 180 (3 hours).

Verbose logs
Outputs more verbose logs from hg commands.

Common repository configuration

- Force Clean Build
  Removes the source directory and checks it out again prior to each build. This may significantly increase build times.

- Clean working directory after each build
  Removes the source directory after each build. This may increase build times but save disk space.

- Enable Quiet Period
  Quiet period allows you to delay building after a single commit is detected, aggregating multiple commits per build.

Include / Exclude Files: None
Customise what files Bamboo uses to detect changes.

Web Repository: None
Select, if any, the browsable web repository associated with this build.

Build Strategy: Polling the Repository for changes
How should Bamboo trigger Builds for this Plan? (Dependent Builds are automatically triggered)

Polling Strategy: Periodically
- Scheduled
Please select a polling strategy.

Schedule: The 1st day of every month at 12:00 am

Trigger Conditions

- Only run Build if other Plans are currently passing

Save Cancel
This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will ‘poll’ the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>The repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Notes

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

  You will not be able to create Plans/Jobs that use a Bitbucket Mercurial repository without specifying the shared local Mercurial capability first. Read more about configuring a Mercurial capability.

Related Topics

Specifying the Source Repository for a Plan
Specifying the Source Repository for a Job

CVS

The instructions on this page describe how to configure a CVS source repository for either a Plan or a Job.

When creating a new Plan, you must define a ‘default source repository’. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The ‘default source repository’ is initially used by the Plan’s ‘Default Job’ and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

On this page:

- Configuring a CVS Source Repository
- Appendix - Build Strategies
- Notes

Configuring a CVS Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

To configure a CVS source repository:

If you are creating a new Plan or new Job, or have come from the Editing a Plan or Editing a Job topics, start at step 4.
1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
2. Enter the details for the CVS repository as described below:

   - **'Use Plan’s Repository'** *(Only available when configuring a Job)* — Select this option if you wish to use the 'default source repository' of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.

   - **'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 pservers, ext (ssh) and local repository access methods. Note that you can use global variables in this field (see Using Global, Plan or Build-specific Variables).
   - If you are importing a Maven 2 Project, this location should contain your project's pom.xml file.
   - **'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'** *(Only available after first saving the Plan/Job with a password)* — 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'** *(Only available after first saving the Plan/Job with a passphrase)* — Select this check box if you want to change the password for your SSH private key.

   - **'Quiet Period'** *(Only available when configuring an existing Plan/Job)* — 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:
     - a. Create a module (e.g. allbuilds).
     - b. Define an ampersand module with the same name. (The ampersand module can be empty.)
     - c. 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/Tag Name'** — Type the relevant branch name or tag. Note that you can use global variables in this field (see Using Global, Plan or Build-specific Variables).

3. **Common Repository Configuration** *(Only available when configuring an existing Plan/Job)*

   - **'Force Clean Build:'** *(Optional)* — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
   - **'Clean working directory after each build:'** *(Optional)* — You can force Bamboo to remove the source directory after the Plan/Job is completed by building by selecting this option. Please note that this may increase build times but saves on disk space.
   - **'Include/Exclude Files:'** *(Only available when configuring a Plan)* — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', the following field will appear:
     - **'File Pattern:'** — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.
   - **'Web Repository:'** — Select the type of web repository ('None', 'Generic Web Repository', 'Mercurial Web Repository', 'FishEye') to be associated with this build. You will be able to view code changes related to your build via the build results.
   - Only a subset of Web Repository options are available for your chosen repository type.
   - If **'Generic Web Repository'** is selected:
     - **'Web Repository URL'** — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
     - **'Web Repository Module'** — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.
   - If **'Mercurial Web Repository'** is selected:
     - **'Mercurial Web Repository Viewer Scheme'** — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial’s own default web server (Default Web Repository Scheme (hgserve)).
   - If **'FishEye'** is selected:
     - **'FishEye URL'** — The URL of your FishEye repository (e.g. 'https://atlaseye.atlassian.com/').
     - **'Repository Name'** — The name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
     - **'Repository Path'** — The path for your FishEye repository (e.g. '/atllassian/bamboo/').
How do I determine my Repository Path?

If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/.

If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

4. Click the 'Save' button to save your changes.
Bamboo 3.1 Documentation

109

Screenshot above: Source Repository — CVS

Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
</tbody>
</table>
The repository triggers the build when changes are committed

**Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.**

**This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.**

Repository Triggers the Build when Changes are Committed

Cron Based Scheduling

**Bamboo will trigger a build of this plan based on a Cron expression.**

**This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.**

Cron Based Scheduling

Single daily build

**Bamboo will trigger a build of this plan once per day at a specified time.**

**This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.**

Single Daily Build

Manual & dependent builds only

**Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.**

**This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.**

Triggering a Plan Build Manually

### Notes

**Related Topics**

- Specifying the Source Repository for a Plan
- Specifying the Source Repository for a Job
- Configuring Source Code Management Triggers for CVS

**Configuring Source Code Management Triggers for CVS**

This page provides instructions on how to configure CVS to send message events that trigger the execution of Bamboo plans.

You only need to configure CVS to send these message events if **'The repository triggers the build when changes are committed' build strategy** has been specified for one or more of your Bamboo plans.

On this page:

- Configuring CVS to Trigger a Build
- Notes

#### Configuring CVS to Trigger a Build

This section explains how to configure CVS to trigger a build when the repository is changed. This involves installing two scripts:

1. A pre-commit trigger keeps track of the last directory to be processed, so we know when the commit has completed.
2. A post-commit trigger that waits until it has processed the last directory of the commit before instructing the Bamboo server to execute the relevant plan(s).

⚠️ The following commands and script files assume that your CVS server runs on a UNIX- or Linux-based operating system. If your CVS server runs on any other operating system, then you will need to modify the script files and if necessary, the commands below to suit that operating system.

**Step 1. Checking out the CVSROOT**

First check out your repository's CVSROOT directory into a temporary directory:

```bash
cvs -d cvsroot-to-your-repository checkout CVSROOT
```

where:

- `cvsroot-to-your-repository` is the root directory pathname of the CVS repository.

⚠️ Using `-d cvsroot-to-your-repository` overrides the any `$CVSROOT` environment variable setting.

The following files should be checked out:
Step 2. Install the Pre-Commit Trigger

Add a line like the following example’s to the CVSROOT/commitinfo pre-commit trigger file. The CVSROOT/commitinfo file contains the list of programs to run whenever a file is about to be committed to the repository.

```
^Moo /path-to-your-bamboo-installation/scripts/cvs-triggers/preCommit.sh
```

where:

- `^Moo` is the regular expression used to identify the name of the module (called Moo) being updated.
- `/path-to-your-bamboo-installation/scripts/cvs-triggers/preCommit.sh` is the Bamboo shell script used to detect the last file of the check in.

⚠️ If your Bamboo installation and CVS server are on different machines, refer to the note below.

Step 3. Install the Post-Commit Trigger

Add a line like the following example’s to the CVSROOT/loginfo post-commit trigger file. The CVSROOT/loginfo file contains the list of programs to run whenever a file has been successfully committed into the repository.

```
^Moo /path-to-your-bamboo-installation/scripts/cvs-triggers/postCommitBuildTrigger.sh %{}
http://bamboo-base-url MOO-KEY
```

where:

- `^Moo` is the regular expression used to identify the name of the module (called Moo) being updated.
- `/path-to-your-bamboo-installation/scripts/cvs-triggers/postCommitBuildTrigger.sh` is the Bamboo shell script to trigger the build.
- `%{}` is how CVS tells the postCommitBuildTrigger.sh script which directory it is committing.
- `MOO-KEY` the key of the Bamboo plan to be executed.

Step 4. Save the changes back to CVS

Commit the changes you made to the CVSROOT/commitinfo and CVSROOT/loginfo files in step 2 and 3, respectively, back to the repository.

```
cvs -d cvsroot-to-your-repository commit
```

where:

- `cvsroot-to-your-repository` is the root directory pathname of the CVS repository.

⚠️ Using `-d cvsroot-to-your-repository` overrides the any `$CVSROOT` environment variable setting.

Step 5. Do a test commit

Conduct a 'test' commit. Bamboo should start building the relevant plan after a few seconds.

The Bamboo log file should contain an entry like this:

```
[INFO] com.atlassian.bamboo.build.UpdateAndBuild - Bamboo build was triggered by remote http call from 127.0.0.1
```
The postCommitBuildTrigger.sh is only triggered when the last file of the commit has been committed.

The preCommit.sh and postCommitBuildTrigger.sh must have sufficient privileges to be executed by the CVS user.

If your Bamboo installation is not running on the same machine as the CVS server, you will also need add the Bamboo preCommit.sh and postCommitBuildTrigger.sh files to the CVSROOT directory and add the names of these files to the end of the checkoutlist file.

Notes

1. **Build Trigger Security** — Bamboo will only accept remote build triggers if the triggers originated from the CVS server(s) identified in the CVS root paths of any Bamboo plans. Requests originating from other CVS servers will be rejected by Bamboo.

Related Topics

CVS
Configuring Source Code Management Triggers for Subversion

Git

The instructions on this page describe how to configure a Git source repository for either a Plan or a Job.

When creating a new Plan, you must define a 'default source repository'. This includes specifying the source repository settings (including the repository's location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The 'default source repository' is initially used by the Plan's 'Default Job' and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

On this page:

- Configuring a Git Source Repository
- Appendix - Build Strategies
- Notes

Configuring a Git Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

To configure a Git source repository:

- If you are creating a new Plan or new Job, or have come from the Editing a Plan or Editing a Job topics, start at step 4.

1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
2. Enter the details for the Git repository as described below:

   - **Use Plan's Repository** *(Only available when configuring a Job)* — Select this option if you wish to use the 'default source repository' of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.

   - **Repository** — select 'Git'.

   - **Repository URL** — Type the full path to your Git repository (e.g. `git://github.org/atlassian/bamboo-git-plugin.git`). Valid URLs are of the form:

     ```
     git://host.xz[:port]/path/to/repo.git
     ssh://[user@]host.xz[:port]/path/to/repo.git
     [user@]host.xz:path/to/repo.git
     http[s]://host.xz[:port]/path/to/repo.git
     /path/to/repo.git
     file:///path/to/repo.git
     ```

     (for further references visit [Git documentation](https://git-scm.com/docs))

   - **Branch** *(Optional)* — Type the name of the relevant branch (or tag) you want to work on. Leave empty to work on master branch.

   - **Authentication Type** — Select the authentication type you want to use to access the Git repository.

     - If you select 'None', Bamboo will not authenticate at all.
     - If you select 'Username/password' (use that authentication type for http/https and SSH hosted repositories), the following fields will appear:

       - **Username** *(Optional)* — Type the username (if any) required to authenticate to the repository hosted with http(s) or SSH protocol.
       - **Password** *(Optional)* — Type the password (if any) required to authenticate to the repository hosted with http(s) or SSH protocol.
http(s) or SSH protocol.

- **'Change Password'** *(Only available after first saving the Plan/Job with a password)* — Select this check box if you want to change the password that is used to access the repository.
- If you select **'SSH private key'** *(use that authentication type for repositories accessible through ssh)*, the following fields will appear:
  - **'SSH Key'** — Choose an SSH private key from your hard drive.
  - **'Change SSH Key'** *(Only available after first saving the Plan/Job with an SSH private key)* — Select this check box if you want to change your SSH private key.
  - **'SSH Passphrase'** *(Optional)* — Type the passphrase for your SSH private key.
  - **'Change Passphrase'** *(Only available after first saving the Plan/Job with a passphrase)* — Select this check box if you want to change the passphrase for your SSH private key.
- **'Use shallow clones'** — Select this setting to allow Bamboo to perform shallow clones (i.e. history truncated to a specified number of revisions). This should increase the speed of the initial code checkpoints, however if your build depends on the full repository history, we recommend that you do not use this option. Shallow clones are enabled by default.
- **'Location of POM file'** *(Only available when importing a Maven 2 project)* — Type the path to your project’s pom.xml file which is relative to the root of your Git repository (e.g. '/atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml').

3. **Common Repository Configuration** *(Only available when configuring an existing Plan/Job)*

- **'Force Clean Build'** *(Optional)* — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being selected by this option. Please note that this will greatly increase the time it takes to complete a build.
- **'Clean working directory after each build:'** *(Optional)* — You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.
- **'Include/Exclude Files:'** *(Only available when configuring a Plan)* — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than **'None'**, the following field will appear:
  - **'File Pattern:'** — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.
- **'Web Repository:'** — Select the type of web repository (**'None'**, **'Generic Web Repository'**, **'Mercurial Web Repository'**, **'FishEye'**) to be associated with this build. You will be able to view code changes related to your build via the build results.

   - Only a subset of **Web Repository** options are selected for your chosen repository type.
   - If **'Generic Web Repository'** is selected:
     - **'Web Repository URL'** — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
     - **'Web Repository Module'** — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.
   - If **'Mercurial Web Repository'** is selected:
     - **'Mercurial Web Repository Viewer Scheme'** — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial’s own default web server (**'Default Web Repository Scheme (hgserve)'**).
   - If **'FishEye'** is selected:
     - **'FishEye URL'** — The URL of your FishEye repository (e.g. 'https://atlassian.atlassian.com/').
     - **'Repository Name'** — The name of your FishEye repository (e.g. 'bamboo'). This is effectively the alias for your repository path.
     - **'Repository Path'** — The path for your FishEye repository (e.g. '/atlassian/bamboo/').

   ![How do I determine my Repository Path?](https://example.com)

   If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/.

   If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

- **'Build Strategy'** *(Only available when configuring a Plan)* — Choose one of the build strategy options (listed below), which will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.

  - You may need to configure other options specific to your chosen build strategy.
  - If you select **Manual & dependent builds only** when creating a new Plan, an initial build will not automatically be run. You can force an initial build to be executed automatically by adding the `fire.initial.build.for.manual.strategy` to your `bamboo.cfg.xml` file as described in Configuring System Properties.

4. Click the **'Save'** button to save your changes.
Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
</tbody>
</table>
Bamboo 3.1 Documentation

<table>
<thead>
<tr>
<th>The repository triggers the build when changes are committed</th>
<th>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</th>
<th>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</th>
<th>Repository Triggers the Build when Changes are Committed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Notes
Related Topics

**Specifying the Source Repository for a Plan**

**Specifying the Source Repository for a Job**

**GitHub**

The instructions on this page describe how to configure a GitHub source repository for either a Plan or a Job.

When creating a new Plan, you must define a 'default source repository'. This includes specifying the source repository settings (including the repository's location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The 'default source repository' is initially used by the Plan's 'Default Job' and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

On this page:

- Configuring a GitHub Source Repository
- Appendix - Build Strategies
- Notes

### Configuring a GitHub Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

**To configure a GitHub source repository:**

1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
2. Enter the details for the GitHub repository as described below:
   - **'Use Plan's Repository'** (Only available when configuring a Job) — Select this option if you wish to use the 'default source repository' of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.
   - **'Repository'** — Select 'GitHub'.
   - **'Username'** — Type the username required to access the GitHub account. It will be also used to access chosen repository.
   - **'Password'** — Type the password required to access the GitHub account. It will be also used to access chosen repository.
   - **Click 'Load Repositories'** to load the repositories for the username entered above, then complete the following:
     - **Select the 'Repository' and 'Branch'** that you want to check out code from.
     - **'Use shallow clones'** — Select this setting to allow Bamboo to perform shallow clones (i.e. history truncated to a specified number of revisions). This should increase the speed of the initial code checkouts, however if your build depends on the full repository history, we recommend that you do not use this option. Shallow clones are enabled by default.
3. **Common Repository Configuration** (Only available when configuring an existing Plan/Job)
• 'Force Clean Build:' (Optional) — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.

• 'Clean working directory after each build:' (Optional) — You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.

• 'Include/Exclude Files:' (Only available when configuring a Plan) — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', the following field will appear:
  • 'File Pattern:' — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.

• 'Web Repository:' — Select the type of web repository ('None', 'Generic Web Repository', 'Mercurial Web Repository', 'FishEye') to be associated with this build. You will be able to view code changes related to your build via the build results.

Only a subset of Web Repository options are available for your chosen repository type.

• If 'Generic Web Repository' is selected:
  • 'Web Repository URL' — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
  • 'Web Repository Module' — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.

• If 'Mercurial Web Repository' is selected:
  • 'Mercurial Web Repository Viewer Scheme' — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial's own default web server (Default Web Repository Scheme (hgserve)).

• If 'FishEye' is selected:
  • 'FishEye URL' — The URL of your FishEye repository (e.g. 'https://atlasEye.atlassian.com/').
  • 'Repository Name' — The name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
  • 'Repository Path' — The path for your FishEye repository (e.g. '/atlassian/bamboo/').

How do I determine my Repository Path?
If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/.
If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

• 'Build Strategy' (Only available when configuring a Plan) — Choose one of the build strategy options (listed below), which will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.

⚠️ You may need to configure other options specific to your chosen build strategy.

⚠️ If you select Manual & dependent builds only when creating a new Plan, an initial build will not automatically be run. You can force an initial build to be executed automatically by adding the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.

4. Click the 'Save' button to save your changes.
### Build Strategies

This table lists Bamboo's available **build strategies** that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
</table>

**Screenshot above: Source Repository — GitHub**

### Appendix - Build Strategies

This table lists Bamboo's available **build strategies** that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
</table>

**Screenshot above: Source Repository — GitHub**
Polling the Repository for changes

<table>
<thead>
<tr>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered. This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td></td>
</tr>
<tr>
<td>Repository Triggers the Build when Changes are Committed</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
</tbody>
</table>

The repository triggers the build when changes are committed

<table>
<thead>
<tr>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan. This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td></td>
</tr>
<tr>
<td>Repository Triggers the Build when Changes are Committed</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
</tbody>
</table>

Cron Based Scheduling

<table>
<thead>
<tr>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo will trigger a build of this plan based on a Cron expression. This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
</tbody>
</table>

Single daily build

<table>
<thead>
<tr>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo will trigger a build of this plan once per day at a specified time. This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
</tbody>
</table>

Manual & dependent builds only

<table>
<thead>
<tr>
<th>Description</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency. This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Notes

Related Topics

Specifying the Source Repository for a Plan
Specifying the Source Repository for a Job
Git

Mercurial

The instructions on this page describe how to configure a Mercurial source repository for either a Plan or a Job.

When creating a new Plan, you must define a 'default source repository'. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The 'default source repository' is initially used by the Plan's 'Default Job' and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

On this page:

- Configuring a Mercurial Source Repository
- Appendix - Build Strategies
- Notes

Configuring a Mercurial Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

To configure a Mercurial source repository:

1. If you are creating a new Plan or new Job, or have come from the Editing a Plan or Editing a Job topics, start at step 4.
2. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
3. Enter the details for the Mercurial repository as described below:
   - 'Use Plan’s Repository’ (Only available when configuring a Job) — Select this option if you wish to use the ‘default source repository’ of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.
   - 'Repository’ — select ‘Mercurial’.
   - 'Repository URL’ — Type the full path to your Mercurial repository (e.g. 'http://bitbucket.org/sinbad/ogre'). Valid URLs are of the form:
     - local/filesystem/path[#revision]
     - file://local/filesystem/path[#revision]
• `Branch` *(Optional)* — Type the name of the relevant branch you want to work on. Leave empty to work on default branch.
• `Username` *(Optional)* — Type the username (if any) required to access the repository.
• `Authentication Type` — Select the authentication type you want to access Mercurial repository:
  • If you select `Password` (use that authentication type for http/https hosted repositories), the following fields will appear:
    • `Password` *(Optional)* — Type the password you want to access the repository.
• If you select `Keyfile with passphrase` (use that authentication type for repositories accessible through ssh), the following fields will appear:
  • `$SSHD Key` — Choose an SSH private key from your hard drive.
  • `Change SSH Key` *(Only available after first saving the Plan/Job with an SSH private key)* — Select this check box if you want to change your SSH private key.
• If you select `Keyfile without passphrase` (this authentication type is similar to above except it is a bit faster), the following fields will appear:
  • `SSH Key` — Choose an SSH private key from your hard drive.
  • `Change SSH Key` *(Only available after first saving the Plan/Job with an SSH private key)* — Select this check box if you want to change your SSH private key.
• If you select `Default Mercurial credentials`, Bamboo will rely on default hg authentication. Use this option, for example, if you had set up the Bamboo server manually with SSH servers defined in `/.ssh/config`, valid SSH identity files, etc.
• `Command timeout` *(Optional)* — Type the number of minutes bamboo should wait for hg commands to finish. This is useful to cut hung Mercurial processes. On the slower networks you may consider increasing default timeout to allow bamboo making initial clone of mercurial repository.
• `Location of POM file` *(Only available when importing a Maven 2 project)* — Type the path to your project's `pom.xml` file which is relative to the root of your Mercurial Repository URL *(defined above).*
• `Verbose logs` — Turns on `--verbose` and `--debug` options in hg commands and pass the output to build logs. Use that option if you encounter problems in Bamboo Mercurial behaviour.

3. **Common Repository Configuration** *(Only available when configuring an existing Plan/Job)*

• `Force Clean Build:` *(Optional)* — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
• `Clean working directory after each build:` *(Optional)* — You can force Bamboo to remove the source directory after the Plan/Job is completed by selecting this option. Please note that this may increase build times but saves on disk space.
• `Include/Exclude Files:` *(Only available when configuring a Plan)* — You can specify a particular inclusion or exclusion pattern for file changes to be detected. The regex pattern must match the file path in the repository.
• `Web Repository:` — Select the type of web repository (`None`, `Generic Web Repository`, `Mercurial Web Repository`, `FishEye`) to be associated with this build. You will be able to view code changes related to your build via the build results.

Only a subset of Web Repository options are available for your chosen repository type.
• If `Generic Web Repository` is selected:
  • `Web Repository URL` — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the `Code Changes` section of a build result.
  • `Web Repository Module` — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.
• If `Mercurial Web Repository` is selected:
  • `Mercurial Web Repository Viewer Scheme` — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial's own default web server *(Default Web Repository Scheme (hgserve))*.  
• If `FishEye` is selected:
  • `FishEye URL` — The URL of your FishEye repository *(e.g. `https://atlaseye.atlassian.com/`).*
  • `Repository Name` — The name of your FishEye repository *(e.g. `Bamboo`). This is effectively the alias for your repository path.
  • `Repository Path` — The path for your FishEye repository *(e.g. `//atlassian/bamboo/`).*
How do I determine my Repository Path?
If you have previously run builds with changes from your repository, the easiest way of
determining your repository path is to view the code changes and copy the path from the
start of the path of one of the changed files, up to (but not including) the appropriate root
directory. The root directories for repositories are the ones shown by FishEye when
browsing a repository (e.g. trunk). For example, if a code change listed
/atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would
be /atlassian/bamboo/.
If you have not previously run builds with changes from your repository, you will need to
ask your FishEye administrator for the repository path indexed by FishEye.

‘Build Strategy’ (Only available when configuring a Plan) — Choose one of the build strategy options (listed below), which
will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.

⚠️ You may need to configure other options specific to your chosen build strategy.

⚠️ If you select Manual & dependent builds only when creating a new Plan, an initial build will not automatically be run. You
can force an initial build to be executed automatically by adding the fire.initial.build.for.manual.strategy to
your bamboo.cfg.xml file as described in Configuring System Properties.

4. Click the ‘Save’ button to save your changes.
The Screenshot above shows a form for configuring a Mercurial source repository in Bamboo. The form includes fields for repository URL, branch name, username, password, command timeout, and various options related to common repository configuration such as force clean build, clean working directory, and include/exclude files. Additionally, there are options for web repository, build strategy, and trigger conditions.

Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
</table>

Bamboo 3.1 Documentation
Polling the Repository for changes

Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.

This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.

The repository triggers the build when changes are committed

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.

Cron Based Scheduling

Bamboo will trigger a build of this plan based on a Cron expression.

This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Single daily build

Bamboo will trigger a build of this plan once per day at a specified time.

This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Manual & dependent builds only

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.

This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.

Notes

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

  You will not be able to create Plans/Jobs that use a Mercurial repository without specifying the shared local Mercurial capability first. Read more about configuring a Mercurial capability.

Related Topics

Specifying the Source Repository for a Plan
Specifying the Source Repository for a Job

Upgrading Remote Agents for Mercurial

The remote agent installer has been modified for Bamboo 2.7 to handle Mercurial source code repositories. This update only impacts Mercurial Plans/Jobs that use the "SSH/Keyfile with passphrase" option to access remote repository:

Screenshot: Choosing the 'Keyfile with passphrase' Option for a Plan or Job

If you need to access Mercurial repository via the SSH protocol with a passphrase-protected keyfile, then you need to upgrade your remote agents to version 2.7 or later. Otherwise, you can keep your old agent. To upgrade your remote agents:

1. Obtain the Bamboo remote agent for version 2.7 or later (i.e. atlassian-bamboo-agent-installer-x.x.jar where 'x.x' is 2.7 or later). Refer to Bamboo Remote Agent Installation Guide for more information.
2. Use this file to replace your existing atlassian-bamboo-agent-installer.x.x.jar (where 'x.x' is 2.6 or earlier) on the computers running your Bamboo remote agents.
3. Restart the remote agent (i.e. kill it among with accompanying wrapper processes and then issue the command `java -jar atlassian-bamboo-agent-installer-2.7.jar yourBambooAgentServer`).

This procedure should prepare your agent to build Mercurial plans using passphrase-protected SSH keyfiles.

**Perforce**

The instructions on this page describe how to configure a Perforce source repository for either a Plan or a Job.

When creating a new Plan, you must define a 'default source repository'. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The 'default source repository' is initially used by the Plan's 'Default Job' and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

---

**Configuring a Perforce Source Repository**

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

**To configure a Perforce repository:**

1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
2. Enter the details for the Perforce repository as described below:
   
   - **'Use Plan's Repository'** *(Only available when configuring a Job)* — Select this option if you wish to use the 'default source repository' of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.
   
   - **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 (Workspace)** *(3)* — 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/Jobs 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 View** — The client view of the depot that contains the source code files for this Plan/Job. 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.
   
   - **Location of POM file** *(Only available when importing a Maven 2 project)* — Type the path to your project’s *pom.xml* file which is relative to the root of your Perforce Client (Workspace) *(defined above)*.
   
   - **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** *(Only available after first saving the Plan/Job with a password)* — Select this check box if you want to change the password that is used to access the Perforce repository.
   
   - **Let Bamboo manage your workspace** *(4)* — This field indicates whether or not you want Bamboo to manage your workspace.
   
   - **Enable Quiet Period** *(Only available when configuring an existing Plan)* — Select this setting to set Quiet Period parameters for the Plan's build. Upon selecting this option, the following fields become available:
      
      - **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.
      
      - **Maximum Retries**: — 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: 3' 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.

3. **Common Repository Configuration** *(Only available when configuring an existing Plan/Job)*

   - **Force Clean Build**: *(Optional)* — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
• 'Clean working directory after each build:' (Optional) — You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.

• 'Include/Exclude Files:' (Only available when configuring a Plan) — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', the following field will appear:
  • 'File Pattern:' — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.

• 'Web Repository:' — Select the type of web repository ('None', 'Generic Web Repository', 'Mercurial Web Repository', 'FishEye') to be associated with this build. You will be able to view code changes related to your build via the build results.
  Only a subset of Web Repository options are available for your chosen repository type.
  • If 'Generic Web Repository' is selected:
    • 'Web Repository URL' — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
    • 'Web Repository Module' — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.
  • If 'Mercurial Web Repository' is selected:
    • 'Mercurial Web Repository Viewer Scheme' — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial's own default web server (Default Web Repository Scheme (hgserve)).
  • If 'FishEye' is selected:
    • 'FishEye URL' — The URL of your FishEye repository (e.g. 'https://atlaseye.atlassian.com/').
    • 'Repository Name' — The name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.
    • 'Repository Path' — The path for your FishEye repository (e.g. '/atlassian/bamboo/').

How do I determine my Repository Path?
If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/

If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

• 'Build Strategy' (Only available when configuring a Plan) — Choose one of the build strategy options (listed below), which will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.
  
  You may need to configure other options specific to your chosen build strategy.

  If you select Manual & dependent builds only when creating a new Plan, an initial build will not automatically be run. You can force an initial build to be executed automatically by adding the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.

4. Click the 'Save' button to save your changes.
Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
</table>

Screenshot above: Source Repository — Perforce
Polling the Repository for changes

Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.

This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.

Polling the Repository for Changes

The repository triggers the build when changes are committed

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.

Repository Triggers the Build when Changes are Committed

Cron Based Scheduling

Bamboo will trigger a build of this plan based on a Cron expression.

This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Cron Based Scheduling

Single daily build

Bamboo will trigger a build of this plan once per day at a specified time.

This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Single Daily Build

Manual & dependent builds only

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.

This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.

Triggering a Plan Build Manually

Notes

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

2. Keep your Perforce configuration up to date — If you are using Perforce as your repository, you must ensure your Perforce configuration in Bamboo is in sync with any changes to your Perforce repository (such as client, depot or user credential changes). If not, your Perforce repository changes may cause unexpected behaviour in Bamboo when Bamboo tries to access the repository.

3. Issue when running Bamboo with Perforce prior to Bamboo 2.0.7 — A known issue exists when running Bamboo with Perforce prior to Bamboo 2.0.7 (See BAM-2866 and BAM-2849). If you change the name of your Perforce client (i.e. via an update) without updating your Perforce configuration in Bamboo, Bamboo will not be able to find the Perforce client to run against. Perforce will then create a default client in your running directory. This can lead to situations where Bamboo will attempt to clear out data from your running directory (e.g. force build). To avoid this problem, ensure that you update the 'Client' in your Perforce configuration whenever you change your Perforce client.

4. Please be aware of the following implications when either letting Bamboo manage or preventing Bamboo from managing your workspace:
   - **If you let Bamboo manage your workspace,**
     - We recommend this configuration if your Jobs 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. Perforce will modify the host and root). You need to ensure that you enter a Client Workspace in the 'Client' field that will be used solely for Bamboo.
     - Under this configuration, you should configure one client per Job 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 Jobs, 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 Job will be built on.
     - Please note that alternate roots does not currently work in Bamboo. See issue BAM-2377 for further details.

Related Topics

Specifying the Source Repository for a Plan  
Specifying the Source Repository for a Job

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.
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 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;

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

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

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

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

```
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

The instructions on this page describe how to configure a Subversion source repository for either a Plan or a Job.

When creating a new Plan, you must define a ‘default source repository’. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the plan will use.

The ‘default source repository’ is initially used by the Plan’s ‘Default Job’ and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

On this page:
- Configuring a Subversion Source Repository
- Appendix - Build Strategies
- Notes

Configuring a Subversion Source Repository

The instructions in this section apply to configuring a source repository for both Plans and Jobs.

To configure a Subversion repository:

1. If you are creating a new Plan or new Job, or have come from the Editing a Plan or Editing a Job topics, start at step 4.

   1. Navigate to the source repository settings for a Plan or Job, as described on Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.
   2. Enter the details for the Subversion repository as described below:

      - ‘Use Plan’s Repository’ (Only available when configuring a Job) — Select this option if you wish to use the ‘default source repository’ of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below, although if you are creating a new Job, continue with the Builder Configuration section of Creating a New Job.

      - ‘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’, ‘SSH’ or ‘SSL Client Certificate’.

         - If you select ‘Password’, the following fields will appear:

            - ‘Password’ (Optional) — Type the password required by the Subversion username (if applicable).
            - ‘Change Password’ (Only available after first saving the Plan/Job with a password) — 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’ (Only available after first saving the Plan/Job with a passphrase) — Select this check box if you want to change the password for your SSH private key.

            - If you are planning to use remote agents the ssh private key file has to be copied to the agent box into the same location as specified.

         - If you select ‘SSL Client Certificate’, the following fields will appear:

            - ‘Private Key’ — Type the absolute path of your SSL client certificate.
            - ‘Passphrase’ — Type the passphrase for your SSL client certificate.

            - Please note, the client certificate has to be in PKCS12 format and the client certificate file must be passphrase protected, otherwise a runtime exception is thrown by JDK security engine while opening the user key.

      3. Advanced Options (The following fields will only appear if you select the ‘Show Advanced Options’ check box)

         - ‘Enable Quiet Period’: (Only available when configuring an existing Plan) — Select this setting to set Quiet Period parameters for the Plan’s build. Upon selecting this option, the following fields become available.

            - ‘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’: — You can specify how many times Bamboo should check for new changes using the Quiet Period settings.
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.

4. Common Repository Configuration (Only available when configuring an existing Plan/Job)

- 'Force Clean Build:' (Optional) — You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.
- 'Clean working directory after each build:' (Optional) — You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.
- 'Include/Exclude Files:' (Only available when configuring a Plan) — You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', the following field will appear:
  - 'File Pattern:' — The regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.
- 'Web Repository:' — Select the type of web repository ('None', 'Generic Web Repository', 'Mercurial Web Repository', 'FishEye') to be associated with this build. You will be able to view code changes related to your build via the build results.

Only a subset of Web Repository options are available for your chosen repository type.

If 'Generic Web Repository' is selected:
- 'Web Repository URL' — If your source repository can be accessed via a web browser, you can specify the URL of the source repository here. If you specify a Web Repository URL, then links to relevant files will be displayed in the 'Code Changes' section of a build result.
- 'Web Repository Module' — The repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.

If 'Mercurial Web Repository' is selected:
- 'Mercurial Web Repository Viewer Scheme' — Choose between using the BitBucket Web Repository Scheme (if you use BitBucket) or Mercurial’s own default web server (Default Web Repository Scheme (hgserve)).

If 'FishEye' is selected:
- 'FishEye URL' — The URL of your FishEye repository (e.g. 'https://atlas-eye.atlassian.com/').
- 'Repository Name' — The name of your FishEye repository (e.g. 'bamboo'). This is effectively the alias for your repository path.
- 'Repository Path' — The path for your FishEye repository (e.g. '/atlassian/bamboo/').

How do I determine my Repository Path?
If you have previously run builds with changes from your repository, the easiest way of determining your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/.
If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

- 'Build Strategy' (Only available when configuring a Plan) — Choose one of the build strategy options (listed below), which will be used for triggering the execution of this Plan. You can change the Build Strategy at a later point in time as required.

You may need to configure other options specific to your chosen build strategy.

If you select Manual & dependent builds only when creating a new Plan, an initial build will not automatically be run. You can force an initial build to be executed automatically by adding the fire.initial.build.for.manual.strategy to your bamboo.cfg.xml file as described in Configuring System Properties.

5. Click the 'Save' button to save your changes.
Appendix - Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will 'poll' the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a 'check out' or 'update' command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
</tbody>
</table>
The repository triggers the build when changes are committed

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.

Repository Triggers the Build when Changes are Committed

Cron Based Scheduling

Bamboo will trigger a build of this plan based on a Cron expression.

This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Cron Based Scheduling

Single daily build

Bamboo will trigger a build of this plan once per day at a specified time.

This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Single Daily Build

Manual & dependent builds only

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.

This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.

Triggering a Plan Build Manually

Notes

- If you are having problems connecting to Subversion, consult our documentation on troubleshooting Subversion connections.
- 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.

Related Topics

Specifying the Source Repository for a Plan
Specifying the Source Repository for a Job

Configuring Source Code Management Triggers for Subversion

This page provides instructions on how to configure Subversion to send message events that trigger the execution of Bamboo plans. You only need to configure Subversion to send these message events if 'The repository triggers the build when changes are committed' build strategy has been specified for one or more of your Bamboo plans.

On this page:

- Configuring Subversion to Trigger a Build
- Notes

Configuring Subversion to Trigger a Build

This section explains how to configure Subversion to trigger a build when the repository is changed. A Subversion hook script is used to perform the trigger action whenever a Subversion repository is changed.

The following commands and script files assume that your Subversion server runs on a UNIX- or Linux-based operating system. If your Subversion server runs on any other operating system, then you will need to modify the script files and if necessary, the commands below to suit that operating system.

Step 1. Enable the Subversion post-commit hook

To do this, run the following commands:

```
  cd svn-repository-containing-the-build-source-code
  cd into the hooks/ directory

  cp post-commit.tmpl post-commit
  chmod a+rx post-commit
```

The Subversion post-commit file is not installed by default. If it does not exist, make a copy of the post-commit.tmpl file in the hooks/ directory, name it post-commit and make it executable:

```
  cp post-commit.tmpl post-commit
  chmod a+rx post-commit
```

Step 2. Install the Post-Commit Trigger

Add a line like the following to the post-commit file, for running Bamboo's build trigger script file.
/path-to-your-bamboo-installation/scripts/svn-triggers/postCommitBuildTrigger.sh base-url
BUILD-KEY

where:

- `base-url` is the base URL of the Bamboo server (such as `http://<name-of-machine>:8085`).
- `BUILD-KEY` is the key of the Bamboo plan to be executed.

Make Bamboo's build trigger script file executable (using `chmod`) so that the Subversion user can execute it.

Step 3. Do a test commit

Conduct a 'test' commit. Bamboo should start building the relevant plan after a few seconds.

The Bamboo log file should contain an entry like this:

```
[INFO] com.atlassian.bamboo.build.UpdateAndBuild - Bamboo build was triggered by remote http call from 127.0.0.1
```

Notes

1. **Build Trigger Security** — Bamboo will only accept remote build triggers if the triggers originated from the Subversion server(s) identified in the Subversion **Repository URL** of any Bamboo plans. Requests originating from other Subversion servers will be rejected by Bamboo.

Related Topics

- [Subversion](http://subversion.apache.org/)
- [Configuring Source Code Management Triggers for CVS](http://subversion.apache.org/)
- [Copy of REV31 - Subversion](http://subversion.apache.org/)

On this page:

- Use Plan's Repository
- Source Repository
- Advanced Options
- Common Repository Configuration
- Trigger Conditions
- Screenshot
- Appendix - Build Strategies
- Notes

**Use Plan's Repository**

If you are configuring a Job, you can elect to use the 'default source repository' of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below.

**Source Repository**

<table>
<thead>
<tr>
<th>Source Control</th>
<th>Select 'Subversion'.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository URL</td>
<td>Enter the location of your Subversion repository (e.g. <code>http://svn.collab.net/repos/svn/trunk</code>). You can use global variables in this field (see Using Global, Plan or Build-specific Variables). If you are importing a Maven 2 Project, this location should contain your project's pom.xml file.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the Subversion username used to access the repository.</td>
</tr>
<tr>
<td>Authentication Type</td>
<td>'Password'</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>*'Password' — Enter the password required by the Subversion username (if applicable).</td>
</tr>
<tr>
<td></td>
<td>*'Change Password' — Select this check box if you want to change the password that is used to access the Subversion repository.</td>
</tr>
</tbody>
</table>

| 'SSH' | *'Private Key' — Enter the absolute path of your SSH private key. |
|       | *'Passphrase' — Enter the passphrase for your SSH private key. |
|       | *'Change Passphrase' (Only available after first saving the Plan/Job with a passphrase) — Select this check box if you want to change the password for your SSH private key.  |
|       | Note, if you are planning to use remote agents, the ssh private key file must be copied to the agent box into the same location as specified. |

| 'SSL Client Certificate' | *'Private Key' — Enter the absolute path of your SSL client certificate. |
|                         | *'Passphrase' — Enter the passphrase for your SSL client certificate. |
|                         | Note, the client certificate must be in PKCS12 format and the client certificate file must be passphrase protected, otherwise a runtime exception will be thrown by the JDK security engine while opening the user key. |

### Advanced Options

The following fields will only appear if you select the 'Show Advanced Options' check box.

| 'Detect Changes in Externals' | Select this check box if your Subversion repository uses svn:externals to link to other repositories (note, your externals must be in the root of the checkout directory, not in a subdirectory). 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' | Select this option to force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built. Note, this will greatly increase the time it takes to complete a build. |
| 'Clean working directory after each build' | Select this option to force Bamboo to remove the source directory after the Plan/Job is completed building. Note, this may increase build times but saves on disk space. |
| 'Enable Quiet Period' | Select this setting to set Quiet Period parameters for the Plan's build. Upon selecting this option, the following fields become available: |
|                     | • '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. |
|                     | • 'Maximum Retries' — 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. |

| 'Include/Exclude Files' | Specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', you must enter a 'File Pattern': |
|                         | • 'File Pattern' — Enter the regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository. |
Select the type of web repository to be associated with this build. By selecting this, you will be able to view code changes related to your build via the build results. Note, only a subset of Web Repository options are available for your chosen repository type.

<table>
<thead>
<tr>
<th>Web Repository</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No further configuration required.</td>
</tr>
<tr>
<td>Generic Web Repository</td>
<td>* 'Web Repository URL' — If your source repository can be accessed via a web browser, specify the URL of the source 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.</td>
</tr>
<tr>
<td></td>
<td>* 'Web Repository Module' — Enter the repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.</td>
</tr>
<tr>
<td>FishEye</td>
<td>* 'FishEye URL' — Enter the URL of your FishEye repository (e.g. '<a href="https://atlaseye.atlassian.com/">https://atlaseye.atlassian.com/</a>').</td>
</tr>
<tr>
<td></td>
<td>* 'Repository Name' — Enter the name of your FishEye repository (e.g. 'Bamboo'). This is effectively the alias for your repository path.</td>
</tr>
<tr>
<td></td>
<td>* 'Repository Path' — Enter the path for your FishEye repository (e.g. '/atlassian/bamboo/').</td>
</tr>
</tbody>
</table>

If you have previously run builds with changes from your repository, the easiest way to determine your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk)). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

| Build Strategy       | See Appendix - Build Strategies below. |

Trigger Conditions

This field is only available when configuring an existing Plan.

Choose whether you only want to run builds of this Plan, if other Plans are currently passing. If you tick the 'Only run Build if other Plans are currently passing', you must specify the Plans:

| Plan Keys            | Enter the keys of the Plans that must pass before this Plan can be built. Separate Plan keys with commas. |

Screenshot
**Appendix - Build Strategies**

This table lists Bamboo’s available **build strategies** that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.

<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will ‘poll’ the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
</tbody>
</table>
The repository triggers the build when changes are committed

Repository
Triggers the Build when Changes are Committed

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Bamboo will trigger a build of this plan once per day at a specified time.

This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.

This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.

The repository triggers the build when changes are committed

Repository
Triggers the Build when Changes are Committed

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.

Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.

This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Bamboo will trigger a build of this plan once per day at a specified time.

This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.

Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.

This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.

Notes

- If you are having problems connecting to Subversion, consult our documentation on troubleshooting Subversion connections.
- If you use pre-1.6 Subversion client to access code checked out by Bamboo, you may encounter problems with your builds. This is due to a change in the default SVNKit behaviour in Bamboo 3.1. Please read this FAQ: How do I manually set the version of new Subversion workspaces?

Related Topics

Specifying the Source Repository for a Plan
Specifying the Source Repository for a Job

REV31 - Subversion

Subversion is an open source version control system, developed as a project of the Apache Software Foundation. For more information on Subversion, please see http://subversion.apache.org/

The instructions on this page describe how to configure a Subversion source repository for either a Plan or a Job. For instructions on how to navigate to the source repository configuration for a Plan/Job, see Specifying the Source Repository for a Plan and Specifying the Source Repository for a Job respectively.

On this page:

- Use Plan’s Repository
- Advanced Options
- Common Repository Configuration and Trigger Conditions
- Notes

Use Plan’s Repository

If you are configuring a Job, you can elect to use the ‘default source repository’ of the Plan to which your Job belongs. If you select this option, you do not need to configure any further source repository options below.

Repository Details

Set the ‘Repository URL’ to the location of your Subversion repository (e.g. http://svn.collab.net/repos/svn/trunk). You can use global variables in this field (see Using Global or Build-specific Variables).

If you are importing a Maven 2 Project, this location should contain your project’s pom.xml file.

Authentication

(Optional) Configure the ‘Username’ and ‘Authentication Type’ for the user used to access the repository.

‘Authentication Type:’

Password

- ‘Password’ — Enter the password required by the Subversion username (if applicable).
- ‘Change Password’ — Select this check box if you want to change the password that is used to access the Subversion repository.
SSH

- ‘Private Key’ — Enter the absolute path of your SSH private key.
- ‘Passphrase’ — Enter the passphrase for your SSH private key.
- ‘Change Passphrase’ (Only available after first saving the Plan/Job with a passphrase) — Select this check box if you want to change the password for your SSH private key.

Note, if you are planning to use remote agents, the ssh private key file must be copied to the agent box into the same location as specified.

SSL Client Certificate

- ‘Private Key’ — Enter the absolute path of your SSL client certificate.
- ‘Passphrase’ — Enter the passphrase for your SSL client certificate.

Note, the client certificate must be in PKCS12 format and the client certificate file must be passphrase protected, otherwise a runtime exception will be thrown by the JDK security engine while opening the user key.

Advanced Options

The following fields will only appear if you select the ‘Show Advanced Options’ check box.

Detect Changes in Externals

Select this check box if your Subversion repository uses svn:externals to link to other repositories (note, your externals must be in the root of the checkout directory, not in a subdirectory).

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 and Trigger Conditions

For information on configuring these options, see REV31 - Configuring Common Repository Options.
**Source Repository**

**Repository**
- Subversion

**Repository URL**
- `https://svn.atlassian.com/svn/private/atlassian/bamboo/trunk`

**Username**
- ali

**Authentication Type**
- Password

**Advanced Options**
- Show Advanced Options

**Common repository configuration**

- **Force Clean Build**
  - Removes the source directory and checks it out again prior to each build. This may significantly increase build time.

- **Clean working directory after each build**
  - Removes the source directory after each build. This may increase build times but save disk space.

**Include / Exclude Files**
- None

**Web Repository**
- FishEye

**FishEye URL**
- `https://atlassian.atlassian.com`

**Repository Name**
- bamboo

**Repository Path**
- `/atlassian/bamboo/`

**Build Strategy**
- Single daily build

**Build Time**
- 00:00

**Trigger Conditions**
- Only run Build if other Plans are currently passing

---

**Notes**

- If you are having problems connecting to Subversion, consult our documentation on troubleshooting Subversion connections.
- If you use pre-1.6 Subversion client to access code checked out by Bamboo, you may encounter problems with your builds. This is due to a change in the default SVNKit behaviour in Bamboo 3.1. Please read this FAQ: How do I manually set the version of new Subversion workspaces?

**Related Topics**

- Specifying the Source Repository for a Plan
- Specifying the Source Repository for a Job
- REV31 - Configuring Common Repository Options

**REV31 - Configuring Common Repository Options**

The configuration options described below are common for all types of repositories in Bamboo. The instructions on this page should be used together with the repository-specific instructions on the following pages:

- Bitbucket - Mercurial
Configuring Common Repository Options

Force Clean Build

You can force Bamboo to remove the source directory and check it out again prior to the Plan/Job build being built by selecting this option. Please note that this will greatly increase the time it takes to complete a build.

Clean working directory after each build

You can force Bamboo to remove the source directory after the Plan/Job is completed building by selecting this option. Please note that this may increase build times but saves on disk space.

Enable Quiet Period

This field is only available when configuring an existing Plan.

Select this setting to set Quiet Period parameters for the Plan's build. Upon selecting this option, the following fields become available:

- **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.
- **Maximum Retries**: 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.

Include/Exclude Files

This field is only available when configuring a Plan.

You can specify a particular inclusion or exclusion pattern for file changes to be detected. If you select an option other than 'None', you must enter a 'File Pattern':

- **File Pattern** — Enter the regular expression for file changes which you wish to include/exclude. The regex pattern must match the file path in the repository.

Web Repository

None

No further configuration required.

Generic Web Repository

- **Web Repository URL** — If your source repository can be accessed via a web browser, specify the URL of the source 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.
- **Web Repository Module** — Enter the repository name of the Plan/Job, if the above Web Repository URL points to multiple repositories.

FishEye

- **FishEye URL** — Enter the URL of your FishEye repository (e.g. https://atlas-eye.atlassian.com/).
- **Repository Name** — Enter the name of your FishEye repository (e.g. 'bamboo'). This is effectively the alias for your repository path.
- **Repository Path** — Enter the path for your FishEye repository (e.g. '/atlassian/bamboo/'). If you have previously run builds with changes from your repository, the easiest way to determine your repository path is to view the code changes and copy the path from the start of the path of one of the changed files, up to (but not including) the appropriate root directory. The root directories for repositories are the ones shown by FishEye when browsing a repository (e.g. trunk). For example, if a code change listed /atlassian/bamboo/trunk/bamboo-acceptance-test/pom.xml, the path would be /atlassian/bamboo/. If you have not previously run builds with changes from your repository, you will need to ask your FishEye administrator for the repository path indexed by FishEye.

Build Strategies

This table lists Bamboo's available build strategies that determine how the execution of a plan (i.e. a build) is triggered. Each build strategy has other options (listed at the far right of this table), which may also require configuration.
<table>
<thead>
<tr>
<th>Build strategy option</th>
<th>Description</th>
<th>Reason for choosing</th>
<th>Related documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling the Repository for changes</td>
<td>Bamboo will ‘poll’ the source code repository for changes based on set intervals or a schedule. If Bamboo detects a change to any code in this repository, a build of this plan will be triggered.</td>
<td>This is the simplest option. However, this does mean that your SCM must service a ‘check out’ or ‘update’ command whenever it is polled, even if no code has changed in the repository.</td>
<td>Polling the Repository for Changes</td>
</tr>
<tr>
<td>The repository triggers the build when changes are committed</td>
<td>Bamboo will wait to receive a message from the source code repository (specified above) about any code changes in this repository. When Bamboo receives such a message, Bamboo will trigger a build of this plan.</td>
<td>This option minimises server load as message events are sent only when code changes to this repository are committed. However, you must configure your source code management system to send message events to Bamboo about code changes in this repository.</td>
<td>Repository Triggers the Build when Changes are Committed</td>
</tr>
<tr>
<td>Cron Based Scheduling</td>
<td>Bamboo will trigger a build of this plan based on a Cron expression.</td>
<td>This option allows you to schedule builds when server load is likely to be minimal, for example, outside office hours. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Cron Based Scheduling</td>
</tr>
<tr>
<td>Single daily build</td>
<td>Bamboo will trigger a build of this plan once per day at a specified time.</td>
<td>This option is suitable if a build of this plan takes a long time to complete. Scheduled builds are triggered irrespective of any code changes in the source code repository.</td>
<td>Single Daily Build</td>
</tr>
<tr>
<td>Manual &amp; dependent builds only</td>
<td>Bamboo only triggers a build of this plan when the user chooses this function manually or through a build dependency.</td>
<td>This option is suitable if a build of this plan will fail, perhaps due to source code problems of failing tests. This frees up Bamboo agents to build other plans which are less likely to fail.</td>
<td>Triggering a Plan Build Manually</td>
</tr>
</tbody>
</table>

Setting Trigger Conditions

This field is only available when configuring an existing Plan.

Choose whether you only want to run builds of this Plan, if other Plans are currently passing. If you tick the ‘Only run Build if other Plans are currently passing’, you must specify the Plans:

- **Plan Keys** — Enter the keys of the Plans separated by commas.

Notes Related Topics

Specifying the Source Repository for a Plan

Disabling or Deleting a Plan

Bamboo allows you to disable or delete Plans/Jobs that you don't want to be built:

- **Disabling a Plan/Job** prevents it from being built. Disabling a Plan prevents Bamboo from submitting that Plan to the build queue. Disabling a Job prevents Bamboo from building that single Job within a Plan, allowing the rest of the Plan's Jobs to be built. You can re-enable the Plan/Job, if you want to build it again.

  For example, if a Plan's/Job's latest build is broken and cannot be fixed quickly, you may want to disable it to temporarily stop the Plan/Job from being built.

- **Deleting a Plan/Job** removes it completely from your Bamboo system. You will need to recreate the new Plan/Job from scratch, if you want to build it again.

  For example, if a Plan/Job is no longer relevant, you may want to delete it.

On this page:

- Disabling a Plan
- Deleting a Plan

Disabling a Plan

To disable a Plan:

1. Navigate to the desired Plan, as described in **Configuring a Plan**.
2. Click the ‘Actions’ dropdown and click ‘Disable Plan’. The plan will be disabled.

Deleting a Plan

Deleting a Plan deletes everything related to that Plan, including the Plan's configuration, all of the Plan's Job configurations and the Plan's Job build results, artifacts, labels and comments.
Before you begin:

- If you need to keep a permanent record of the Job build results for your Plan, see Exporting Data for Backup.
- The 'Admin' global permission is required to delete a Plan.
- A Plan that is currently being built cannot be deleted. If you need to delete such a Plan, stop the Plan's build first. Refer to Stopping an Active Job Build for more information.

To delete a Plan:

There are two ways to delete a Plan:

- From the Dashboard:
  1. Navigate to the desired Plan, as described in Configuring a Plan.
  2. Click the 'Actions' menu (at the right of this page) and select the 'Delete Plan' menu option.
- Via the Administration Console:
  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 above: Deleting Plans via Bamboo's Administration Console

Related Topics

Configuring a Plan
Exporting Data for Backup
Stopping an Active Job Build

Deleting the Results of a Plan Build

If the results of a Plan build* is no longer required, you can completely remove it from your Bamboo system. The 'results of a Plan build' (also known as a Plan build result) includes the results of all Job builds that were processed as part of an individual Plan build (with a specific build number).

On this page:

- Deleting the Results of a Plan Build
- Notes

Deleting the Results of a Plan Build

Before you begin:

- The 'Admin' global permission or 'Admin' plan permission is required to delete Plan build results.
- The result of a Plan build cannot be deleted if that Plan is currently being built. If you need to delete the result of a Plan build, stop the Plan's build first. Refer to Stopping an Active Job Build for more information.

To delete the result of a Plan build:

1. Navigate to the desired Plan, as described in Configuring a Plan.
2. Click the 'History' tab. A table of the previous Plan build results will be displayed, with the most recent builds at the top of the table.
3. Locate the desired build result and click the 'Delete' (trash) icon (see screenshot below).
4. Confirm the removal of the Plan build result. The Plan build result and any artifacts generated as a result of the Plan build's execution will be deleted.

![Completed Plan Results](image)

**Completed Plan Results**

<table>
<thead>
<tr>
<th>Status</th>
<th>Reason</th>
<th>Completed</th>
<th>Duration</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ #36</td>
<td>Scheduled build</td>
<td>Tue, 8 Feb, 01:05 AM</td>
<td>55 minutes</td>
<td>45 passed</td>
</tr>
<tr>
<td>✔️ #35</td>
<td>Scheduled build</td>
<td>Mon, 7 Feb, 12:35 AM</td>
<td>35 minutes</td>
<td>56 passed</td>
</tr>
<tr>
<td>✔️ #34</td>
<td>Scheduled build</td>
<td>Sun, 5 Feb, 12:40 AM</td>
<td>30 minutes</td>
<td>24 passed</td>
</tr>
<tr>
<td>✔️ #32</td>
<td>Scheduled build</td>
<td>Sat, 5 Feb, 12:32 AM</td>
<td>32 minutes</td>
<td>48 passed</td>
</tr>
<tr>
<td>✔️ #31</td>
<td>Scheduled build</td>
<td>Fri, 4 Feb, 12:27 AM</td>
<td>27 minutes</td>
<td>10 passed</td>
</tr>
<tr>
<td>✔️ #30</td>
<td>Scheduled build</td>
<td>Thu, 3 Feb, 12:38 AM</td>
<td>30 minutes</td>
<td>56 passed</td>
</tr>
</tbody>
</table>

*Screenshot above: Build History of a Plan listing individual Plan build results*

**Notes**

- You can also remove Job build result data that reaches a particular age. See Configuring Global Expiry of Job Build Results or Configuring Expiry of a Plan's Job Build Results for more information.

**Related Topics**

Deleting a Job's Current Working Files

**Stopping an Active Plan Build**

The instructions on this page describe how to stop an Plan build that is running.

**On this page:**
- Stopping an Active Plan Build
- Notes

**Stopping an Active Plan Build**

To prevent Bamboo submitting a Plan to the build queue, refer to Disabling or Deleting a Plan.

**To stop an active Plan build:**

1. Click 'Home' to go to the Dashboard and do either of the following:
   - Click the Current Activity tab and locate the relevant Plan in the Building section of this page, or
   - Click the All Plans tab and locate the relevant Plan in the list.
2. Click the 'Stop Build' icon next to the relevant Plan

   *To start a building a Plan manually, see Triggering a Plan Build Manually.*

**Notes**

- If your Bamboo server runs on Windows, it may only be possible to stop an active build by going to the Windows Task Manager and ending the relevant processes.

**Related Topics**

Triggering a Plan Build Manually
Disabling or Deleting a Plan
Disabling or Deleting a Job
Stopping an Active Plan Build

**Configuring Concurrent Builds**

Bamboo's concurrent builds feature allows you to run more than one build of a single Plan concurrently on multiple agents. This is extremely useful if a Plan's build triggers are likely to fire in a shorter period of time than what Bamboo requires to complete running the Plan's builds.

You can configure a default value for the maximum number of builds of a Plan that your Bamboo server can run concurrently, via the Bamboo administration console. This value is a default – it can be overridden in a Plan's configuration.
To configure the default maximum number of concurrent builds of a Plan allowed by Bamboo:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Concurrent Builds' link in the left navigation column under the 'Plans' subheading. The Concurrent Builds page is displayed. The maximum number of builds of a Plan that Bamboo will run concurrently by default, is displayed next to Default number of concurrent builds allowed.
3. Click the 'Edit' button.
4. In the Default number of concurrent builds allowed field, enter the default maximum number of builds of a Plan that Bamboo can run concurrently on different agents. Be aware that this value can be overridden in a Plan's configuration.
5. Click the 'Save' button to save your changes.

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.
- Run manual build
  This action allows you to run manual builds for multiple plans at once, with the option of triggering dependencies.
- Update Maven 2 dependencies
  This action allows you to update the Maven 2 dependency option of multiple plans at once.

To navigate to Bulk Actions:

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.

On this page:

- Adding a New Notification to Multiple Plans
- Updating the SVN Repository URL for Multiple Plans
- Updating the SVN Repository Credentials for Multiple Plans
- Updating the CVS Root and Credentials for Multiple Plans
- Updating the CVS Module for Multiple Plans
- Updating the Web Repository URL for Multiple Plans
- Running Manual Builds for Multiple Plans
- Updating the Maven 2 Dependencies for Multiple Plans
Adding a New Notification to Multiple Plans

To add a new notification to multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select ‘Add new notification’ from the ‘Select bulk action’ dropdown and click ‘Next’. The ‘Choose Plans’ screen will display.
3. Select the plans that you wish to add a new notification to and click ‘Next’. The ‘Perform Bulk Action’ screen will display.
4. Configure the new notification that you want to add to the previously selected plans and click ‘Next’. Please see Configuring Notifications for a Plan and its Jobs if you need detailed information on adding new notifications.
5. A confirmation window will display. Click ‘Confirm’ to add your new notification to the previously selected plans.
6. The ‘Results’ page will display. Click ‘Done’ to return to the ‘Select Bulk Action’ screen.

Updating the SVN Repository URL for Multiple Plans

To update the SVN repository URL for multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select ‘Update SVN repository URL’ from the ‘Select bulk action’ dropdown and click ‘Next’. The ‘Choose Plans’ screen will display.
3. Select the plans that you wish to update the SVN repository URL for and click ‘Next’. The ‘Perform Bulk Action’ screen will display.
4. 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.
5. A confirmation window will display. Click ‘Confirm’ to update the SVN repository URL for the previously selected plans.
6. The ‘Results’ page will display. Click ‘Done’ to return to the ‘Select Bulk Action’ screen.

Updating the SVN Repository Credentials for Multiple Plans

To update the SVN repository credentials for multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select ‘Update SVN credentials’ from the ‘Select bulk action’ dropdown and click ‘Next’. The ‘Choose Plans’ screen will display.
3. Select the plans that you wish to update the SVN repository credentials for and click ‘Next’. The ‘Perform Bulk Action’ screen will display.
4. 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.
5. A confirmation window will display. Click ‘Confirm’ to update the SVN credentials for the previously selected plans.
6. The ‘Results’ page will display. Click ‘Done’ to return to the ‘Select Bulk Action’ screen.

Updating the CVS Root and Credentials for Multiple Plans

To update the CVS root and credentials for multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select ‘Update CVS root and credentials’ from the ‘Select bulk action’ dropdown and click ‘Next’. The ‘Choose Plans’ screen will display.
3. Select the plans that you wish to update the CVS root and credentials for and click ‘Next’. The ‘Perform Bulk Action’ screen will display.
4. 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.
5. A confirmation window will display. Click ‘Confirm’ to update the CVS root and credentials for the previously selected plans.
6. The ‘Results’ page will display. Click ‘Done’ to return to the ‘Select Bulk Action’ screen.
**Updating the CVS Module for Multiple Plans**

To update the CVS module for multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select 'Update CVS module' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
3. Select the plans that you wish to update the CVS module for and click 'Next'. The 'Perform Bulk Action' screen will display.
4. 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.
5. A confirmation window will display. Click 'Confirm' to update the CVS module for the previously selected plans.
6. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

**Updating the Web Repository URL for Multiple Plans**

To update the web repository URL for multiple plans:

1. Navigate to Bulk Actions, as described above.
2. Select 'Update web repository URL' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
3. Select the plans that you wish to update the web repository URL for and click 'Next'. The 'Perform Bulk Action' screen will display.
4. 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.
5. A confirmation window will display. Click 'Confirm' to update the web repository URL for the previously selected plans.
6. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

**Running Manual Builds for Multiple Plans,**

You may want to use this bulk action to initialise Maven 2 dependencies for multiple plans, since Maven 2 dependencies are only known after a build.

To run manual builds for multiple plans,

1. Navigate to Bulk Actions, as described above.
2. Select 'Run manual build' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
3. Select the plans that you wish to run manual builds for and click 'Next'. The 'Perform Bulk Action' screen will display showing the selected plans and the latest build for each of those plans.
4. If you want to disable dependencies when running the manual builds for these plans, check the 'Disable dependencies for this bulk manual build' checkbox.
5. Click 'Next'. A confirmation window will display.
6. Click 'Confirm' to run manual builds for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

**Updating the Maven 2 Dependencies for Multiple Plans**

This bulk action updates the 'Determine plan dependencies from my Maven pom.xml' field for build plans, as described in the Maven documentation.

Before you begin:

- You will need to run a build before the Maven 2 dependencies will become known to Bamboo.

To update the Maven 2 dependencies for multiple plans,

1. Navigate to Bulk Actions, as described above.
2. Select 'Enable Maven 2 Dependencies' from the 'Select bulk action' dropdown and click 'Next'. The 'Choose Plans' screen will display.
3. Select the plans that you wish to update the Maven 2 dependencies for and click 'Next'. The 'Perform Bulk Action' screen will display.
4. Choose whether or not you want to determine plan dependencies for all plans from your Maven pom.xml, by checking or unchecking the 'Determine plan dependencies from my Maven pom.xml' checkbox accordingly.
5. Click 'Next'. A confirmation window will display.
6. Click 'Confirm' to update the Maven 2 dependencies for the previously selected plans.
7. The 'Results' page will display. Click 'Done' to return to the 'Select Bulk Action' screen.

**Moving Plans to a Different Project**

A project is a collection of Plans. Projects enable you to easily group and identify plans which are logically related to each other. They are especially useful when generating reports across multiple Plans.

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 Job build results within the Plan.
Moving a Plan to a Different Project

Before you begin:

- We recommended that you back up your Bamboo build results before you move a plan. See Exporting Data for Backup for instructions.

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 to display the initial 'Move Build Plan Wizard' page.
3. Select an existing 'Destination Project' from the list in the dropdown box. You can create a new project to move your Plans to by selecting 'New Project' from this list. If you select this option, complete the following fields that appear:
   - Project Name — Enter a name for a new project or select an existing project. The plan which you create on this page will be associated with this project.
   - Project Key — Enter a unique key for this project or select an existing project key.
4. Select the check box(es) for the Plan(s) you wish to move.
   - In the list of Plans on this page, Plans are grouped under the name of the project (in bold text) to which they belong (as shown in Screenshot 1 below). Selecting a project's check box automatically selects all Plans which belong to that project.
5. Click the 'Move' button to display the 'Configure New Plan Details' page (as shown in Screenshot 2 below).
6. If the destination project already includes a Plan with the same Plan Name, or if you wish to change the Plan Name, replace the existing content in the 'New Name' field.
7. If the destination project already includes a Plan with the same Plan Key, or if you wish to change the Plan Key, replace the existing content in the 'New Key' field.
8. Click the 'Move' button.
## Move Build Plan Wizard

**Warning:** It is strongly recommended that you ensure that all agents are disabled before you perform the move. **Disable all agents**

### Select Plans

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.

<table>
<thead>
<tr>
<th>Destination Project</th>
<th>New Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Project</td>
<td>New Project</td>
</tr>
<tr>
<td>Project Name*</td>
<td>A New Project</td>
</tr>
<tr>
<td>How do you want to call the Project within Bamboo? e.g. &quot;Issue Tracking Application&quot;.</td>
<td></td>
</tr>
<tr>
<td>Project Key*</td>
<td>NEWPROJ</td>
</tr>
<tr>
<td>This is the unique Project key to identify a Project. The key must contain only uppercase alphanumeric characters. e.g. &quot;ITA&quot;.</td>
<td></td>
</tr>
</tbody>
</table>

**Select:** All, None

### Artifact Sharing Dogfooding

- [ ] Artifact sharing
- [x] CI Tests
- [ ] Maven Sharing CI Tests
- [ ] Maven Sharing CI Tests with Maven 3

### Bamboo Main

- [ ] CI Tests
- [ ] Extras
- [x] Stable CI Tests
- [ ] Stable Extras

### UI

- [ ] Default
- [ ] MC Test
- [ ] WebDriver Tests
- [ ] WebDriver Tests JDK 1.6

**Move**

*Screenshot 1 above: 'Moving Plans - Select Plans'*
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 to display the Plan's Configuration pages.
4. Click the ‘Dependencies’ tab. Any dependencies that currently exist for this Plan are displayed on this page, as follows:
   - **Automatic Dependency Management** section — Select the Automatic Dependency Management check box if you want Bamboo to automatically establish this Plan's build dependencies based on the dependencies defined in the `pom.xml` file of one of its Jobs. Please see the Notes below.
   - In the Dependency provider field, select the appropriate Job of this Plan whose `pom.xml` you want Bamboo to base the Plan's build dependencies on. Only Jobs which use a Maven 2 builder are displayed in this field.
   - **Manual Dependency Management** section — All Plans in your Bamboo system (other than the one you are configuring) are listed in two ‘Parent plans’ and ‘Child plans’ sections (see screenshot below). Select the check boxes of the Plans in the following subsections:
     - **Child plans** — Plans whose builds you want triggered as a result of successful completion of the Plan you are configuring.
     - **Parent plans** — Plans whose successful completion will trigger a build of the Plan you are configuring. Any other Plans in your Bamboo system that specify build dependencies, with the Plan you are configuring, will have their check boxes selected.
**Dependency Blocking Strategy** section — Select the dependency blocking strategy that you want to apply to this build. For more information, see [Dependency Blocking Strategies](#). Please note, dependency blocking only works when the Plan uses a build strategy based on source code updates.

5. Click the 'Save' button.

---

### Notes

- If you choose 'Automatic Dependency Management', please note:
  - Maven 2 will parse your `pom.xml` after the Job’s initial build and generate the dependencies.
  - You may not need to specify any further build dependency options in the Manual Dependency Management section.
  - This feature is not applicable, nor available if you are using Maven 1.

- **Build Dependencies and Build Strategies** — Build dependencies work together with the build strategies of Plans to trigger builds of these Plans. For example, you can set up Plan A to poll its repository for changes as well as configure a build dependency on a parent Plan (Plan B). In such a case, builds of Plan A will be triggered when code changes are detected in its repository as well as when builds of Plan B 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 Plan Build Manually](#).

- **Tips for Setting Up Dependent Builds** — Please take note of the following information when setting up dependent builds:
  - 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 Plan 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.

### Dependency Blocking Strategies

Dependency blocking is an advanced feature of dependent build triggering that can be used to manage Plan builds with parent build dependencies. This ensures that a "tree" of dependent builds always runs in tree hierarchy order, even if child Plan builds are triggered independently of their parents.
Dependency blocking only works when the Plan uses a build strategy based on source code updates (i.e. the Polling the Repository for changes or Repository triggers the build when changes are committed build strategy options). This feature will not work when a Plan uses a build strategy based on a schedule or triggered via a parent build (when there are multiple parent Plan builds in progress).

The three dependency blocking strategies are explained below:

- **'Do not block'** — If a Plan build with this dependency blocking strategy is triggered by a source code update, the Plan build will run whenever it is triggered by a source code update, regardless of whether or not it has parent Plan build dependencies.
- **'Block build if parent builds are queued or in progress'** — If a Plan build with this dependency blocking strategy is triggered by a source code update, Bamboo will check whether its parent Plans' Jobs are building or are waiting in the build queue.
  - If so, the Plan's build will be blocked.
  - If not, the Plan's build will run.
- **'Block build if parent plans have unbuilt changes'** — If a Plan build with this dependency blocking strategy is triggered by a code update, Bamboo will check whether its parent Plans are building, queued to build or have changes.
  - If so, the Plan build will be blocked.
  - If not, Bamboo will check if any of the parent Plan's builds have changes. If there are parent Plan builds with changes, those Plans will be triggered and your Plan's build will be blocked. If there are no parent Plan builds with changes, your Plan's build will run.

These dependence blocking strategies are illustrated in the flowchart below:

---

**Configuring a Stage**

**About Stages**

*Stages* group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.
Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

Navigating to the Stages for a Plan

To navigate to the Stages for a Plan:

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. In the list of Plans, click the name of the desired Plan. The Plan's Plan Summary page will be displayed.
3. Click the 'Actions' dropdown and click 'Configure Plan'.
4. Click the 'Stages' tab to view the stages currently set up for the Plan.

Related Topics

For more information on configuring Stages, see the following topics:

Creating a Stage Deleting a Stage Editing a Stage

Creating a Stage

The instructions on this page describe how to create a Stage in a Plan.

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

On this page:

- Creating a Stage
- Notes

Creating a Stage

To create a Stage within a Plan:

1. Navigate to the Stages for a Plan, as described in Configuring a Stage.
2. Click the 'Create Stage' button. The 'Create a new Stage' window will appear.
3. Fill out the fields and click 'Create' to create the Stage.
4. (optional) You may want to do one or more of the following with your new Stage:
   - Order your new Stage in the list of Stages, by dragging and dropping it.
   - Create a new Job to your new Stage.
   - Move a Job from another Stage to your new Stage by dragging and dropping the Job.
Notes

- You may break artifact dependencies by moving Stages or Jobs between Stages. Bamboo will warn you, if a dependency will be broken by moving a Stage or a Job.

Related Topics

Editing a Stage
Deleting a Stage
Creating a Job
Editing a Job
Configuring a Job's Build Artifacts

Deleting a Stage

The instructions on this page describe how to delete a Stage in a Plan.

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

On this page:
- Deleting a Stage
- Notes

Deleting a Stage

Before you begin, please see the Notes for this page.

To delete a Stage within a Plan:

1. Navigate to the Stages for a Plan, as described in Configuring a Stage.
2. Click the cog (🔧) icon and click 'Delete Stage'.
3. Click 'Confirm' on the confirmation screen to delete the Stage.

Delete Functional Tests

⚠️ You are about to delete Stage "Functional Tests". This Stage has 0 Job. If you proceed with this action Bamboo will irrevocably delete all Job configurations, artifacts, logs and results related to this Stage.

Confirm  Cancel

Screenshot above: Confirming deletion of a Stage

Notes

- Deleting a Stage will delete all Job configurations, artifacts, logs and results related to the Stage. These cannot be recovered after the Stage is deleted.
- You may break artifact dependencies by deleting a Stage.
Related Topics
Creating a Stage
Editing a Stage
Configuring a Job's Build Artifacts

Editing a Stage

The instructions on this page describe how to edit and move a Stage in a Plan.

Stages group (or ‘map’) Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

On this page:
- Editing a Stage
- Notes

Editing a Stage

To edit a Stage within a Plan:

1. Navigate to the Stages for a Plan, as described in Configuring a Stage.
2. Edit the Stage as desired:
   - To edit the name and description of the Stage, click the cog (⚙️) icon and click 'Configure Stage'.
   - To move the Stage, drag and drop the Stage to the desired place in the Plan.

Screenshot: Stages for a Plan

Notes
- You may break artifact dependencies by moving Stages. Bamboo will warn you, if a dependency will be broken by moving a Stage.

Related Topics
Creating a Stage
Deleting a Stage
Configuring a Job's Build Artifacts

Configuring a Job

About Jobs
A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job’s Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

Navigating to a Job
To navigate to a Job:

1. Click 'Home' to go to the Dashboard and click the 'All Plans' tab.
2. In the list of Plans, click the name of the desired Plan. The Plan's Plan Summary page will be displayed.
3. Click the 'Actions' dropdown and click 'Configure Plan'.
4. Click the 'Stages' tab to view the stages currently set up for the Plan.
5. Click the name of the Job that you want to configure.

Related Topics
For more information on configuring Jobs, see the following topics:

Creating a Job Editing a Job Disabling or Deleting a Job Configuring Tasks Specifying the Source Repository for a Job

Creating a Job
The instructions on this page describe how to create a Job in a Stage.

A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job’s Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).
Create a New Job

When you create a new Job, you can define a single unit of execution within a Stage, including what gets built and what Builder to use.

Before you begin:

- You must have the 'Admin' or 'Create Plan' global permission to create a new Job.

To create a new Job:

1. Navigate to the Stages for the desired Plan, as described on Configuring a Stage.
2. Click the 'Create Job' link in the Stage that you want to create a new Job in. Choose 'Create a New Job' on the page that appears (see screenshot below).
3. In the Job Details section, enter or select the relevant items for the following fields:
   - **Stage** — Select an existing Stage to which this Job will be added.
   - **Job Name** — Enter a name for the new Job.
   - **Job Key** — Enter a unique key for the new Job.
   - **Job Description** — Enter a short sentence to describe this Job.
4. In the Source Repository section, specify the source repository configuration used for building this Job. For details, refer to the following source repository configuration procedures (from step 4), since many of these fields and options change depending on the 'Repository' you select:
   - **CVS**
   - **Subversion**
   - **Perforce**
   - **Mercurial**
5. In the Builder Configuration section, specify the configuration of the builder used for building this Job. The builder that you select will become one of this Job's requirements. For details please see Configuring a Job's Requirements. Refer to the following documentation on Bamboo's builders (from step 2) for details about these options, since many of these fields and options change depending on the 'Builder' you select:
   - **Ant**
   - **Custom Command Builder**
   - **devenv.com**
   - **Grails**
   - **Maven**
   - **NAnt**
   - **PHPUnit**
   - **Script Builder**
6. In the Enable this Plan/Job section, choose whether or not to enable this Job. Enabling the Job instructs Bamboo to execute builds of the Job whenever a Build of the Job's Plan is triggered. To enable this Job, select the 'Yes please!' check box.
7. Click the 'Create' button to create the Job and the Plan's Stages page will be displayed, indicating your new Job in its designated Stage.

If you wish to configure more Job options, please refer to Editing a Job.
Create a New Job

On this page, you can create a completely new Job and specify its source repository and Builder. More advanced configuration options (including those for plugins) will be available to you after creating this Job.

Job Details

- **Stage**: Packaging Stage
- **Job Name**: Core Branch
- **Job Key**: CORE
- **Job Description**: This is the Core Branch Job

Source Repository

- Use Plan's Repository
- Repository URL: https://svn.atlassian.com/svn/private/atlassian/bamboo/trunk
- Username: mchei
- Authentication Type: password

Builder Configuration

- **Builder**: Ant
- **Build File**: build.xml
- **Target**: clean test
- **Build JDK**: JDK
- **Environment Variables**: (Optional) Any extra environment variables you want to pass to your build. e.g. MAVEN_OPTS=-Xmx256m -Xms128m
- **Worker Sub Directory**: (Optional) Bamboo assumes that the build root directory is the working directory. Use this option to specify an alternative working directory (must be a subdirectory of the root directory).

Where should Bamboo look for the test result files?

- The build will produce test results. Note that test files must be in JUnit XML format. For more information on JUnit visit JUnit.org
- Specify custom results directories: **Test-reports**.xml

Enable this Job?

- Yes please!

Screenshot above: Creating a new Job

**Clone an Existing Job**
When you clone an existing Job, you make a copy of that Job and its entire configuration.

Before you begin:

- You must have the 'Admin' or 'Create Plan' global permission to clone an existing Job.

To clone an existing Job:

1. Navigate to the Stages for the desired Plan, as described on Configuring a Stage.
2. Click the 'Create Job' link in the Stage that you want to create a new Job in. Choose 'Clone an Existing Job' on the page that appears (see screenshot below).
3. In the 'Job to clone from' section and dropdown, enter or select the relevant items for the following fields:
   - 'Job to clone from' — Select the Plan containing the Job you wish to clone. Plans are grouped by Project in the list.
   - 'Job to clone' — Select the Job you wish to clone from your selected Plan. Jobs are grouped by Stage in the list.
4. In the 'Job Details' section, enter or select the relevant items for the following fields:
   - Stage — Select an existing Stage to which this Job will be added.
   - If you have not added any further Jobs or Stages to this Plan since the Plan was created, there will only be one Stage available (with the default name of Default Stage).
   - Job Name — Enter a name for the new Job.
   - Job Key — Enter a unique key for the new Job.
   - Job Description — Enter a short sentence to describe this Job.
5. In the Enable this Plan/Job section, choose whether or not to enable this Job. Enabling the Job instructs Bamboo to execute builds of the Job whenever a Build of the Job's Plan is triggered. To enable this Job, select the 'Yes please!' check box.
6. Click the 'Create' button to create the Job and the Plan's Stages page will be displayed, indicating your new Job in its designated Stage.

If you wish to configure more Job options, please refer to Editing a Job.

![Screenshot above: Cloning an existing Job](image_url)

**Notes**

Related Topics

- Configuring a Plan
- Configuring a Stage
- Editing a Job
Disabling or Deleting a Job

Editing a Job

To edit a Plan’s existing Job:

1. Navigate to the Job, as described on Configuring a Job.
2. Click the ‘Actions’ menu and click ‘Configure Job’.
3. Click the appropriate tab to begin editing that aspect of your Job:
   - ‘Job Details’ — see Renaming a Job or changing the Job’s Description (from step 3).
   - ‘Source Repository’ — see Specifying the Source Repository for a Plan or Job.
   - ‘Builder’ — see Specifying a Builder.
   - ‘Requirements’ — see Configuring a Job’s Requirements.
   - ‘Artifacts’ — see Configuring a Job’s Build Artifacts (from step 3).
   - ‘Post Actions’ — see Configuring Automatic Labelling of Job Build Results (from step 3).
   - ‘Miscellaneous’ — see Configuring Miscellaneous Settings for a Job.

Screenshot above: Job’s Configuration Pages

Configuring a Job’s Build Artifacts

Artifacts are files created by a Job build (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual Jobs.

The instructions on this page describe how to define which artifacts you want to keep from a Job’s build. For example, you may wish to keep reports, websites or files (e.g. JAR files) generated by a Job build.

You can also configure artifact sharing between Jobs in a Plan. For example, you may want to run acceptance tests on a build, share the WAR from one Job to another without rebuilding it each time. Artifact sharing is described on this page: Configuring Artifact Sharing between Jobs.

On this page:

- Defining which Artifacts to keep for a Job
- Notes
You can specify which artifacts to keep by setting up an artifact definition for the Job. The artifacts will be available after each build of a Job.

**Defining which Artifacts to keep for a Job**

To set up an artifact definition for a Job:

1. Navigate to the desired Job, as described on Configuring a Job.
2. Click the 'Artifacts' tab to display the Job's Artifacts settings (see screenshot above).
3. Click the 'Create Definition' button.
4. Complete the fields on the screen (see screenshot below) and click 'Create'. For example, if you want to keep the latest version of a JAR you have built, you could specify Copy Pattern to be '*/.jar' and the Location to be 'target'.

Please note:
- The location is relative to the build directory. Do not use the absolute path to refer to the location.
- The copy pattern is relative to the location specified.
- If you want to share artifacts with other Jobs in the plan, you will need to mark the artifacts as shared. Sharing artifacts is described on this page: Configuring Artifact Sharing between Jobs.
Notes

1. Artifacts are copied to a subdirectory (/JOB_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 artifacts (see Viewing a Build's Artifacts in the Bamboo User's Guide).

Related Topics

Viewing a Build's Artifacts
Configuring Artifact Sharing between Jobs

Configuring Artifact Sharing between Jobs

The instructions on this page describe how to configure artifact sharing between Jobs in a Plan. Artifact sharing allows you to pass an artifact from one Job to a Job in a subsequent Stage. That is, the artifact is copied to the subsequent Job's agent. Note, you cannot pass artifacts between Jobs in the same stage.

For example, you may want to run acceptance tests on a build, share the WAR from one Job to another without rebuilding it each time.

The instructions on this page also describe how to configure artifact sharing when you are using a Maven builder. In this case, the artifact is deployed to and resolved from a Maven repository by Jobs, rather than being copied from agent to agent.

On this page:

- Sharing Artifacts between Jobs
- Sharing Artifacts between Maven Jobs (Beta)
- Notes

Sharing Artifacts between Jobs

You can share artifacts between Jobs in different Stages via artifact dependencies. Each time the artifact is shared with a subsequent Job, it is copied to the Job's agent.

To share an artifact between two Jobs in different Stages:

1. Navigate to the Job that will produce the artifact, as described on Configuring a Job.
2. Click the 'Artifacts' tab to display the Job's Artifacts settings (see screenshot above).
3. Click the 'Edit' link for the artifact that you want to share. The artifact definition will be displayed.
4. Tick the 'Shared' checkbox.
5. Navigate to the Job in a subsequent Stage that will consume the artifact, and click the 'Artifacts' tab.
6. Click the 'Create Dependency' button.
7. Complete the fields on the screen (see screenshot below) and click 'Create'.

Please note:

- The artifact dropdown only lists artifacts from Jobs in previous stages that have been marked as shared. This is described in setting up artifact definitions above.
- The destination directory is relative to the build directory. Do not use the absolute path to refer to the destination directory.
Before you begin:

- Maven artifact sharing is currently in beta. We recommend that you do not use it with any critical production systems.
- Maven artifact sharing is not supported for Maven 1.

About Maven Artifact Sharing

Maven artifact sharing works by producing new remote repositories when you run a Plan. These are temporary equivalents of the deployment repositories configured within the project’s `pom.xml` files. For example, if the project deploys its repository to the 'mycorp-private' repository at [http://repository.example.com/](http://repository.example.com/) then Bamboo will create a 'mycorp-private' repository for each Plan result hosted on the Bamboo server.

When a Job produces a Maven artifact, it is deployed to this repository. A subsequent Job that consumes an artifact created by this process will resolve it from the repository within Bamboo. This process of deploying and resolving artifacts can then be repeated, as necessary across subsequent Jobs in later Stages. Once a build has completed, the temporary repositories are removed to conserve disk space, unless specified otherwise (see configuration instructions below).

The diagram below shows an example of how artifact sharing works. Two Jobs, Job A and Job B, share an artifact in Plan, MyPlan. Job A builds the artifact then deploys it to the temporary remote Maven repository. Job A triggers Job B. Job B resolves the artifact, builds, then deploys the artifact back to the repository.
Diagram above: Maven artifact sharing example

Configuring Maven Artifact Sharing

Artifact sharing is configured differently when using Maven 2 or 3 as a builder for your Jobs. Artifacts are deployed to and resolved from a Maven repository, rather than copied from agent to agent. You will need to change your pom.xml file, as well as configure the relevant Jobs to set up artifact sharing.

Before you begin:

- Your Bamboo URL must be specified correctly for Maven artifact sharing to work correctly. For example, do not use localhost as part of your Bamboo URL. For more information, see Specifying Bamboo's URL.

To share an artifact between two Maven Jobs in different Stages:

1. Edit your pom.xml file and add the following plugin definition:

```xml
<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.bamboo.maven.sharing</groupId>
      <artifactId>bamboo-artifact-sharing-maven-plugin</artifactId>
      <version>3.0-i5</version>
      <executions>
        <execution>
          <id>sharing</id>
          <goals>
            <goal>share</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

- Note, the version number specified above `<version>` must match the version of Bamboo you are using. This version number number will be compatible across minor releases of Bamboo, unless specified otherwise in the relevant upgrade.

2. Navigate to the Plan that you want to enable artifact sharing for, and click the ‘Miscellaneous’ tab (see screenshot below).

3. Tick the ‘Enabled’ checkbox.

4. If you don’t want to keep the Maven artifact repositories generated for each build of the Plan, tick the ‘Expire Maven Repository Artifacts’ checkbox.

If you do not tick this checkbox, please note that each plan result may contain hundreds of megabytes of Maven artifacts. Ensure...
that you have sufficient disk space to accommodate this.
5. Configure artifact sharing between the desired Jobs, as described above. That is,
   • If a Job needs to deploy an artifact (i.e. produce the artifact), mark the artifact as shared in the relevant artifact definition.
   • If a Job needs to resolve an artifact (i.e. consume the artifact), create an artifact dependency for the artifact that it needs to resolve.

Screenshot above: Enabling Maven artifact sharing for a Plan
Notes

1. Artifacts are copied to a subdirectory (/JOB_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 artifacts (see Viewing a Build's Artifacts in the Bamboo User's Guide).

Related Topics

Viewing a Build's Artifacts
Configuring a Job's Build Artifacts

Configuring Automatic Labelling of Job Build Results

For each Job of a Plan, you can (as an option) specify a 'post action' label that can be applied to the Job's build results automatically after (or 'post') each build of that Job.

Automatic labelling of Job builds is a 'post action' feature (i.e. performed at the completion of Job builds), which is built into Bamboo itself. There are a number of third-party plugin modules available that can provide additional post actions (e.g. the Pre-Post Build Command plugin). You can also write your own plugins to provide additional post actions for a Job. See the Bamboo Plugin Guide for further details.

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.

On this page:

- Specifying Labels for a Job's Build Results
- Regex Examples:
- Notes

Specifying Labels for a Job's Build Results

To specify labels for a Job's build results:

1. Navigate to a Job's configuration pages, as described on Editing a Job.
2. Click the 'Post Actions' sub-tab (see screenshot below).
3. In the 'Regex Pattern' field, you can either:
   - Specify a regular expression to match content in the log files of this Job's builds. Labels will be applied to a build of this Job if this regular expression matches content in the build's log files (see the examples below).
   - For more information about regular expressions, please refer to the Java documentation on regular expression constructs.
3. Leave this field blank to label every build of this Job.

4. 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.

5. Click the ‘Save’ button to save your changes.

Regex Examples:

A simple regex example:

```regex
'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’

A 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, the following 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.

- Enter the following into the ‘Regex Pattern’ field:

  ```regex
  (PERFORMANCE_IMPROVED|PERFORMANCE_DETERIORATED)
  ```

- Enter the following into the ‘Labels’ field:

  ```regex
  \1
  ```

Notes

Related Topics

Editing a Job

Renaming a Job or changing the Job’s Description

To rename a Job or change the Job’s description:
1. Navigate to the desired Job's configuration pages, as described on Editing a Job.
2. Update the fields, as desired. Note that a Job's Job Key is not editable.
3. Click the 'Save' button.

![Screenshot above: Editing a Job's details](image)

**Configuring a Job’s Requirements**

This page describes how to configure the capability requirements for a Job.

The page REV31 - requirement does not exist.

There are four types of capabilities in Bamboo:

- **Executable capability** — Every Task has one 'Executable' capability requirement. A Job inherits the 'Executable' capability requirements of its Tasks.
- **JDK capability** — A Task has either one 'JDK' capability requirement or none, depending on the Task's 'Executable'. A Job inherits the 'JDK' capability requirements of its Tasks.
- **Custom capability** — A Job can have multiple 'custom' capability requirements (or none). Before you can specify a custom capability in your Job, you must first define the custom capabilities in your Bamboo system (see Configuring a new Custom Capability). Custom capabilities can be used to control which Jobs will be built by a particular agent. For example, if the builds for a particular Job 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 Job.
- **Perforce capability** — A Job will have a 'Perforce' capability requirement automatically specified, if Perforce was selected as the source repository.

**On this page:**

- Specifying Extra Requirements for a Job
- Viewing Current Capable Agents
- Notes
Specifying Extra Requirements for a Job

A Job's will inherit the requirements of its Tasks by default. However, you can specify extra requirements for the Job, in addition to the Task requirements.

To specify extra requirements for a Job:

1. Navigate to the desired Job's configuration pages, as described on Editing a Job.
2. Click the 'Requirements' tab to display the Job's Requirements settings (see screenshot above). This page shows a list of all the Job's current requirements and the number of 'Matching Agents' and 'Matching Elastic Images' (i.e. agents/elastic images which meet the Job's requirements and can run a build for this Job). See Viewing Current Capable Agents below for more information.
3. If you had previously set up the capability (executable, JDK, custom), you can select it from the 'Requirement' dropdown in the 'Add Extra Requirement' section. If you are setting up a new custom requirement, select 'New custom requirement' instead.
4. Fill out the fields for the requirement:
5. 'Key' (new custom requirement only) — enter the key of the new capability.
6. Select the condition for the requirement from the dropdown:
   - 'exists' — this Job can be built by any agent that has the capability with the same Key.
   - 'equals' — this will display a text field for you to enter an exact value. This Job can be built by any agent that has the 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://download.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html#sum). This Job can be built by any agent that has a capability with the same Key, and the value matches this regular expression.
7. Click the 'Add' button to add your new custom capability to the Job's list of requirements. The list of 'Matching Agents' and 'Matching Elastic Images' displayed will be updated, as the plan can now only be built by agents which have a capability that meets the new custom requirement you have specified.

Viewing Current Capable Agents

From a Job's Requirements page (above), you can view which agents and elastic images are currently capable of building the Job.

To view details about agents/elastic images that are currently capable of building your Job:

1. On the Job's Requirements page (described above), click the name of the requirement in the table (e.g. 'Maven 2').
2. The summary page for the capability will be displayed, showing the Agents and Elastic Images that have the capability. See Viewing a Capability's Agents and Jobs for more information.
Configuring Miscellaneous Settings for a Job

For each Job of a Plan, you can optionally specify a number of miscellaneous settings including:

- Build hanging detection
- NCover output
- Clover code coverage

To configure the miscellaneous settings for a Job:

1. Navigate to the desired Job, as described in Editing a Job.
2. Edit the desired settings as follows:

   - `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). If selected, the following options will appear:
     - `Build Time Multiplier` — This setting is used to calculate the ‘Expected Build Time’ for the build (i.e. ‘Expected Build Time’ = ‘Build Time Multiplier’ multiplied by ‘Average Build Time’).
     - `Log Quiet Time` — This is the amount of time since Bamboo last recorded an entry in the build log for a build.
     - `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 Job Build Queue Timeout Event).
   - `NCover output will be produced` — Do not select this option. NCover is a code coverage tool that supports .NET projects.
   - `Use Clover to collect Code Coverage for this build` — Select this check box if:
     - This Job will be building a Java-/Groovy-based project using a builder such as Ant, Maven or Grails.
     - 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).
   - `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.
     - 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 Job build's root directory, for example:

     ```
     target/site/clover/clover.xml
     ```
     
     Do not specify an absolute path.

3. Click ‘Save’ to save your changes.
Bamboo allows you to disable or delete Plans/Jobs that you don't want to be built:

- **Disabling a Plan/Job** prevents it from being built. Disabling a Plan prevents Bamboo from submitting that Plan to the build queue. Disabling a Job prevents Bamboo from building that single Job within a Plan, allowing the rest of the Plan's Jobs to be built. You can re-enable the Plan/Job, if you want to build it again.

  For example, if a Plan's/Job's latest build is broken and cannot be fixed quickly, you may want to disable it to temporarily stop the Plan/Job from being built.

- **Deleting a Plan/Job** removes it completely from your Bamboo system. You will need to recreate the new Plan/Job from scratch, if you want to build it again.

  For example, if a Plan/Job is no longer relevant, you may want to delete it.

---

**Disabling a Job**

To disable a Plan's Job:

1. Access the Job you want to disable. To do this:
   a. Click 'Home' to go to the Dashboard.
   b. Click the 'All Plans' tab.
   c. Locate the Job's Plan in the list and click the Plan's name (i.e. before the first '>'). The Plan's **Plan Summary** page will be displayed.
   d. In the **Stages** section of this page, click the name of the Job you wish to access. The Job's **Job Summary** page will be displayed.

   ![Image](image.png)

   If necessary, expand the appropriate Stage to access your Job.

2. Click the 'Actions' menu (at the right of this page) and select the 'Disable Job' menu option.

---

**Deleting a Job**

Deleting a Job deletes everything related to that Job, including the Job's configuration, build results, artifacts, labels and comments. However, everything else related to the Job's Plan and this Plan's other Jobs are retained by Bamboo.

Before you begin:

- The 'Admin' global permission is required to delete a Job.
- A Job that is currently being built cannot be deleted. If you need to delete such a Job, stop the Plan's build first. Refer to Stopping an Active Job Build for more information.
- If you need to keep a permanent record of the Job build results for your Job, see Exporting Data for Backup.

To delete a Job:

1. Access the Job you want to disable. To do this:
   a. Click 'Home' to go to the Dashboard.
   b. Click the 'All Plans' tab.
   c. Locate the Job's Plan in the list and click the Plan's name (i.e. before the first '>'). The Plan's **Plan Summary** page will be displayed.
   d. In the **Stages** section of this page, click the name of the Job you wish to access. The Job's **Job Summary** page will be displayed.

   ![Image](image.png)

   If necessary, expand the appropriate Stage to access your Job.

2. Click the 'Actions' menu (at the right of this page) and select the 'Delete Job' menu option.

---

**Notes**

Related Topics

- Creating a Job
- Editing a Job
- Disabling or Deleting a Plan

**Deleting a Job's Current Working Files**

If you only run a single Bamboo server (i.e. with no remote or elastic agents) and you:

- need to ensure that a Plan's Job cleanly checks out its source code when Bamboo next executes a build of that Plan and
- do not to use the Force Clean Build option when specifying/configuring the source repository for this Job,

Simply delete the current working files for that Job to ensure its source code is cleanly checked out.
Before you begin:

- Only people with the 'Admin' global permission or the 'Admin' plan permission can delete current working files.

To delete a Job's current working files:

1. Access the Job whose current working files you want to delete. To do this:
   a. Click 'Home' to go to the Dashboard.
   b. Click the 'All Plans' tab.
   c. Locate the Job's Plan in the list and click the Plan's name (i.e. before the first '>'). The Plan's Plan Summary page will be displayed.
   d. In the Stages section of this page, click the name of the Job you wish to access. The Job's Job Summary page will be displayed.
      ✔️ If necessary, expand the appropriate Stage to access your Job.
2. Click the 'Files' tab and a list of the Job's working files will be displayed.
   📁 The Files tab is only available if:
   - the current working files resulting from that Job's previous build reside on the Bamboo server (not a remote/elastic agent) and
   - working files exist in this directory.
3. At the end of this page (scroll down if necessary), click the Delete all build files link.

Notes

Related Topics

Specifying the Source Repository for a Job
Disabling or Deleting a Job
Deleting the Results of a Plan Build

Stopping an Active Job Build

The instructions on this page describe how to stop an Job build that is running.

**On this page:**

- Stopping an Active Job Build
- Notes

**Stopping an Active Job Build**

To prevent Bamboo submitting a Plan to the build queue, refer to Disabling or Deleting a Job.

To stop an active Job build:

1. Click 'Home' to go to the Dashboard and access the current Plan build via the following tabs:
   - Using the Current Activity tab:
     - Locate the relevant Plan in the list and click the Plan's name in the Building section of this page to display the current Plan build's Summary page, or
   - Using the All Plans tab:
     a. Locate the relevant Plan in the list and click the Plan's name to display the Plan's Plan Summary page.
     b. In the Current Activity section, click the Plan's current build number link to display the current Plan build's Summary page.
2. In the Stages section, click the 'Stop Build' icon next to the active Job build you want to stop.
   
   To start a building a Plan manually, see Triggering a Plan Build Manually.

Notes

- If your Bamboo server runs on Windows, it may only be possible to stop an active build by going to the Windows Task Manager and ending the relevant processes.

Related Topics

Triggering a Plan Build Manually
Disabling or Deleting a Plan
Disabling or Deleting a Job
Stopping an Active Plan Build

**Configuring Tasks**

A Task is an operation that is run on a Bamboo working directory using an executable. Tasks are run sequentially within a single Job. Once a Task is defined in the Bamboo system, it can then be specified in Jobs by a Bamboo administrator. A Job can be configured to execute a number of Tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers or execute a Script.
Creating a Task for a Job

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

To create a Task for a Job:

1. Navigate to the Tasks configuration for the desired Job, i.e.:
   - navigate to the 'Tasks' tab when configuring an existing Job, or
   - create a Plan (you will be configuring Tasks for the default Job).
2. Click 'Add Task'.
3. Click the desired Task type on the 'Task Types' popup.
4. Update the values to configure the Task as desired. The fields displayed will vary depending on the executable chosen. See the following pages for further details:
   - Configuring a Builder Task
   - Configuring a Test Task
   - Using Global, Plan or Build-specific Variables
5. Click 'Save' to save your changes.

Ordering Tasks within a Job

Tasks can be designated as Build Tasks or Final Tasks within a Job:

- Build Tasks will run sequentially in the order specified in the Job. If a Build Task fails, all subsequent tests will not be executed.
- Final Tasks will run sequentially, once the build Tasks have completed. Final Tasks will always be executed, regardless of whether any Build Tasks or other Final Tasks fail.

To create a Task for a Job:

1. Navigate to the Tasks for the desired Job.
2. Drag and drop the Tasks into the desired order in the table on the left. If you want to change a Build Task to a Final Task or vice versa, drag and drop it under the desired header in the table. Your changes will be saved immediately.
About the ‘Compatibility Task’ — The ‘Compatibility Task’ is created by the Bamboo when upgrading from Bamboo 3.0 or earlier and Bamboo cannot match a builder to a Task. This may occur if you are using a builder enabled by custom plugin. For more information, see the Bamboo 3.1 Upgrade Guide.

Adding new executables — At least one executable is configured automatically after installing Bamboo. You can add more executables of different types as described in Configuring a New Executable.

Adding new JDKs — At least one JDK is configured automatically after installing Bamboo. You can add more JDKs as described in Configuring a new JDK.

Related Topics

Creating a Plan
Creating a Job
Editing a Job

Configuring a Builder Task

Builder Tasks execute a Job’s builds using build tools. Build tools are used to automate build processes. The range of features for a build tool differ significantly from one build tool to the next. Generally, build tools compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Build tools typically have their own form of scripting language and the ability to manage dependencies correctly.

See the following pages for more information on configuring specific builder Tasks:

• Ant
Notes

Related Topics

Configuring a Test Task

**Ant**

This page describes how to configure an Ant executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

---

**On this page:**

- Configuring an Ant Executable for a Task
- Notes

---

### Configuring an Ant Executable for a Task

To configure an Ant executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Ant Task, or click 'Add Task' and then 'Ant' if creating a new Task.
3. In the 'Executable' dropdown, select the Ant executable that you wish to configure for this Task (e.g. "Ant"). The executable that you select will become one of the Task's capabilities (and hence, one of the Job's requirements). For details, please see Configuring a Job's Requirements.
4. The screen will refresh to display the fields specific to Ant:
   - **Task Description** — Enter a description of the Task, for display in Bamboo.
   - **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'). Multiple Ant targets can be specified with a space-delimited list. 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 Job's capability requirements. For details please see Configuring a Job's Requirements.
5. If required, update the system environment variables and working directory settings:
To configure a custom command executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Command Task, or click 'Add Task' and then 'Command' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'Executable' — Select the custom command executable that you wish to configure for this Task (e.g. "Bash"). The executable that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, please see Configuring a Job's Requirements.
   - 'Argument' (Optional) — 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, Plan or Build-specific Variables).
4. If required, update the system environment variables and working directory settings:
   - '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, Plan or Build-specific Variables).
   - 'System Environment Variables' field.
   - 'Working Sub Directory' (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.

6. Specify the following general build parameters:

   - 'The build will produce test results' — Select this check box if you want Bamboo to gather test results data for each build result. If selected, the following options will appear:
     - 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' radio button (Only available with Maven and Grails builders) — Select this option if Bamboo should look in the Builder's standard test results directory.
     - 'Specify custom results directories' radio button (Only available with Maven and Grails builders) — Select this option if the Builder will place generated test results in an alternative directory. If selected, the following will appear:
       - 'Specify custom results directories' field — 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 Job build's root directory — do not specify an absolute path like /home/bamboouser/bamboo-home/xml-data/build-dir/JOB_KEY/.
       - For Jobs that use CVS, the Job build's root directory is <bamboo-home>/xml-data/build-dir/JOB_KEY/<cvs-module>

7. Click the 'Save' button to save your changes.

**Notes**

- Configuring Tasks
- Editing a Job

**Custom Command Executable**

This page describes how to configure a custom command (e.g. Bash) executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.
• '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, Plan 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 executable, thus you don’t need to specify them in the 'System Environment Variables' field.

• 'Working Sub Directory' (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.

5. Specify the following general build parameters:

• 'The build will produce test results' — Select this check box if you want Bamboo to gather test results data for each build result. If selected, the following options will appear:

  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' radio button (Only available with Maven and Grails builders) — Select this option if Bamboo should look in the Builder's standard test results directory.

  • 'Specify custom results directories' radio button (Only available with Maven and Grails builders) — Select this option if the Builder will place generated test results in an alternative directory. If selected, the following will appear:

    • 'Specify custom results directories' field — Type the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as **/*.xml. Please specify file path relative to your Job build's root directory — do not specify an absolute path like /home/bamboouser/bamboo-home/xml-data/build-dir/JOB_KEY/.

    For Jobs that use CVS, the Job build’s root directory is <bamboo-home>/xml-data/build-dir/JOB_KEY/<cvs-module>

6. Click the 'Save' button to save your changes.

Notes
Related Topics
Configuring Tasks
Creating a Job

Grails

This page describes how to configure a Grails executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

On this page:

• Configuring a Grails Executable for a Task
• Notes

Configuring a Grails Executable for a Task

Before you begin:

• Bamboo supports Grails versions 1.2.x and 1.3.x.

To configure an executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Grails Task, or click 'Add Task' and then 'Grails' if creating a new Task.
3. In the 'Executable' dropdown, select the Grails executable that you wish to configure for this Task (e.g. "Grails"). The executable that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, see Configuring a Job's Requirements.
4. The screen will refresh to display the settings specific to Grails:
• ‘Grails Commands’ — Specify the Grails command that you want Bamboo to execute each time the source code changes. For example, the following example will run the Grails command 'clean' followed by the Grails command 'test-app':

```
clean
test-app
```

• ‘Build JDK’ — Choose a JDK from the list. The JDK that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, please see Configuring a Job’s Requirements.

5. If required, update the system environment variables and working directory settings:

   • ‘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, Plan 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 executable, thus you don't need to specify them in the ‘System Environment Variables’ field.

   • ‘Working Sub Directory’ (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.

6. Specify the following general build parameters:

   • The build will produce test results’ — Select this check box if you want Bamboo to gather test results data for each build result. If selected, the following options will appear:

   ```
   ![image]
   ```

   `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' radio button (Only available with Maven and Grails builders) — Select this option if Bamboo should look in the Builder's standard test results directory.

   • 'Specify custom results directories' radio button (Only available with Maven and Grails builders) — Select this option if Bamboo should place generated test results in an alternative directory. If selected, the following will appear:

   ```
   ![image]
   ```

   For Jobs that use CVS, the Job build's root directory is `<bamboo-home>/xml-data/build-dir/JOB_KEY/’

7. Click the 'Save' button to save your changes.

Notes

• **About Grails Commands** — Please refer to the Grails Command Line Reference documentation for more details on Grails commands. In particular, note the following:

  • You can use `'-D'` to define one or more JVM parameters, e.g.: `-Djava.awt.headless=true` will pass the parameter 'java.awt.headless' with a value of 'true'

  • You can include variables (see Using Global, Plan or Build-specific Variables).

  • Please be aware that unlike Ant or Maven, Grails generally does not permit the execution of a series of commands on the same line. Therefore, each Grails command should be entered on a new line in the 'Grails Commands' text box. For example, the following Grails commands:

   ```
   clean
   set-version $(bamboo.buildKey)
   test-app
   war
   ```

   will set the version of your Grails application to that of your Bamboo build key value, run unit tests within the Grails application and then create a JEE Web Application Archive (WAR) file from the Grails application.

• **Specifying Generic JDK Requirements** — You can specify generic JDK capabilities as requirements for your Task. You will need to configure a generic JDK capability first:

  • 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 ‘JDK Label’ = ‘JDK’ and ‘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 ‘JDK Label’ = ‘JDK 1.5’ and ‘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 ‘JDK Label’ = ‘JDK 1.5.0_07’ and ‘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.

Related Topics
Configuring Tasks
Editing a Job
Configuring an Agent-specific JDK Capability

Maven

This page describes how to configure a Maven executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

To configure a Maven executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Maven Task, or click 'Add Task' and then a Maven option (e.g. Maven 2.x) if creating a new Task.
3. In the 'Executable' dropdown, select the Maven executable that you wish to configure for this Task (e.g. "Maven 2"). The executable that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, please see Configuring a Job's Requirements.
4. The screen will refresh to display the settings specific to Maven:
   - 'Project File' (Optional, Maven 2.x or later only) — Enter the path to your Maven project file, relative to the working sub directory specified (below). If you do not specify this, Maven will use the pom.xml in the root of the working sub directory.
   - 'Goal' — Specify the Maven goal that you want Bamboo to execute each time the source code changes. For example, clean test will run the Maven goal 'clean' followed by the Maven goal 'test'.
   - 'Build JDK' — Choose a JDK from the list. The JDK that you select will become one of the Job's capability requirements. For details please see Configuring a Job's Requirements.
5. If required, update the system environment variables and working directory settings:
   - '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, Plan 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 executable, thus you don’t need to specify them in the ' System Environment Variables' field.
   - 'Working Sub Directory' (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.

6. Specify the following general build parameters:
   - The build will produce test results — Select this check box if you want Bamboo to gather test results data for each build result. If selected, the following will appear:
     - 'Look in the standard test results directory' radio button (Only available with Maven and Grails builders) — Select this option if Bamboo should look in the Builder's standard test results directory.
     - 'Specify custom results directories' radio button (Only available with Maven and Grails builders) — Select this option if the Builder will place generated test results in an alternative directory. If selected, the following will appear:
     - 'Specify custom results directories' field — 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 Job build's root directory — do not specify an absolute path like /home/bamboouser/bamboo-home/xml-data/build-dir/JOBDATA/KEY/
   - For Jobs that use CVS, the Job build's root directory is <bamboo-home>/xml-data/build-dir/JOBDATA/KEY/<cvs-module>

7. Click the 'Save' button to save your changes.
Notes

- **About the Maven Goal:**
  - You can use `-D` to define one or more JVM parameters. For example, `-Djava.awt.headless=true` will pass the parameter `java.awt.headless` with a value of `true`.
  - Multiple maven goals can be specified with a space-delimited list.
  - You can include variables (see Using Global or Build-specific Variables).
- **Specifying Generic JDK Requirements** — You can specify generic JDK capabilities as requirements for your Task. You will need to configure a generic JDK capability first:
  - **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 `JDK Label` = 'JDK' and `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 `JDK Label` = 'JDK 1.5' and `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 `JDK Label` = 'JDK 1.5.0_07' and `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.

Related Topics
- Configuring Tasks
- Editing a Job
- Configuring an Agent-specific JDK Capability

**MSBuild**

This page describes how to configure an MSBuild executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

On this page:
- Configuring an MSBuild Executable for a Task
- Notes

### Configuring an MSBuild Executable for a Task

To configure an MSBuild executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired MSBuild Task, or click 'Add Task' and then 'MSBuild' if creating a new Task.
3. Update the Task settings:
   - **Task Description** — Enter a description of the Task, for display in Bamboo.
   - **Project File** — Type the Solution, Project File or MSBuild project to execute when this Job Builds, e.g. ExampleSolution.sln. You can include variables (see Using Global, Plan or Build-specific Variables).
   - **Options** — Specify the MSBuild command line options that you want to include. You can also include variables (see Using Global, Plan or Build-specific Variables).
4. If required, update the system environment variables and working directory settings:
   - **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, Plan 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 executable, thus you don't need to specify them in the 'System Environment Variables' field.
5. Click the 'Save' button to save your changes.
Notes

- You cannot use Clover to collect code coverage for MSBuild builds, as Clover only supports builders of Java/Groovy-based projects, such as Ant, Maven or Grails.

Related Topics

Configuring Tasks
Editing a Job

**NAnt**

This page describes how to configure an NAnt executable for a Task.

When creating a new **Job** or configuring an existing one, you need to specify the **Tasks** that will execute the Job's builds. You must specify an **executable** for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new **Plan**, you can configure the Tasks for the Plan's default Job.

An **executable** is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as **artifacts** in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as **capabilities** in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

---

**Configuring a NAnt Executable for a Task**

To configure a NAnt executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired NAnt Task, or click 'Add Task' and then 'NAnt' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'Build File' — Type the relevant filename (e.g. default.build). You can include variables (see Using Global, Plan 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, Plan or Build-specific Variables).
   - 'Options' — Specify the NAnt command line options that you want to include. You can also include variables (see Using Global, Plan or Build-specific Variables).
4. If required, update the system environment variables and working directory settings:
   - '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, Plan 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 executable, thus you don’t need to specify them in the 'System Environment Variables' field.
   - 'Working Sub Directory' (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.
5. Click the 'Save' button to save your changes.

Notes

- You cannot use Clover to collect code coverage for NAnt builds, as Clover only supports builders of Java/Groovy-based projects, such as Ant, Maven or Grails.

Related Topics

Configuring Tasks
Editing a Job

**Script Builder**

This page describes how to configure a script executable for a Task.
Configuring a Script Executable for a Task

To configure a script builder for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name desired Task or click ‘Add Task’ if creating a new Task.
3. In the ‘Executable’ dropdown, select the script executable that you wish to configure for this plan (e.g. "Script1"). The executable that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, please see Configuring a Job's Requirements.
4. The screen will refresh to display the settings specific to script executables:
   - ‘Script location’ — Select the location of the script file.
     - If you selected ‘File’, enter the location of the file in the ‘Script file’ field. This can be either relative to the repository root of the plan, or absolute. You can include variables (see Using Global, Plan or Build-specific Variables).
     - If you selected ‘Inline’, enter the script in the ‘Script body’ field.
   - ‘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, Plan or Build-specific Variables).
5. If required, update the system environment variables and working directory settings:
   - ‘Environment Variables’ (Optional) — Specify any additional* 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, Plan 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 executable, thus you don’t need to specify them in the ‘System Environment Variables’ field.
   - ‘Working Sub Directory’ (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.
6. Click the ‘Save’ button to save your changes.

Notes
Related Topics
Configuring Tasks
Editing a Job

Visual Studio

This page describes how to configure a Visual Studio (devenv.exe) executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.
Configuring a Visual Studio Executable for a Task

To configure a Visual Studio executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Visual Studio Task, or click ‘Add Task’ and then ‘Visual Studio’ if creating a new Task.
3. Update the Task settings:
   - ‘Task Description’ — Enter a description of the Task, for display in Bamboo.
   - ‘Solution’ — Enter 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, Plan or Build-specific Variables).
   - ‘Options’ — Specify any Visual Studio command-line options that you want to include (e.g. /build Debug). You can also include variables (see Using Global, Plan or Build-specific Variables).
   - ‘Platform’ — Select the platform toolset required to compile your Solution. This is provided as an argument to Vcvarsall.bat (see this MSDN article for more details).
3. If required, update the system environment variables and working directory settings:
   - ‘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, Plan or Build-specific Variables).
   - ‘Working Sub Directory’ (Optional) — A subdirectory relative to the Job build’s root directory where Bamboo will execute your specified executable’s commands. The Job build’s root directory contains everything checked out from your Job’s configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.
4. Click the ‘Save’ button to save your changes.

Notes

Related Topics

Configuring Tasks
Editing a Job

Configuring a Test Task

Test Tasks run tests and/or parse test data, using a particular testing framework.

Please note,

- Java Builder Tasks in Bamboo (e.g. Maven) parse test information as part of the Task. You do not need to configure a Test Task, if you have specified that test results will be produced as part of the Builder Task. However, you can configure a Builder Task to not produce test results and use a Test Task to parse the test data instead. For example, you may want to set up one JUnit Parser Task to parse test data for a number of Maven Tasks after they have executed.
- .Net Builder Tasks in Bamboo (e.g. NAnt) do not parse test information as part of the Task. You must configure a Test Task (e.g. NUnit Parser), if you want test results from the Builder Task to be parsed.

See the following pages for more information on configuring specific Test Tasks:

- JUnit Parser
- MBUnit Parser
- MSTest Parser
- MSTest Runner
- NUnit Parser
- PHPUnit

Notes

Related Topics

Configuring a Builder Task

JUnit Parser

This page describes how to configure a JUnit Parser executable for a Task.
When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

### Configuring a JUnit Parser Executable for a Task

Before you begin:

- Java Builder Tasks in Bamboo (e.g. Maven) parse test information as part of the Task. You do not need to configure a Test Task, if you have specified that test results will be produced as part of the Builder Task.

To configure a JUnit executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired JUnit Parser Task, or click 'Add Task' and then 'JUnit Parser' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'Specify custom results directories' — Enter the name of the test results directory (or multiple directories, separated by commas). You can also use Ant-style patterns such as */*.xml. Please specify file path relative to your Job build's root directory. Do not specify an absolute path.
     For Jobs that use CVS, the Job build's root directory is `<bamboo-home>/xml-data/build-dir/JOB_KEY/<cvs-module>`.  
4. Click the 'Save' button to save your changes.

### Notes

#### Related Topics

Configuring Tasks
Editing a Job
Configuring a Test Task

**MBUnit Parser**

This page describes how to configure a MBUnit Parser executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

### Configuring a MBUnit Parser Executable for a Task

Before you begin:

- .Net Builder Tasks in Bamboo (e.g. NAnt) do not parse test information as part of the Task. You must configure a Test Task (e.g. MBUnit Parser), if you want test results from the Builder Task to be parsed.
To configure a MBUnit Parser executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired MBUnit Parser Task, or click 'Add Task' and then 'MBUnit Parser' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'MBUnit Test Results File' — Enter the name of the test results file. The test file must be in MBUnit XML format. For more information on MBUnit, see http://www.mbunit.com/.
4. Click the 'Save' button to save your changes.

Notes

Related Topics

Configuring Tasks
Editing a Job
Configuring a Test Task

MSTest Parser

This page describes how to configure a MSTest Parser executable for a Task.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job’s builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

On this page:

- Configuring a MSTest Parser Executable for a Task
- Notes

Configuring a MSTest Parser Executable for a Task

Before you begin:

- .Net Builder Tasks in Bamboo (e.g. NAnt) do not parse test information as part of the Task. You must configure a Test Task (e.g. MSTest Parser), if you want test results from the Builder Task to be parsed.

To configure a MSTest Parser executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired MSTest Parser Task, or click 'Add Task' and then 'MSTest Parser' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'MBUnit Test Results File' — Enter the name of the test results file. The test file must be in MSTest format. For more information on MSTest, see this MSDN page.
4. Click the 'Save' button to save your changes.

Notes

Related Topics

Configuring Tasks
Editing a Job
Configuring a Test Task

MSTest Runner

This page describes how to configure a MSTest Runner executable for a Task. The MSTest Runner executable runs and parses tests for .Net builds.

When creating a new Job or configuring an existing one, you need to specify the Tasks that will execute the Job's builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan's default Job.
An **executable** is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as **artifacts** in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as **capabilities** in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

### Configuring a MSTest Runner Executable for a Task

**Before you begin:**

- .Net Builder Tasks in Bamboo (e.g. NAnt) do not parse test information as part of the Task. You must configure a Test Task (e.g. MSTest Parser), if you want test results from the Builder Task to be parsed.

**To configure a MSTest Runner executable for a Task:**

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired MSTest Runner Task, or click ‘Add Task’ and then ‘MSTest Runner’ if creating a new Task.
3. Update the Task settings:
   - **‘Task Description’** — Enter a description of the Task, for display in Bamboo.
   - **‘Executable’** — In the ‘Executable’ dropdown, select the MSTest Runner executable that you wish to configure for this Task (e.g. “Visual Studio 2010”). The executable that you select will become one of the Task’s capability requirements (and hence, one of the Job’s requirements). For details, please see Configuring a Job’s Requirements.
   - **‘Container’** — Specify the test container, i.e. the file that contains the tests you want to run. For example, tests.dll. The value of this field is passed to the MSTest.exe as the /testcontainer parameter. See MSTest.exe Command-Line Options (MSDN).
   - **‘Result Filename’** — Specify the file that you want to save the test results to. For example, testResults.trx. The value of this field is passed to the MSTest.exe as the /resultsfile parameter. See MSTest.exe Command-Line Options (MSDN).
   - **‘Run Configuration’** — Specify the run configuration that you want to use. For example, localtestrun.Testrunconfig. The value of this field is passed to the MSTest.exe as the /runconfig parameter. See MSTest.exe Command-Line Options (MSDN).

4. Click the ‘Save’ button to save your changes.

**Notes**

### Related Topics

- Configuring Tasks
- Editing a Job
- Configuring a Test Task

#### NUnit Parser

This page describes how to configure a NUnit Parser executable for a Task.

When creating a new **Job** or configuring an existing one, you need to specify the **Tasks** that will execute the Job’s builds. You must specify an executable for each Task. If you specify an Ant, Grails or Maven executable, you will also need to choose a JDK.

When creating a new Plan, you can configure the Tasks for the Plan’s default Job.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as **artifacts** in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as **capabilities** in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.
.Net Builder Tasks in Bamboo (e.g. NAnt) do not parse test information as part of the Task. You must configure a Test Task (e.g. MSTest Parser), if you want test results from the Builder Task to be parsed.

To configure a NUnit Parser executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired NUnit Parser Task, or click 'Add Task' and then 'NUnit Parser' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'NUnit Test Results File/Directory' — Enter the name of the test results file/directory. The test files must be in NUnit XML format. For more information on NUnit, see http://www.nunit.org/.
4. Click the 'Save' button to save your changes.

Notes

Related Topics

Configuring Tasks
Editing a Job
Configuring a Test Task

PHPUnit

This page describes how to configure a PHPUnit executable for a Task.

Before you begin:

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

To configure a PHPUnit executable for a Task:

1. Navigate to the Task configuration for the Job (this will be the default Job if creating a new Plan).
2. Click the name of the desired Command Task, or click 'Add Task' and then 'PHPUnit' if creating a new Task.
3. Update the Task settings:
   - 'Task Description' — Enter a description of the Task, for display in Bamboo.
   - 'Executable' — Select the PHPUnit executable that you wish to configure for this Task (e.g. "PHPUnit 3.3.x"). The executable that you select will become one of the Task's capability requirements (and hence, one of the Job's requirements). For details, please see Configuring a Job's Requirements.
   - 'Arguments' — Type the name of the directory/files that will be analysed recursively by PHPUnit. The default value is "." (i.e. the working subdirectory, if specified). You must specify at least one argument.
4. If required, update the system environment variables and working directory settings:
   - '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, Plan or Build-specific Variables).
   - 'Working Sub Directory' (Optional) — A subdirectory relative to the Job build's root directory where Bamboo will execute your specified executable's commands. The Job build's root directory contains everything checked out from your Job's configured source repository. If you leave this field blank, Bamboo will look for the build files in the build root directory (which is assumed to be the Working Directory, as described in Locating Important Directories and Files). This option is useful, if your Task has a build script in a subdirectory and the executable needs to be run from within that subdirectory, in which case, you would type the name of that subdirectory in this field.
5. Update the following build parameters:
5. Click the 'Save' button to save your changes.

### Notes

**Related Topics**

- Configuring Tasks
- Editing a Job

## Using Global, Plan or Build-specific Variables

Variables can be used to set static values that are used when building Plans in Bamboo.

- **Global variables** are defined across your entire Bamboo instance, and have the same (static) value for every plan that is built by Bamboo. They are defined via the Global Variables page.
- **Plan variables** are similar to global variables, but are defined for specific Plans. Plan variables override global variables with the same name. You can also override a Plan variable for a build, if you have triggered the build manually.
- **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).
- **System variables** also apply across your entire Bamboo instance and inherit their values from system or environment variables of the same name.

### Specifying Global, Plan, Build-specific or System Variables

#### Global Variables

See [Defining Global Variables](#) for information on defining global variables.

The usage format for all global variables is:

```
{$bamboo.globalVarName}
```

#### Plan Variables

See [Defining Plan Variables](#) for information on defining Plan variables. You can override a Plan variable for a build, if you have triggered the build manually. For details, see [Triggering a Plan Build Manually](#).

The usage format for all Plan variables is:

```
{$bamboo.varName}
```

#### Build-Specific Variables

The following build-specific variables are also 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>buildPlanName</td>
<td>Bamboo property</td>
<td>The Bamboo plan name e.g. Some Project name - Some plan name</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>buildTimeStamp</td>
<td>The time when build was started in ISO 8601 format e.g. 2010-01-01T01:00:00+01:00</td>
<td></td>
</tr>
<tr>
<td>buildForceCleanCheckout</td>
<td>Whether the &quot;Force Clean Build&quot; option was used, values:true/false</td>
<td></td>
</tr>
<tr>
<td>build.working.directory</td>
<td>The working directory that the build is being executed on</td>
<td></td>
</tr>
<tr>
<td>custom.svn.revision.number</td>
<td>Plugin (For Subversion only) The revision number</td>
<td></td>
</tr>
<tr>
<td>custom.svn.lastchange.revision.number</td>
<td>Plugin (For Subversion only) The last changed revision number</td>
<td></td>
</tr>
<tr>
<td>custom.svn.username</td>
<td>Plugin (For Subversion only) User name used for repository authentication</td>
<td></td>
</tr>
<tr>
<td>custom.cvs.last.update.time</td>
<td>Plugin (For CVS only) The last updated timestamp</td>
<td></td>
</tr>
<tr>
<td>custom.cvs.last.update.time.label</td>
<td>Plugin (For CVS only) The last updated timestamp to be used as a label for post build result labelling. The spaces in the cvs version string are replaced with '_'</td>
<td></td>
</tr>
<tr>
<td>custom.p4.revision.number</td>
<td>Plugin (For Perforce only) The change set number</td>
<td></td>
</tr>
<tr>
<td>custom.p4.username</td>
<td>Plugin (For Perforce only) User name used for repository authentication</td>
<td></td>
</tr>
<tr>
<td>custom.p4.port</td>
<td>Plugin (For Perforce only) Port used for repository communication</td>
<td></td>
</tr>
<tr>
<td>custom.p4.client</td>
<td>Plugin (For Perforce only) Client used for repository communication</td>
<td></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 build-specific variables is:

```
${bamboo.varName}
```

**System Variables**

The usage format for all system variables is:

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

For example, 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.

**Using 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>Goal (for Maven builders only)</td>
<td>Global variables</td>
</tr>
<tr>
<td>Build File (for Ant and NAnt builders only)</td>
<td>Build-specific variables</td>
</tr>
<tr>
<td>Target (for Ant and NAnt builders only)</td>
<td>System variables</td>
</tr>
<tr>
<td>Options (for NAnt builders only)</td>
<td>Global variables</td>
</tr>
<tr>
<td>Script (for Scripts only)</td>
<td>Build-specific variables</td>
</tr>
<tr>
<td></td>
<td>System variables</td>
</tr>
</tbody>
</table>
Examples of Variables Usage

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>  
<packaging>jar</packaging>  
<version>1.1.$(env.bambooBuildNumber)-SNAPSHOT</version>  
...  
```

You can then specify the following in the 'Goal' field of your build plan:

```shell
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`.

Ant Example

You can then specify the following in the 'Target' field of your build plan:

```shell
-f build.xml -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 Ant build to use.

Specifying 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:

```shell
$(bamboo.capability.<capability_key>)
```

For example,
Custom

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

JDK

```yaml
${bamboo.capability.system.jdk.<jdk_label>}
```

Builder

```yaml
${bamboo.capability.system.builder.<builder_type>.<builder_label>}
e.g. ${bamboo.capability.system.builder.maven.Maven1}
```

Perforce

```yaml
${bamboo.capability.system.p4Executable}
```

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.

Using Capabilities

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,

```text
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:

   ```yaml
   ${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.
```

Notes

Related Topics

Defining Global Variables
Defining Plan Variables
Triggering a Plan Build Manually
Configuring Plugins

**Defining Global Variables**

When configuring a plan, you may want to specify variables to be used in the build process. For details on how variables are used, see Using Global, Plan 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. If you want to define a variable for a specific Plan rather than across all Plans, define a Plan variable as described in Defining Plan Variables.

Global variables can be accessed by using [${bamboo.globalVarName}]. Global variables can also be overridden at runtime when running a manual build. For more information, see Triggering a Plan Build Manually.
Defining a Global Variable

To access the Global Variables page:

1. Click the ‘Administration’ link in the top navigation bar.
2. In the left navigation column under the ‘Build Resources’ section, click ‘Global Variables’.
3. Add, update or delete the Global variables, as desired:
   - Click ‘Add’ to add a new variable once you have entered the key and value for it.
   - Updates to existing rows will be saved as you move between cells in the table.
   - Click the trash can to delete a variable.

### Screenshot above: Global Variables

#### Notes

Related Topics

- Using Global, Plan or Build-specific Variables
- Defining Plan Variables
- Triggering a Plan Build Manually

### Specifying the Source Repository for a Job

When creating a new Plan, you must define a ‘default source repository’. This includes specifying the source repository settings (including the repository’s location), any required authentication details and other options specific to the type of repository. You can also specify the type of build strategy the Plan will use.

The ‘default source repository’ is initially used by the Plan’s ‘Default Job’ and can also be used by other Jobs added to this Plan. You can override the default repository for any Job, by defining a custom source repository for it (after creating the Plan).

For details, please refer to the following pages (see Notes below):

- Bitbucket - Mercurial
- CVS
- Git
- GitHub
- Mercurial
- Perforce
Navigating to the Source Repository Settings for a Job

To navigate to the source repository settings for a Job:

1. Click ‘Home’ to go to the Dashboard.
2. Click the ‘All Plans’ tab.
3. Locate the Plan in the list which contains the Job you wish to configure and click the Plan’s name. The Plan’s Plan Summary page will be displayed.
4. Locate the Job in the ‘Plan Navigator’ on the left and click the name of the Job. The Job’s Summary will be displayed.
5. Click the ‘Actions’ menu and click ‘Configure Job’.
6. Click the ‘Source Repository’ tab to display the repository settings for your Job.

Screenshot above: Configuring a Job — Subversion Source Repository

Notes

- A number of source repositories are supported ‘out of the box’, as described on the Supported Platforms page. If you need to use a type of repository that is not supported, 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.

Configuring Agents and Capabilities

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

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

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.

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 for more information.

On this page:

- How Do Capabilities Work With Requirements?
- How Are Builds Distributed to Agents?
- How do Capabilities Affect the Distribution of Builds to Agents?*
- Related Topics

How Do Capabilities Work With Requirements?

A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

Together, capabilities and requirements control which agents can execute builds for particular Jobs. Each Job can only be built by agents whose capabilities meet the Job's requirements. See Configuring a Job's Requirements for more information.

How Are Builds Distributed to Agents?

An agent will consume a single Job at a time and block any other Bamboo Jobs from being processed until that Job build is complete. If you would like to build multiple Jobs simultaneously on the Bamboo server, then simply set up multiple local agents. If the agents are remote, then you will 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?
A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

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

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.

**On this page:**
- Viewing an Agent
- Viewing the Status of all Agents
- Creating a New Agent
- Editing an Agent's Details
- Configuring an Agent's Capabilities
- Disabling or Deleting an Agent
- Notes

**Viewing an Agent**

To view an agent, including the agent properties, capabilities and the plans that an agent can build, see Viewing an Agent.

**Viewing the Status of all Agents**

To view the status of all of your agents, see Monitoring Agent Status.

**Creating a New Agent**

To create a new agent, see:
- Creating a Local Agent, or
- Creating a Remote Agent.

**Editing an Agent's Details**

To edit an existing agent's details, see Editing an Agent's Details.

**Configuring an Agent's Capabilities**

To configure an existing agent's capabilities, see Configuring Capabilities.

**Disabling or Deleting an Agent**

To disable or delete an agent, see Disabling or Deleting an Agent.

**Notes**

- 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).

**Related Topics**

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

**Creating a Local Agent**

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:
- local agents run as part of the Bamboo server.
- remote agents are computers other than the Bamboo server that run the remote agent tool.

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2).

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.
Creating a Local Agent

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 to display the Agents screen, which lists all Local and Remote Agents that currently exist in your Bamboo system.
3. Click the 'Add Local Agent' button. The Add Local Agent screen will be displayed (see screenshot below).
4. In the 'Name' field, type the name of the new agent. This name will be displayed on the dashboard.
5. In the 'Description' field, type a description of the new agent. This description will only be shown to administrators.
6. Click the 'Add' button. This returns you to the Agents screen, specific to the agent you added, showing you details about your agent's capabilities, the Jobs it can build and its system properties. Note, new agents are enabled by default.

Your new agent will be able to run builds for all Jobs whose requirements are met by the agent's capabilities (see Configuring a Job's Requirements).

Add Local Agent

Enter a new unique name and a description for this local agent.

Information

Name

Description

Add    Cancel

Notes

- One local agent, with the default name of 'Default Agent', is automatically created after installing Bamboo.

Related Topics

Creating a Remote Agent

Creating a Remote Agent

This page describes how to install the Bamboo Remote Agent manually.

Before you begin:

- Not sure whether to install a remote agent? See Configuring Agents and Capabilities 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
- Step 3. Configure the Remote Agent's Capabilities
- Step 4. (Optional) Rename the Remote Agent

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:

![Installing a Remote Agent](#)

To install a remote agent, please follow these instructions:

1. Ensure that you have Java Runtime Environment 5.0 or later installed on the agent machine.
2. Download the remote agent JAR file to a directory on the agent machine.

![DOWNLOAD Remote Agent JAR](#)

Running a Remote Agent

Once installed, you can run the remote agent by executing the following command line from the directory containing the remote agent JAR file:

```
java -jar atlassian-bamboo-agent-installer-3.0-SNAPSHOT.jar
http://brydie:8086/bamboo/agentServer/
```

This will start a service wrapper for your agent, which will automatically restart in case of failure. You may also add extra system properties like `-Dbamboo.home=`... to customise the agent's home location. For more information, see our [Bamboo Remote Agent Installation Guide](#).

Screenshot above: Installing a remote agent

6. Click the 'DOWNLOAD Remote Agent JAR' button and save the JAR file to the directory you created in step 1.1.
7. Note the command under the heading 'Running a Remote Agent' for use in step 2 below.

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 -jar atlassian-bamboo-agent-installer-2.2-SNAPSHOT.jar
```

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.
If you are having issues bringing up the Agent, then take a look at our troubleshooting guide.

**Changing where the remote agent stores its data**

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

```
---------------------------------------------------------------------------------------------------------------------
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**

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 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

```
---------------------------------------------------------------------------------------------------------------------
```

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

```
---------------------------------------------------------------------------------------------------------------------
```

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.

```
<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.

```
<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 and Enabling Remote Agents Support

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

- **local agents** run as part of the Bamboo server.
- **remote agents** are computers other than the Bamboo server that run the remote agent tool.

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2).

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 and can only run builds for Jobs whose requirements are met by the agent's capabilities.

For more information, see:
Disabling Remote Agent 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.

**Before you begin:**

- 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 to display the Agents screen.
3. Click the 'Disable Remote Agent Support' link (see screenshot below).

![Screenshot above: Disabling remote agent support](image1)

Re-Enabling Remote Agent Support

**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. This displays the Agents screen with a message indicating that remote agent support is disabled.
3. Click the 'Enable Remote Agent Support' link (see screenshot below).

![Screenshot above: Re-enabling Remote Agent Support](image2)

Editing an Agent's Details

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

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

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 and can only run builds for Jobs whose requirements are met by the agent's capabilities.
For more information, see:

- Configuring Agents
- Configuring Agents and Capabilities
- Configuring a Job’s Requirements

On this page:

- Editing an Agent’s Name or Description
- Editing an Agent’s Capabilities
- Notes

**Editing an Agent’s Name or Description**

To edit an agent’s name or description:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. Click the ‘Edit Details’ link.
3. In the ‘Name’ field, type the agent’s new name.
4. In the ‘Description’ field, type the new description of the agent.
5. Click the ‘Save’ button.

**Editing an Agent’s Capabilities**

To edit an agent’s capabilities, see:

- Configuring an Agent-specific Executable Capability
- Configuring an Agent-specific JDK Capability
- Configuring an Agent-specific Custom Capability

**Screenshot above: Editing the details of an agent**

**Notes**

Related Topics

- Configuring Capabilities
- Configuring an Agent-specific Executable 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, to prevent that agent from running any further builds.

- **Disabling an agent** lets you keep the agent in Bamboo, but stops it from running builds.
  - If you need to prevent Bamboo from building any Plans at all (e.g. while you re-index Bamboo), you can disable all agents. By doing so, 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 for more information).

Note that you can also delete/disable individual Plans and/or their Jobs. This prevents the Plan(s) and/or their Jobs from being submitted to the build queue. See Disabling or Deleting a Plan and Disabling or Deleting a Job.
Disabling an Agent

To disable an agent:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column to display the Agents screen, which lists all agents that currently exist in your Bamboo system. The Status column indicates which agents are currently enabled/disabled.
3. Locate the relevant agents you wish to disable and select their check boxes in the left-most column of the table.
4. Click the 'Disable' button above the table.

Deleting an Agent

To delete an agent:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column to display the Agents screen, which lists all agents that currently exist in your Bamboo system.
3. Locate the relevant agents you wish to delete and select their check boxes in the left-most column of the table.
4. Click the 'Delete' button above the table.

Notes

Related Topics
Disabling or Deleting a Plan
Disabling or Deleting a Job
Creating a Local Agent
Creating a Remote Agent

Viewing an Agent

About Agents

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

- local agents run as part of the Bamboo server.
- remote agents are computers other than the Bamboo server that run the remote agent tool.

An elastic agent is a remote agent that runs in the Amazon Elastic Compute Cloud (EC2).

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.)

Viewing an Agent

To view an agent:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column to display the Agents page, which lists all Local and Remote Agents that
1. bamboo 3.1 documentation

2. Related Topics

For more information on configuring Agents, see the following topics:

- Viewing an Agent's Capabilities
- Viewing the Jobs that an Agent can build
- Determining which Agents can build which Jobs
- Viewing an Agent's System Properties

3. 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).

4. Notes

To define a new capability, see Configuring Capabilities.
Notes

- How capabilities work with requirements — A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

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

Related Topics

Viewing the Jobs that an Agent can build
Determining which Agents can build which Jobs

REV30 - 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).

**On this page:**
- Viewing an Agent's Capabilities
- Notes

**Viewing an Agent's Capabilities**

**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 to display the Agents page, which lists all Local and Remote Agents that currently exist in your Bamboo system.
3. Click the name of the agent whose capabilities you wish to view. The Agent page is displayed.
4. If necessary, click the 'Capabilities' tab to show a list of all 'Agent-Specific Capabilities' and 'Shared Capabilities' (see screenshot below). The capabilities in each of these sections are grouped into the following subsections:
   - 'Custom' — custom capabilities.
   - 'Builder' — builder capabilities.
   - 'JDK' — JDK capabilities.
   - 'Perforce' — perforce capability.

   These subsections will only be shown if you have at least one of that particular type of capability defined in Bamboo.

   To define a new capability, see Configuring Capabilities.
**Agent-Specific Capabilities**

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom.

No agent-specific capabilities currently exist.

**Shared Capabilities**

The following 'shared' capabilities are inherited by this agent. Note that the value of a shared capability is overridden by the value of an agent-specific capability with the same key/label (if one exists).

**Builder**

<table>
<thead>
<tr>
<th>Builder Label</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant (Ant)</td>
<td>/Users/bydlo/develtools/apache-ant-1.8.1</td>
<td>View</td>
</tr>
<tr>
<td>Bash (Command)</td>
<td>/bin/bash</td>
<td>View</td>
</tr>
<tr>
<td>Maven 1 (Maven 1.x)</td>
<td>/Users/bydlo/develtools/maven-1.0.2</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.x)</td>
<td>/Users/bydlo/develtools/maven-2.1.0</td>
<td>View</td>
</tr>
<tr>
<td>NANT (NAnt)</td>
<td>woo</td>
<td>View</td>
</tr>
</tbody>
</table>

**JDK**

JDK capabilities define the JDKs which are available to your build plans.

<table>
<thead>
<tr>
<th>JDK Label</th>
<th>Java Home</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDK</td>
<td>/System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.5</td>
<td>/System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home</td>
<td>View</td>
</tr>
<tr>
<td>JDK 1.6.0-20</td>
<td>/System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home</td>
<td>View</td>
</tr>
</tbody>
</table>

**Perforce**

The Perforce capability defines the location of the Perforce executable available to your build plans.

<table>
<thead>
<tr>
<th>Perforce</th>
<th>Perforce executable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The location of the Perforce P4 client application</td>
</tr>
</tbody>
</table>

**Mercurial**

The path to the Mercurial executable (e.g. 'C:\Program Files (x86)\Mercurial\hg.exe' or '/usr/local/bin/hg')

<table>
<thead>
<tr>
<th>Executable</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/usr/local/bin/hg</td>
</tr>
</tbody>
</table>

**Notes**

- How capabilities work with requirements — A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

  Together, capabilities and requirements control which agents can execute builds for particular Jobs. Each Job can only be built by agents whose capabilities meet the Job's requirements.
Viewing the Jobs that an Agent can build
Determining which Agents can build which Jobs

Viewing the Jobs that an Agent can build

A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job's Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

On this page:
- Viewing the Plans that an Agent Can Build
- Notes

Viewing the Plans that an Agent Can Build

To view the plans that an agent can build:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agents' link in the left navigation column to display the Agents screen, which lists all Local and Remote Agents that currently exist in your Bamboo system.
3. Click the name of the agent of interest. The Agent page is displayed.
4. If necessary, click the 'Executable Plans' tab to show a list of Jobs which this agent is capable of building (see screenshot below).
Bamboo 3.1 Documentation

**Notes**

- Determining which Plans an agent can build — A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

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

**Related Topics**

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

### Determining which Agents can build which Jobs

Determining whether an agent can build a Plan's Job depends on whether the capabilities of an agent match the requirements of that Job. Read more on Configuring a Job's Requirements and Configuring Capabilities.

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

**Viewing Which Agents Can Build Which Jobs**

**To view which agents can build which Jobs:**

In the pipeline navigation, select the plan that you wish to view the agents for, then click the 'Viewing Which Agents Can Build Which Jobs' link in the navigation menu.
1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Agent Matrix' link in the left navigation column to display the 'Agents and Plans Matrix' page (see screenshot below).
   - This matrix shows all Jobs currently set up in your Bamboo system, including disabled Jobs.
   - Every shared agent, remote agent and elastic image is listed against each Job with either a tick ✓ (agent is capable of building this Job) or a cross ✗ (agent is not capable of building this Job).
   - To view the Jobs that a specific agent can build, refer to Viewing the Jobs that an Agent can build.
3. If an agent is not capable of building a particular Job, hover your mouse over the cross ✗ to see which Job requirements are not being met.

**Agents and Plans Matrix**

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

<table>
<thead>
<tr>
<th>Job Name</th>
<th>1 Local Agent</th>
<th>2 <a href="mailto:bamboo-PC@server.atlassian.com">bamboo-PC@server.atlassian.com</a></th>
<th>Default</th>
<th>Maven 2.1 image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Job</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Default Job</td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Func Test Clean</td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Func Test Dependencies</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Func Test Misc</td>
<td></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Screenshot above: The 'Agents and Plans Matrix'*

**Notes**

**Related Topics**

- Configuring a Job's Requirements
- Configuring Capabilities.

**Viewing an Agent's System Properties**

To view the system properties for an agent:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. Click the 'System Properties' tab to view the agent's system properties.
Monitoring Agent Status

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

Online versus Offline agents:

- An “Online” agent is an agent which is currently accessible to Bamboo. Local agents are always online, although remote agents may be either online or offline.
- An “Offline” agent is a remote agent which has been registered with the Bamboo server, was online, but now can not interact with the
Bamboo server because:

- The Bamboo remote agent process (running on the remote hardware) was stopped.
- The Bamboo server (for whatever reason) cannot communicate with the remote hardware that is running the Bamboo remote agent process.

Bamboo administrators can manually 'disable' an online agent to prevent it from being used in build generation. The agent will still be online and it can be 'enabled' at a later point in time. It is not possible to disable offline agents.

On this page:

- Monitoring the Status of Your Agents
- Notes

Monitoring the Status of Your Agents

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. 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:
   - 'Idle' — available to execute Job builds
   - 'Building' — currently executing a Job build
   - 'Cancelling' — currently cancelling a Job build
   - 'Disabled' — not available to execute Job builds (see Disabling or Deleting an Agent)
   - 'Disabled - Building' — currently executing a Job build, but disabled so cannot execute further Job builds
   - 'Disabled - Cancelling' — currently cancelling a Job build, and disabled so cannot execute further Job builds

Remote agents are divided into 'Online Remote Agents' and 'Offline Remote Agents'. You can access more details on either of these categories by clicking the appropriate tab.

- Online remote agents are remote agents which are currently available for use by Bamboo.
- Offline remote agents are remote agents which are currently inaccessible to Bamboo and are unavailable for generating Job builds.
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.

Select All, None, Idle, Disabled Action: Delete Disable Enable

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Agent</td>
<td>Building - SANDBOX-RFIN-JOB1-22</td>
<td>View</td>
</tr>
<tr>
<td>Reporting agent</td>
<td>Idle (Disabled)</td>
<td>View</td>
</tr>
</tbody>
</table>

Add Local Agent

Remote Agents

Remote agents run on computers other than the Bamboo server.

Select All, None, Idle, Disabled Action: Delete Disable Enable

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Agent  on 01bcc86d</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent  on 03bcc86f</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent  on 1fbcc873</td>
<td>Building - BTS-FUNC-JOB1-7815</td>
<td>View</td>
</tr>
</tbody>
</table>

There are currently 3 remote agents online (3 elastic). Start elastic agents here. A maximum of 25 agents are supported by your license.

Notes

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

On this page:
- Viewing an Agent's Capabilities
- Defining a New Capability
- Editing a Capability
- Renaming a Capability
- Deleting a Capability
- Viewing the Agents and Plans Related to a Capability
- Notes

Viewing an Agent's Capabilities

To find out what capabilities an agent already has, please see Viewing an Agent's Capabilities.

Defining a New Capability

To define a new capability, see:

- Configuring a new Executable, or
- Configuring a new JDK, or
- Configuring a new Custom Capability, or
- Configuring a new Perforce Capability.

Editing a Capability

To edit an existing capability, see Editing a Capability.

Renaming a Capability

To rename an existing capability, see Renaming a Capability.

Deleting a Capability

To delete a capability, see Deleting a Capability.

Viewing the Agents and Plans Related to a Capability

To view the agents and plans related to a capability, see Viewing a Capability's Agents and Jobs.

Notes

- A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

Together, capabilities and requirements control which agents can execute builds for particular Jobs. Each Job can only be built by agents whose capabilities meet the Job's requirements. See Configuring a Job's Requirements for more information.

Related Topics

Configuring Agents
Configuring a new Executable

About Executables

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

At least one executable was automatically configured when you installed Bamboo. You can configure more by defining capabilities for the new executables (i.e. executable capabilities). Bamboo supports the following types of executables:

- Ant
- Maven
- Grails
- NAnt
- devenv.com
- msbuild.exe
- PHPUnit
- Custom command (e.g. ‘make’)
- Script

The executables listed above are supported out-of-the-box. If you need to use a different executable, a number of third-party plugin modules are available (e.g. NoseXUnit). You can also create your own executable plugin (see the Bamboo Plugin Guide for details).

Configuring a new Executable

You can define a new executable capability for:

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

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

Once you have configured a new executable capability in your Bamboo system, its label (e.g. ‘Ant’) will appear in the Executable dropdown list when you configure the executable for a Task (see Configuring Tasks). The executable you select will be used every time the Task is executed during a Job build. That is, the Task can only be run by agents which have a capability that matches the executable specified in the Task’s Executable field.

Notes

- Pre-configured executables — The executable that was automatically configured when you installed Bamboo depends on the system environment variables (e.g. ‘ANT_HOME=/opt/java/ant’) that were present on the machine that Bamboo was installed on.
  - 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.
- msbuild.exe — 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 executable.
- PHPUnit — 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 executable.

Viewing your Executable Capabilities

You can view all the executable capabilities that have been defined in Bamboo on the ‘Executables’ page. These include local server capabilities, local agent-specific capabilities and remote agent-specific capabilities.

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

On this page:

- Viewing and Configuring Executable Capabilities
- Notes
Viewing and Configuring Executable Capabilities

To view and configure the Executable capabilities defined in Bamboo:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Executables' link in the left navigation column. The Executables page will be displayed, showing the list of all the Executable capabilities defined in your Bamboo system.
   
   - To view a specific executable capability, click the specific executable's tab on the left. This will show you the agents and Jobs related to this executable capability, with which you can perform the following actions:
     
     - View more details about an agent with this executable capability — To do this, click the linked name of the agent in the Agent column. This will show you the complete list of capabilities and Jobs associated with that agent.
     
     - Edit the executable path of an agent with this executable capability — To do this, click the 'Edit' link in the Operations column for the agent you wish to configure.
     
     - Remove this executable capability from an agent — To do this, click the 'Delete' link in the Operations column for the agent that currently possesses this executable capability.
     
     - Be aware that you can only remove a executable capability from all local agents, not from individual local agents. See the note below for more information.
     
     - View details about (and configure) an elastic image with this executable capability — To do this, click the linked name of the elastic image in the Elastic Image Configuration column.
     
     - Edit the configuration of a Job which relies on/requires this executable capability — To do this, click the linked name of the Job in the Plan column.
     
     - If you are currently viewing a Maven (2.x or later) executable capability, you can configure repository isolation for it by clicking the 'Edit Capability Configuration' link. Please refer to Configuring Repository Isolation for Maven Executables for more information.

   - To add a new executable as a local server capability, click the 'Add executable to server capabilities' link to navigate to the Server Capabilities page.

![Executables Table](image)

Screenshot above: 'Executables'

Notes

- **Bamboo's automatic detection of executables** — When you install the Bamboo server application or the Bamboo Remote Agent
application on another machine, either of these applications will automatically look for existing executables installed on the same machine (based on a combination of the machine's environment variables and other conditions). A 'executable capability' will be created for each executable that that either of these Bamboo applications find.

The environment variables and conditions that Bamboo uses to automatically detect and create executable capabilities are listed below. With the exception of the 'Command' executable, the paths for each automatically detected executable are based on the path 'string' values found within these environment variables.

- **Ant** — the `ANT_HOME` environment variable
- **Maven** — the `MAVEN_HOME` or `M2_HOME` environment variable (Maven 1), or `MAVEN2_HOME` environment variable (Maven 2.x)
- **Grails** — the existence of the `/bin/grails` file
- **PHPPUnit** — the existence of the `phpunit` file anywhere within the machine's `PATH` environment variable value

- **Local agents and executable capabilities** — Since Bamboo automatically looks for executables installed on the same machine and creates a 'executable capability' for each executable installation it finds, all existing and subsequent local agents that you create will possess these executable capabilities. Hence, when you access the 'Executables' page and view these executable capabilities, all local agents will be grouped together in the 'All local agents' category and you will only be able to remove these executable capabilities from all local agents, not from individual local agents.

Related Topics

**Configuring Capabilities**

**Configuring an Agent-specific Executable Capability**

Once you have configured a new executable capability in your Bamboo system, its label (e.g. ‘Ant’) will appear in the Executable dropdown list when you configure the executable for a Task (see Configuring Tasks). The executable you select will be used every time the Task is executed during a Job build. That is, the Task can only be run by agents which have a capability that matches the executable specified in the Task's Executable field.

An agent-specific capability is a capability that 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).

---

Configuring a New Agent-Specific Builder Capability

**To configure a new agent-specific Builder capability:**

1. Navigate to the desired agent, as described on Viewing an Agent.
2. In the ‘Agent-Specific Capabilities’ section of the ‘Capabilities’ tab, click the ‘Add Capability’ link. The ‘Add Capability’ page is displayed (see screenshot below).
3. Set the ‘Capability Type’ field to ‘Executable’.
4. Select the appropriate ‘Type’ of Executable from the dropdown list.
5. In the ‘Executable Label’ field, type a name/label for the Executable, which Bamboo presents in the ‘Executables’ dropdown list whenever a Task's executable is configured.
6. 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’.
7. Click the ‘Add’ button. This will verify whether the Executable 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 — Executable](image-url)
Configuring a Shared Executable Capability

Once you have configured a new executable capability in your Bamboo system, its label (e.g. ‘Ant’) will appear in the Executable dropdown list when you configure the executable for a Task (see Configuring Tasks). The executable you select will be used every time the Task is executed during a Job build. That is, the Task can only be run by agents which have a capability that matches the executable specified in the Task’s Executable 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).

On this page:
- Configuring a New Local Server Executable Capability
- Configuring a New Shared Remote Executable Capability
- Notes

Configuring a New Local Server Executable Capability

Before you begin:
- Shared remote executable capabilities are not shared with elastic agents.
- If you want to run multiple Maven agents on your local server, you will need to configure repository isolation for your Maven executables. See Configuring Repository Isolation for Maven Executables for details.

To configure a new local server executable capability:

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Server Capabilities’ link in the left navigation column to display the Server Capabilities page, which lists all local server capabilities currently defined in your Bamboo system.
3. In the Add Capability section at the end of this page (see screenshot below), select ‘Executable’ from the ‘Capability Type’ dropdown list.
4. Select the appropriate type of Executable from the ‘Type’ dropdown list.
5. In the ‘Executable Label’ field, type a name/label for the Executable, which Bamboo presents in the ‘Executable’ dropdown list whenever a Task’s executable is configured.
6. 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’.

⚠️ For Ant and Maven, Bamboo requires the Path to be the location of the Executable installation folder.
7. Click the ‘Add’ button. This will verify whether the Executable and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.

Configuring a New Shared Remote Executable Capability

To configure a new shared remote Executable capability:

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Agents’ link in the left navigation column to display the Agents page, which lists all Local and Remote Agents that currently exist in your Bamboo system.
3. In the Remote Agents section, click the ‘Shared Remote Capabilities’ link at the right. The Shared Remote Capabilities page is displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
4. In the Add Capability section at the end of this page (see screenshot below), select ‘Builder’ from the ‘Capability Type’ dropdown list.
5. Select the appropriate type of Executable from the ‘Type’ dropdown list.
6. In the ‘Executable Label’ field, type a name/label for the Builder, which Bamboo presents in the ‘Executables’ dropdown list whenever a Task’s executable is configured.
7. 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’.

⚠️ For Ant and Maven, Bamboo requires the Path to be the location of the Executable installation folder.
8. Click the ‘Add’ button. This will verify whether the Executable and Path you have specified are valid. If they are not, re-enter the values and then click the ‘Add’ button again.
Bamboo allows you to isolate Maven (2.x or later only) executables on an agent-specific basis. If you configure repository isolation for a particular Maven executable capability, each agent that uses this executable will have its own private Maven 2.x artifacts directory, thereby allowing you to avoid these jar and dependency file corruptions. Each isolated repository directory has the path:

```
$BAMBOO_HOME/.m2/AGENT-${agendid}/repository
```

You may want to configure repository isolation for Maven executables, if you run multiple Maven executables on one server machine which run under the same user account on that server, but belong to different Bamboo agents. In this case, the agents will use the same default Maven artifacts directory: $HOME/.m2/repository (or %USERPROFILE%\.m2\repository for Windows-based servers). This is the directory to which Maven dependency jars are downloaded and where project artifacts are installed during the "install" phase of a Maven build.

Hence, problems can arise if Bamboo uses these multiple Maven executables simultaneously. For example, if multiple agents on a single computer, each with a different Maven executable capability, start to run Maven builds simultaneously from the queue, the different Maven executables may attempt to download the same dependency to the same artifacts directory location, resulting in corruption of the downloaded jar and dependency files.

### Configuring Repository Isolation for Maven Executables

**Before you begin:**

- This feature is not available for Maven 1.x executables.
- When configuring any Maven executables in Bamboo in which you want to force local repository isolation, ensure that the executable label you use is one that identifies it as such — for example, 'Maven 2.x with local repository isolation'.

**To configure a new local server Maven capability with repository isolation:**

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Server Capabilities' link in the left navigation column to display the 'Server Capabilities' page, which lists all local server capabilities currently defined in your Bamboo system.
3. In the Add Capability section at the end of this page, choose your executable and enter its details as described:
   - 'Capability Type' — Select 'Executable'
   - 'Type' — Select one of the Maven options (2.x or later)
   - 'Executable Label' — Enter 'Maven with local repository isolation'
   - 'Path' — Enter the path for your Maven executable

4. Click the 'Add' button.
5. Click the label for the executable you have just added. The executable capability summary screen will be displayed (see 'Maven 2.x Executable' screenshot below).
6. Click the 'Edit Capability Configuration' link. The 'Configure Capability' screen will be displayed (see 'Maven 2.x Repository Isolation' screenshot below).
7. Select the 'Local repository isolation' check box.
8. Click the 'Save' button.

### Maven 2 with local repository isolation

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

#### Agents with capability

The following agents have this capability.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>All local agents</td>
<td>/Volumes/Pharlap/opt/dev/tools/maven</td>
<td>Edit</td>
</tr>
<tr>
<td>bamboo-per-sydney.atlassian.com</td>
<td>C:\Program Files\Java\apache-maven-2.1.0</td>
<td>Edit</td>
</tr>
</tbody>
</table>

#### Elastic Image Configurations with capability

3 elastic image configurations have this capability.

<table>
<thead>
<tr>
<th>Elastic Image Configuration</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBTEST</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Default</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Maven 2.1 Image</td>
<td>/opt/maven-2.0</td>
</tr>
</tbody>
</table>

#### Jobs with requirement

28 jobs rely on this capability.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact Sharing Dogfooding › Artifact sharing › Consumer</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Artifact Sharing Dogfooding › Artifact sharing › Final</td>
<td>exists</td>
<td></td>
</tr>
</tbody>
</table>

#### Capability Configuration

Local repository isolation: Yes

Rename Capability | Edit Capability Configuration

---

**Screenshot above: Maven Executable**

---

### Executable > Maven 2

#### Configure Capability

Update specific features of this capability.

- [ ] Local repository isolation

  When enabled, executable will use per-agent private maven local repository.

[Save] [Cancel]
Notes

Related Topics

Configuring a Shared Executable Capability

Configuring a new JDK

At least one JDK was automatically configured when you installed Bamboo. 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 dropdown list when you configure a Job's builder (see Configuring Tasks). The JDK you select will be used for every one of that Job's builds. That is, the Job can only be built by agents which have a JDK capability whose label is specified in the Job's Build JDK field.

1 This 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 dropdown list when you configure a Job's builder (see Configuring Tasks). The JDK you select will be used for every one of that Job's builds. That is, the Job can only be built by agents which have a JDK capability whose label is specified in the Job's Build JDK field.

An agent-specific capability is a capability that 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).

On this page:

- Configuring an Agent-Specific JDK Capability
- Notes

Configuring an Agent-Specific JDK Capability

To configure a new agent-specific JDK capability:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. In the 'Agent-Specific Capabilities' section on the 'Capabilities' tab, click the 'Add Capability' link. The 'Add Capability' page is displayed (see screenshot below).
3. Set the 'Capability Type' field to 'JDK'.
4. In the 'JDK Label' field, type a name/label for the JDK, which Bamboo presents in the 'Build JDK' dropdown list whenever a Job's builder is configured.
5. In the 'Java Home' field, type the location of the JDK Home Directory.
6. 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.

Screenshot: 'Add Capability — JDK'
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 'JDK Label' = 'JDK' and '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 'JDK Label' = 'JDK 1.5' and '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 'JDK Label' = 'JDK 1.5.0_07' and '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.

Related Topics

**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** dropdown list when you configure a Job's builder (see Configuring Tasks). The JDK you select will be used for every one of that Job's builds. That is, the Job can only be built by agents which have a JDK capability whose label is specified in the Job'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).

### On this page:

- Configuring a New Local Server JDK Capability
- Configuring a New Shared Local Remote JDK Capability
- Notes

**Configuring a New Local Server JDK Capability**

Before you begin:

- 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 to display the **Server Capabilities** page, which lists all local server capabilities currently defined in your Bamboo system.
3. In the Add Capability section at the end of the page (see screenshot below), select 'JDK' from the 'Capability Type' dropdown list.
4. In the 'JDK Label' field, type a name/label for the JDK, which Bamboo presents in the 'Build JDK' dropdown list whenever a Job's builder is configured.
5. In the 'Java Home' field, type the location of the JDK Home Directory.
6. 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 a New Shared Local Remote JDK Capability**

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 to display the Agents page, which lists all Local and Remote Agents that currently exist in your Bamboo system.
3. In the Remote Agents section, click the 'Shared Remote Capabilities' link at the right. The Shared Remote Capabilities page is displayed, showing a list of all shared remote capabilities currently defined in your Bamboo system.
4. In the Add Capability section at the end of this page (see screenshot below), select 'JDK' from the 'Capability Type' dropdown list.
5. In the 'JDK Label' field, type a name/label for the JDK, which Bamboo presents in the 'Build JDK' dropdown list whenever a Job's builder is configured.
6. In the 'Java Home' field, type the location of the JDK Home Directory.
7. 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.

**Screenshot: ‘Add Capability — JDK’**

![Add Capability](image)

**Notes**

- **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 'JDK Label' = 'JDK' and '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 'JDK Label' = 'JDK 1.5' and '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 'JDK Label' = 'JDK 1.5.0_07' and '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.

**Related Topics**

Configuring an Agent-specific JDK Capability

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.

**On this page:**

- Viewing and Configuring JDKs
- Notes

Viewing and Configuring JDKs

To view and configure 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. The JDKs page will be displayed, showing the list of all the JDK capabilities defined in your Bamboo system.
   - To view a specific JDK capability, click the specific JDK's tab on the left. This will show you the agents and Jobs related to this JDK capability, with which you can perform the following actions:
     - View more details about an agent with this JDK capability — To do this, click the linked name of the agent in the Agent column. This will show you the complete list of capabilities and Jobs associated with that agent.
     - Edit the JAVA_HOME path of an agent with this JDK capability — To do this, click the 'Edit' link in the Operations column for the agent you wish to configure.
     - Remove this JDK capability from an agent — To do this, click the 'Delete' link in the Operations column for the agent that currently possesses this JDK capability. **Be aware that you can only remove a JDK capability from all local agents, not from individual local agents. See the note below for more information.**
     - View details about (and configure) an elastic image with this JDK capability — To do this, click the linked name of
2.
the elastic image in the **Elastic Image Configuration** column.

- **Edit the configuration of a Job which relies on this JDK capability** — To do this, click the linked name of the Job in the **Plan** column.
- **To add a new JDK as a local server capability**, click the ‘**add a JDK as a server capability**’ at the top of the page. This opens the **Server Capabilities** page at the **Add Capability** section, with the JDK selected as the **Capability Type**.

### JDKs

The following table lists the JDKs that have been defined as capabilities in Bamboo. You can add a JDK as a server capability on this page (shared by all local agents), or delete any of the existing JDK capabilities listed. A JDK can only be added as an agent-specific capability on that agent’s page.

<table>
<thead>
<tr>
<th>JDK</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.6.0.12</td>
<td></td>
</tr>
<tr>
<td>1.6.0.18 (JRE)</td>
<td></td>
</tr>
<tr>
<td>1.6.0.21</td>
<td></td>
</tr>
<tr>
<td>1.6.0.20</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>1.6.0.16</td>
<td></td>
</tr>
<tr>
<td>1.6.0.17</td>
<td></td>
</tr>
<tr>
<td>1.6.0.11</td>
<td></td>
</tr>
<tr>
<td>1.6.0.10</td>
<td></td>
</tr>
<tr>
<td>1.6.0.9</td>
<td></td>
</tr>
<tr>
<td>1.6.0.8</td>
<td></td>
</tr>
<tr>
<td>1.6.0.7</td>
<td></td>
</tr>
<tr>
<td>1.6.0.6</td>
<td></td>
</tr>
<tr>
<td>1.6.0.5</td>
<td></td>
</tr>
<tr>
<td>1.6.0.4</td>
<td></td>
</tr>
<tr>
<td>1.6.0.3</td>
<td></td>
</tr>
<tr>
<td>1.6.0.2</td>
<td></td>
</tr>
<tr>
<td>1.6.0.1</td>
<td></td>
</tr>
<tr>
<td>1.6.0.0</td>
<td></td>
</tr>
</tbody>
</table>

### Elastic Image Configuration with capability

<table>
<thead>
<tr>
<th>Elastic Image Configuration</th>
<th>Java Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>kETest</td>
<td>/opt/jdk-6</td>
</tr>
<tr>
<td>Default</td>
<td>/opt/jdk-6</td>
</tr>
<tr>
<td>Maven 2 1 Image</td>
<td>/opt/jdk-6</td>
</tr>
</tbody>
</table>

### Jobs with requirement

16 jobs rely on this capability.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Java Home</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bamboo Testing, Acceptance Test J1.6, Default Job</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Bamboo Testing, Remote Agent Functional Tests on JDK 1.6, Default Job</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Common, Atlassian HC, Default Job</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>JIRA Performance, JIRA Stable - UI Performance Test FF3.0, Default Job</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>JIRA Performance, JIRA Stable - UI Performance Test IE7, Default Job</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>JIRA Performance, JIRA Stable - UI Performance Test IE8, Default Job</td>
<td>exists</td>
<td></td>
</tr>
</tbody>
</table>

### Screenshot above: Viewing the JDKs in Bamboo

### Notes

- **Bamboo’s automatic detection of JDKs** — When you install either Bamboo or the Bamboo Remote Agent, it will automatically look for an existing JDK installed on the same machine (based on the machine’s JAVA_HOME environment variable) and create a ‘JDK capability’ for that JDK installation, with its path being the value of JAVA_HOME.
- **Local agents and JDK capabilities** — Since Bamboo automatically looks for an existing JDK installed on the same machine and...
creates a ‘JDK capability’ for it, all existing and subsequent local agents that you create will possess this JDK capability. Hence, when you access the ‘JDKs’ page and view this JDK capability, all local agents will be grouped together in the ‘All local agents’ category and you will only be able to remove this JDK capability from all local agents, not from individual local agents.

Related Topics

Configuring a new JDK

Configuring a new Custom Capability

Custom capabilities can be used to control which Jobs will be built by a particular agent. For example, if the builds for a particular Job 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 Job. (See Configuring a Job's 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 Jobs will be built by a particular agent. For example, if the builds for a particular Job 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 Job. (See Configuring a Job's Requirements.)

An agent-specific capability is a capability that 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).

On this page:
- Configuring a New Agent-Specific Custom Capability
- Notes

Configuring a New Agent-Specific Custom Capability

To configure a new agent-specific custom capability:

1. Navigate to the desired agent, as described in Viewing an Agent.
2. Click the ‘Add Capability’ link in the top right of the ‘Agent-Specific Capabilities’ section. The ‘Add Capability’ page will be displayed (see screenshot below).
3. Set the ‘Capability Type’ field to ‘Custom’.
4. In the ‘Key’ field, type the key for your new custom capability.
5. In the ‘Value’ field, type the value of your new custom capability.
6. Click the ‘Add’ button.

Add Capability - Local Agent

You can add an agent specific capability on this page. The value of this capability will override the value of a shared capability of the same name (if one exists).

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Custom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>local</td>
</tr>
<tr>
<td>Value</td>
<td>true</td>
</tr>
</tbody>
</table>

Add Cancel

Screenshot above: Adding an agent-specific custom capability

Notes

Related Topics
Configuring a Shared Custom Capability

Custom capabilities can be used to control which Jobs will be built by a particular agent. For example, if the builds for a particular Job 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 Job.

(See Configuring a Job's 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).

On this page:
- Configuring a New Local Server Custom Capability
- Configuring a New Shared Remote Custom Capability
- Notes

Configuring a New Local Server Custom Capability

Before you begin:
- 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. The 'Server Capabilities' screen will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
3. Locate the 'Add Capability' section at the bottom of the screen (see screenshot below).
4. In the 'Capability Type' field, select 'Custom'.
5. In the 'Key' field, type the key for your new custom capability.
6. In the 'Value' field, type the value of your new custom capability.
7. Click the 'Add' button.

Configuring a New Shared Remote Custom Capability

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. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
3. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section. The 'Shared Remote Capabilities' page will be displayed, showing a list of all shared remote 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.

Add Capability

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Custom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>local</td>
</tr>
<tr>
<td>Value</td>
<td>true</td>
</tr>
</tbody>
</table>

Add Cancel

Screenshot above: Adding a shared custom capability

Notes

Related Topics

Configuring an Agent-specific Custom Capability
Configuring a new Perforce Capability

If you wish to build Plans on your Bamboo server and remote agents using a Perforce repository, you need to specify the location of the Perforce P4 client application for your Bamboo server and each remote agent that uses Perforce. These locations are set by specifying:

- a mandatory local server Perforce capability for your Bamboo 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 is a capability that 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).

On this page:
- Configuring a New Local Server Perforce Capability
- Configuring a New Agent-Specific Remote Perforce Capability
- Notes

Configuring a New Local Server Perforce Capability

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.

Configuring a New Agent-Specific Remote Perforce Capability

To configure a new agent-specific remote Perforce capability:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. 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.
3. The 'Add Capability' page will be displayed (see screenshot below).
4. Set the 'Capability Type' field to 'Perforce'.
5. In the 'Perforce Executable' field, type the location of the P4 client application for your remote agents.
6. Click the 'Add' button.

Add Capability

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

The location of the Perforce P4 client application

Add Cancel

Screenshot above: Adding a new Perforce capability

Notes

Related Topics

Configuring Capabilities

Configuring a new Mercurial Capability

If you wish to build plans on your server and remote agents using a Mercurial repository, you need to specify the location of the Mercurial hg client application for your Bamboo server and for each remote agent using Mercurial. These locations are set by specifying:

- a mandatory local server Mercurial capability for your Bamboo server and
- agent-specific remote Mercurial capabilities for each of your remote agents using Mercurial.
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 is a capability that 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).

On this page:
- Configuring a New Local Server Mercurial Capability
- Configuring a New Agent-Specific Remote Mercurial Capability
- Notes

Configuring a New Local Server Mercurial Capability

To configure a new local server Mercurial 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 'Mercurial'.
6. In the 'Executable' field, select 'Mercurial'.
7. In the 'Path' field, provide a full path (or make sure it is on the system PATH) to the hg client application for your server.
8. Click the 'Add' button.

Configuring a New Agent-Specific Remote Mercurial Capability

To configure a new agent-specific remote Mercurial capability:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. 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.
3. The 'Add Capability' page will be displayed (see screenshot below).
4. In the 'Capability Type' field, select 'Mercurial'.
5. In the 'Executable' field, select 'Mercurial'.
6. In the 'Path' field, provide a full path (or make sure it's on the system PATH) to the hg client application for your remote agent.
7. Click the 'Add' button.

Screenshot: Adding a new Mercurial capability

Notes

Related Topics
- Configuring Capabilities
- Editing a Capability

Editing 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 for more information.

Because each agent can only run builds for Jobs whose requirements are met by the agent's capabilities (see Configuring a Job's Requirements), modifying a capability may mean that some plans can no longer be built.

The instructions on this page do not describe how to rename a capability. If you want to rename a capability, see Renaming a Capability.

On this page:
- Editing an Agent-Specific Capability
- Editing a Local Server Capability
- Editing a Shared Remote Capability
- Notes

Editing an Agent-Specific Capability

To edit an agent-specific capability:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. The 'Agent Capabilities' screen will be display, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
3. Click the 'Edit' corresponding to the agent-specific capability you wish to edit.
4. 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.
   - 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.

Editing a Local Server Capability

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.
   - 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.

Editing a Shared Remote Capability

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. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section.
4. The 'Shared Remote Capabilities' screen will display, showing a list of all shared remote capabilities currently defined in your Bamboo system.
5. Click the 'Edit' link corresponding to the capability you wish to edit.
6. 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.
   - 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.

Notes

Related Topics
Configuring Capabilities

Renaming a Capability

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

- builder (e.g. Maven)
Bamboo 3.1 Documentation

- 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 for more information.

On this page:
- Renaming an Agent-Specific Capability
- Renaming a Local Server Capability
- Renaming a Shared Remote Capability

Renaming an Agent-Specific Capability

To rename an agent-specific capability:

1. Navigate to the desired agent, as described in Viewing an Agent.
2. The ‘Agent Capabilities’ screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
3. Click the ‘View’ corresponding to the agent-specific capability you wish to rename.
4. The ‘View Capability’ page will display. Click the ‘Rename Capability’ link.
5. The ‘Rename Capability’ page will display. Update the ‘Key’ and click ‘Rename Capability’.

Renaming a Local Server Capability

To rename 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’ corresponding to the agent-specific capability you wish to rename.
5. The ‘View Capability’ page will display. Click the ‘Rename Capability’ link.
6. The ‘Rename Capability’ page will display. Update the ‘Key’ and click ‘Rename Capability’.

Renaming a Shared Remote 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 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’ 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 above: Renaming a custom capability

Deleting 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 for more information.

Before you begin:
- Because each agent can only run builds for Jobs whose requirements are met by the agent’s capabilities (see Configuring a Job’s Requirements), modifying a capability may mean that some plans can no longer be built.

On this page:
- Deleting an Agent-Specific Capability
- Deleting a Local Server Capability
- Deleting a Shared Remote Capability
- Notes

Deleting an Agent-Specific Capability

To delete an agent-specific capability:
1. Navigate to the desired agent, as described in Viewing an Agent.
2. The ‘Agent Capabilities’ screen will be displayed, showing a list of shared capabilities and agent-specific capabilities for your chosen agent.
3. Click the ‘Delete’ corresponding to the agent-specific capability you wish to delete.

Deleting a Local Server Capability

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.

Deleting a Shared Remote Capability

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.

Notes

Related Topics

Configuring Capabilities

Viewing a Capability’s Agents and Jobs

You can view a capability to determine the following information about it:
- which agents have/inherit the capability.
- which Jobs have the capability specified as a requirement.
Viewing an Agent-Specific Capability

To view an agent-specific capability:

1. Navigate to the desired agent, as described on Viewing an Agent.
2. If the 'Capabilities' tab is not selected, select it to show the list of this agent's agent-specific and shared capabilities.
3. Click the 'View' link corresponding to the capability you wish to view. The capability's summary page will be displayed (see the screenshot below), showing a list of agents and elastic images that possess this capability and the Bamboo plans that rely on this capability.
   - Click one of the listed agents to show further information about that agent:
     - Executable Jobs tab — all Jobs whose requirements match the capabilities of this agent
     - Capabilities tab — the capabilities of the agent itself
     - System Properties tab — system information about this agent
     - Recent Activity link — view recent builds for the agent
   - Click one of the listed elastic images to show the configuration page for that image. For more information about this configuration page, please refer to Viewing an Elastic Image.

Viewing a Local Server 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. The 'Server Capabilities' page will be displayed, showing a list of all local server capabilities currently defined in your Bamboo system.
3. Click the 'View' link corresponding to the capability you wish to view. The capability's summary page will be displayed (see screenshot below), showing a list of agents and elastic images that possess this capability and the Bamboo plans that rely on this capability.
   - Click one of the listed agents to show further information about that agent:
     - Executable Jobs tab — all Jobs whose requirements match the capabilities of this agent
     - Capabilities tab — the capabilities of the agent itself
     - System Properties tab — system information about this agent
     - Recent Activity link — view recent builds for the agent
   - Click one of the listed elastic images to show the configuration page for that image. For more information about this configuration page, please refer to Viewing an Elastic Image.

Viewing a Shared Remote Capability

Before you begin:

- Shared remote capabilities are not shared with elastic agents.

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. The 'Agents' page will be displayed, showing lists of all Local Agents and Remote Agents in your Bamboo system.
3. Click the 'Shared Remote Capabilities' link in the top right of the 'Remote Agents' section.
4. The 'Shared Remote Capabilities' page will be displayed, showing a list of all capabilities which are shared amongst all remote agents currently defined in your Bamboo system.
5. Click the 'View' link corresponding to the capability you wish to view. The capability's summary page will be displayed (see the screenshot below), showing a list of agents and elastic images that possess this capability and the Bamboo plans that rely on this capability.
   - Click one of the listed agents to show further information about that agent:
     - Executable Jobs tab — all Jobs whose requirements match the capabilities of this agent
     - Capabilities tab — the capabilities of the agent itself
     - System Properties tab — system information about this agent
     - Recent Activity link — view recent builds for the agent
   - Click one of the listed elastic images to show the configuration page for that image. For more information about this configuration page, please refer to Viewing an Elastic Image.
### Maven 2 with local repository isolation

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

#### Agents with capability

The following agents have this capability.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>All local agents</td>
<td>/Volumes/Pharlap/opt/dev/tools/maven</td>
<td>Edit</td>
</tr>
<tr>
<td>bamboopen-sydney.atlassian.com</td>
<td>C:\Program Files\Java\apache-maven-2.1.0</td>
<td>Edit</td>
</tr>
</tbody>
</table>

#### Elastic Image Configurations with capability

3 elastic image configurations have this capability:

<table>
<thead>
<tr>
<th>Elastic Image Configuration</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>KBTEST</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Default</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Maven 2.1 Image</td>
<td>/opt/maven-2.0</td>
</tr>
</tbody>
</table>

#### Jobs with requirement

28 jobs rely on this capability.

<table>
<thead>
<tr>
<th>Plan</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact Sharing Dogfooding &gt; Artifact sharing &gt; Consumer</td>
<td>exists</td>
<td></td>
</tr>
<tr>
<td>Artifact Sharing Dogfooding &gt; Artifact sharing &gt; Final</td>
<td>exists</td>
<td></td>
</tr>
</tbody>
</table>

### Capability Configuration

<table>
<thead>
<tr>
<th>Local repository isolation</th>
<th>Yes</th>
</tr>
</thead>
</table>

[Rename Capability | Edit Capability Configuration]

**Screenshot above: Viewing a Capabilities of an Agent**

### Notes

Related Topics

Configuring Capabilities

### Working with Builds

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

- Triggering a Plan Build
- Reordering Jobs in the Build Queue
- Monitoring Job Builds

### Triggering a Plan Build

There are a number of methods by which Bamboo can 'trigger' (i.e. execute) a Plan's build,

- **Build Strategy:**
  - Code updated — a Plan's build is triggered whenever one or more authors checks in code.
  - Scheduled build — a Plan's build is triggered at scheduled times or specified time intervals.
Manual build — a Plan's build is only ever triggered manually.
Initial clean build — a Plan's build is triggered immediately after a new Plan has been created.
Build Dependency:
  * Dependency — a build can be triggered whenever another Plan's build is successfully completed.

The trigger method for a Plan's build is listed in the 'Reason' column on the Dashboard.

On this page:
  * Build Dependencies and Build Strategies
  * Considerations for choosing a Build Strategy
  * Related Topics

Build Dependencies and Build Strategies

Build dependencies are "parent-child" relationships between Plan builds, which can be configured for triggering a series of Plan's builds. See Setting up Build Dependencies.

Build dependencies work together with the build strategies of Plans to trigger builds of these Plans. For example, you can set up Plan A to poll its repository for changes as well as configure a build dependency on a parent Plan (Plan B). In such a case, builds of Plan A will be triggered when code changes are detected in its repository as well as when builds of Plan B 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 Plan Build Manually.

Considerations for choosing a Build Strategy

Code Updated

"Triggering a Plan's build when code is updated" ensures that a Plan build only occurs when something changes in the Plan's source repository (which may affect the outcome of a Plan build). There are two ways to trigger a Plan's build when code is updated:

  * "Pull strategy" — "Polling the Repository for changes" means that Bamboo will check out the source code on a regular basis and examine it for changes. If Bamboo detects a change, Bamboo triggers a build of your Plan. See Polling the Repository for Changes.
  * "Push strategy" — "Repository triggers the build when changes are committed" has the advantage of placing minimal load on your Bamboo server. However, it requires that your source repository is configured to fire an event to the Bamboo server (which the configured Plan will 'listen for'). See Repository Triggers the Build when Changes are Committed.

Scheduled Build

"Triggering a Plan build based on a 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 Single Daily Build.
  * Cron-Based Scheduling — "Cron Based Scheduling" allows you to run builds based on a schedule. This schedule can be configured using a Cron expression or via the Schedule Editor.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a Cron expression. See Cron Based Scheduling.

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 Plan Build Manually.

Related Topics

For more information on triggering Plan builds, see the following topics:

Triggering a Plan Build when Code is Updated
Triggering a Plan Build Based on a Schedule
Triggering a Plan Build Manually

Triggering a Plan Build when Code is Updated

"Triggering a Plan's build when code is updated" ensures that a Plan build only occurs when something changes in the Plan's source repository (which may affect the outcome of a Plan build). There are two ways to trigger a Plan's build when code is updated:

  * "Pull strategy" — "Polling the Repository for changes" means that Bamboo will check out the source code on a regular basis and examine it for changes. If Bamboo detects a change, Bamboo triggers a build of your Plan. See Polling the Repository for Changes.
  * "Push strategy" — "Repository triggers the build when changes are committed" has the advantage of placing minimal load on your Bamboo server. However, it requires that your source repository is configured to fire an event to the Bamboo server (which the configured Plan will 'listen for'). See Repository Triggers the Build when Changes are Committed.
Bamboo server. However, it requires that your source repository is configured to fire an event to the Bamboo server (which the configured Plan will 'listen for'). See Repository Triggers the Build when Changes are Committed.

You can also trigger builds based on a schedule or manually, rather than when code is updated.

Polling the Repository for Changes

"Polling the Repository for changes" means that Bamboo will check out the source code on a regular basis and examine it for changes. If Bamboo detects a change, Bamboo triggers a build of your Plan.

You can configure Bamboo to poll the repository for changes periodically (e.g. every 180 seconds) or based on a schedule (e.g. the second Sunday of every month at 5:00 am).

On this page:
- Triggering a Plan Build by Polling for Changes Periodically
- Triggering a Plan Build by Polling for Changes on a Schedule
- Notes

Triggering a Plan Build by Polling for Changes Periodically

To trigger a Plan's build by polling its source repository for changes:

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the 'Build Strategy' field, select 'Polling the Repository for changes' (see screenshot below).
3. In the 'Polling Strategy' field, select 'Periodically'.
4. In the 'Polling Frequency' field, specify how often (in seconds) Bamboo should check out the source code and examine it for changes.
5. Click the 'Save' button.

Screenshot above: Polling the repository for changes

Triggering a Plan Build by Polling for Changes on a Schedule

You can use the Schedule Editor to set up a polling schedule for your repository. Note, this is a schedule for polling your repository for changes (that can then trigger a build), not a schedule for directly triggering Plans.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a Cron expression.

To trigger a Plan's build by polling its source repository for changes:

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the 'Build Strategy' field, select 'Polling the Repository for changes' (see screenshots below).
3. In the 'Polling Strategy' field, select 'Scheduled'.
4. The 'Schedule' field will display the last configured schedule, e.g. 'The second Sunday of every month at 5:00 am'. Click the edit
In the Schedule Editor, select the type of polling schedule that you want to configure for your repository:

- **Daily** — Polls at specified intervals during the day, every day. For example, daily every 3 hours from 12:00am to 9:00am.
- **Days per Week** — Polls at specified intervals during the day, on specified days of the week. For example, each Sunday and Saturday every 3 hours from 12:00am to 9:00am.
- **Days per Month** — Polls at a specified time during the day, on specified days of the month. For example, the second Sunday of every month at 5:00 am.
- **Cron** — Use a Cron expression to specify when the repository will be polled. 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. For example, 0 0 1 ? * 1#2. For information on Cron expressions, see this FAQ: How do I construct a cron expression in Bamboo?

6. Click the ‘Save’ button.

**Screenshots: Scheduling polling for changes**

**Notes**

**Related Topics**

- Triggering a Plan Build when Code is Updated
- Triggering a Plan Build Based on a Schedule

**Repository Triggers the Build when Changes are Committed**

"Repository triggers the build when changes are committed" has the advantage of placing minimal load on your Bamboo server. However, it requires that your source repository is configured to fire an event to the Bamboo server (which the configured Plan will 'listen for').

Configuring the repository to trigger the build when changes are committed requires two changes:

1. Configuring your source repository, and
2. Configuring Bamboo to trigger a build on code check in.

**Step 1. Configuring your Source Repository**

To configure your source repository:

1. Configure your source code management system’s repository to send post-commit event messages which tell Bamboo to commence building plans that use this repository. These event messages are sent whenever a code commit has occurred.
   - For CVS, edit two files in the CVSROOT module: commitinfo and loginfo.
     - For commitinfo add a line like this:

   ```
   ^jira(//|$) /path/to/preCommit.sh
   ```
where "jira" is your module.

- For `loginfo` add a line like this:

  ```
  "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, JIRA being the project key and BRANCH or MAIN being the plan key.

- For **Subversion**, edit the Subversion repository's `hooks/post-commit` trigger file with something like:

  ```
  /path/to/postCommitBuildTrigger.sh http://bambooserver JIRA-MAIN JIRA-BRANCH
  ```

- For **Perforce**, add the script as a change-commit trigger.

  ```
  triggerName change-commit //myDepot/... "/usr/local/bin/postCommitBuildTrigger.sh
  http://bambooserver/ MYPLAN-DEFAULT"
  ```

- For **Git**, edit the Git repository's `.git/hooks/post-receive` trigger file with something like:

  ```
  /path/to/postCommitBuildTrigger.sh http://bambooserver JIRA-MAIN JIRA-BRANCH
  ```

- For more details about configuring these message events for CVS and Subversion, please refer to Configuring Source Code Management Triggers for Subversion.

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. Configuring Bamboo to Trigger a Build on Code Check In**

**Before you begin:**

- Triggering a build when there is no update — Bamboo will ignore the build triggers, if the local working copy and the repository copy have the same revision numbers. When testing your build triggers, please check that the local working copy is not the latest version in which case, no further action will be taken.

**To configure Bamboo to trigger a build on code check in:**

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the 'Build Strategy' field, select 'Repository triggers the build when changes are committed', which displays the 'Trigger IP Address' field (see screenshot below).
3. 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.
4. Click the 'Save' button.
Triggering a Plan Build Based on a Schedule

"Triggering a Plan build based on a 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 Single Daily Build.

- **Cron-Based Scheduling** — "Cron Based Scheduling" allows you to run builds based on a schedule. This schedule can be configured using a Cron expression or via the Schedule Editor. The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a Cron expression. See Cron Based Scheduling.

You can also trigger builds based on code changes or manually, rather than on a schedule.

Cron Based Scheduling

"Cron Based Scheduling" allows you to run builds based on a schedule. This schedule can be configured using a Cron expression or via the Schedule Editor.

The schedule can be daily (times per day), weekly (days per week), monthly (days per month) or based on a Cron expression.

To schedule a Plan build using a Cron expression:

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the 'Build Strategy' field, select 'Cron Based Scheduling' (see screenshot below).
3. The 'Schedule' field will display the last configured schedule, e.g. 'Each Sunday, Wednesday and Saturday every 30 minutes from 1:00 am to 2:00 am'. Click the edit icon next to this value to display the Schedule Editor.
4. In the Schedule Editor (see screenshots below), select the type of build schedule that you want to configure for your Plan:
   - ‘Daily’ — Build runs at specified intervals during the day, every day. For example, daily every 3 hours from 12:00am to 9:00am.
   - ‘Days per Week’ — Build runs at specified intervals during the day, on specified days of the week. For example, each Sunday and Saturday every 3 hours from 12:00am to 9:00am.
   - ‘Days per Month’ — Build runs at a specified time during the day, on specified days of the month. For example, the second Sunday of every month at 5:00 am
   - ‘Cron’ — Use a Cron expression to specify when the build will be run. 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. For example, 0 0 1 * 1#2. For information on Cron expressions, see this FAQ: How do I construct a cron expression in Bamboo?
5. Click the 'Save' button.
1. Daily
2. Week
3. Month
4. Cron

Screenshots: Scheduling polling for changes

**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 Plan build at a specified time each day:**

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the ‘Build Strategy’ field, select ‘Single daily build’ (see screenshot below).
3. 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.
4. Click the ‘Save’ button.
Triggering a Plan Build Manually

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. You can also run override any Plan variables with your own parameters when triggering a build manually. This is referred to as running a 'parameterised Plan build'.

If you want to trigger builds automatically rather than manually, you can trigger builds based on code changes or based on a schedule.

On this page:
- Starting a Plan Build Manually
- Starting a Parameterised Plan Build
- Configuring a Plan's Build to be Triggered Only Manually or via a Dependency
- Notes

Starting a Plan Build Manually

To start a Plan build manually:

1. Locate the relevant Plan on the Dashboard.
2. Click the 'Run' icon.

Starting a Parameterised Plan Build

If you trigger a Plan build manually, you can override any Plan variables by substituting in your own parameters.

Before you begin:
- You cannot run a parameterised Plan build if it is triggered automatically.
- The Plan must have Plan variables defined to run a parameterised Plan build.

To start a parameterised Plan build:

1. Navigate to the Plan Summary (not Plan configuration) for the desired Plan.
2. Click the 'Actions' menu, then click 'Run Parameterised Plan'.
3. In the 'Replace Build Variables' popup window, change the value for the Plan variables as desired and click 'Run'.
Configuring a Plan's Build to be Triggered Only Manually or via a Dependency

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 allow a Plan's build to be triggered only manually or via a dependency:

1. Navigate to the repository settings for desired Plan, as described on Specifying the Source Repository for a Plan.
2. In the 'Build Strategy' field, select 'Manual & dependent builds only' (see screenshot below).
3. Click the 'Save' button.

Related Topics:
- Stopping an Active Job Build
- Defining Plan Variables
- Reordering Jobs in the Build Queue

Reordering Jobs in the Build Queue

Bamboo automatically assigns a Plan's Jobs to the build queue when the Plan is triggered and no agents are available to run them. If you want to prioritise one Job build over another in the build queue, you can manually reorder these Jobs in the build queue. This will not force a Job build to run immediately, but will promote it in the build queue. Your Job build will still require an agent (which has the capabilities to meet the Job's requirements) to become available. Similarly, you can demote a Job build in the build queue if you do not need it to run.
urgently.
For details about how to re-order the build queue, please refer to Viewing Bamboo’s Current Activity.

Monitoring Job Builds

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

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

Configuring the Hanging Job Build Event

The hanging Job build event is thrown when a Bamboo determines that a Job’s 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 Job build.

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

This event is currently used by Bamboo to send notifications.

Configure the Hanging Job Build Event

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

Note, the hanging Job build criteria can be also be set for a specific Job, when specifying a Job’s builder. Job-level criteria will override the global criteria described on this page (including disabling this event).

To edit the hanging Job build event settings:

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Build Monitoring’ link in the left navigation column. The Build Monitoring page will be displayed.
3. Click the ‘Edit’ button and the Default Build Monitoring Criteria section of the Build Monitoring page will now be editable (see screenshot below). Update the values for the ‘Build Time Multiplier’ and ‘Log Quiet Time’ fields as desired.
4. Click the ‘Save’ button to update your settings.
Disabling the Hanging Build Event

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

Note, you cannot disable the hanging Job build event without disabling all build monitoring features for your Bamboo installation.

Changing How Often Bamboo Checks for Hung Job Builds

By default, Bamboo will check whether a hanging Job 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.

Notes

Related Topics

Configuring Notifications for a Plan and its Jobs
Configuring the Job Build Queue Timeout Event
Disabling Build Monitoring
Configuring Tasks

Configuring the Job Build Queue Timeout Event

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

This event is currently used by Bamboo to send notifications.
On this page:

- Configuring the Build Queue Timeout Event
- Disabling the Build Queue Timeout Event
- Changing How Often Bamboo Checks for Build Queue Timeouts
- Notes

**Configuring 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. The Build Monitoring page will be displayed.
3. Click the 'Edit' button and the Default Build Monitoring Criteria section of the Build Monitoring page will now be editable (see screenshot below). Update the value of the 'Build Queue Timeout' field (in minutes) as desired.
4. Click the 'Save' button to update your settings.

**Disabling 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.

**Changing 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.

Notes

Related Topics

Configuring Notifications for a Plan and its Jobs
Disabling Build Monitoring

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.

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 Job builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2.

Job builds can be run on these 'elastic agents' in the same way that Job builds are run non-elastic agents.

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

- 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.
About Elastic Bamboo

On this page:

- Conceptual Overview
- Key Terms
- Setting Up Elastic Bamboo

Conceptual Overview

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

A Job’s build can be run on an elastic agent, provided that the capabilities of the elastic agent meet the requirements of the Job. Bamboo will assign the relevant Job build to an available elastic agent from the build queue automatically, in the same way that Job builds are assigned to non-elastic agents. The elastic agent must already be running for a Job 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 Job’s build has completed running on an elastic agent, its results are made available (like those of any other Job build executed on a non-elastic agent). The elastic agent and instance will continue to run until they are 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 Job builds which it has run.

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.

Did you know you can configure Bamboo to start and shut down elastic instances automatically, based on build queue demands? Please refer to Configuring Elastic Bamboo for more information.
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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system. Each elastic image in the Amazon S3 has its own unique identifier, known as an AMI ID. 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 a running instance of an elastic image. One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one elastic agent is created on that instance. Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down:</td>
</tr>
</tbody>
</table>

• Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist
• Any customisations to the instance itself will also be lost.

The Amazon Elastic Block Store can provide persistent storage for your elastic instances. |
| 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 elastic instance of an elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from. |

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 Job 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 Managing your Elastic Image Configurations 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 costs only.
• 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 Job builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Job builds can be run on these ‘elastic agents’ in the same way that Job 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 Plan Build and its Jobs
- 5. Shut Down your 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 and Enabling 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
4. 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 Job 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 Plan build and its Jobs 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 Plan Build and its Jobs

To run a Plan build and its Jobs on your elastic agent, you must set up a Plan with its Default Job (plus any other optional Jobs) all of whose requirements can meet your elastic agent's capabilities. Elastic agents inherit the capabilities of the image 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 Plan so that its Jobs' requirements can only be met by the elastic agent's capabilities. This will ensure that the Jobs' builds run on your elastic agent. If you cannot set up your Jobs' requirements to meet your elastic agent's capabilities, 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.

Job builds on elastic agents are run just like Job builds on any other agent. You will see the progress of your Job 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 Job build by configuring Elastic Bamboo to use Amazon's Elastic Block Store (EBS).

5. Shut Down your Elastic Instance

When your Job build(s) complete 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 Job build with Elastic Bamboo.

Further Information

You may be interested in reading the following related topics below to help you manage and improve Elastic Bamboo's handling of Job builds:

- Configuring Elastic Instances to use the EBS — information on configuring Elastic Bamboo to use the Amazon Elastic Block Store (EBS) to improve Job 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 Job builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Job builds can be run on these 'elastic agents' in the same way that Job 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 the following logical groups:

- Amazon Web Services (AWS) Account Settings — AWS account details for Elastic Bamboo.
- Elastic Bamboo AWS Settings — AWS configuration settings for the Bamboo installation, including Elastic Block Store (EBS) settings.
- Automatic Elastic Instance Management — Settings defining how Bamboo automatically starts and shuts down elastic instances.
All of these 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.

⚠️ If you have disabled remote agent support, you must enable it before you can enable Elastic Bamboo. Refer to Disabling and Enabling Remote Agents Support for instructions on how to enable remote agent support.

### On this page:
- Amazon Web Services (AWS) Account Settings
- Elastic Bamboo Global Settings
- Configuring Elastic Bamboo to Use EC2 Spot Instances
- Elastic Bamboo AWS Settings
- Automatic Elastic Instance Management
- Notes

#### Screenshots above: Elastic Bamboo Configuration

### Amazon Web Services (AWS) Account 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.

Before you begin:
- Please note, 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.
- Please note, if you change your AWS account details, Bamboo will stop all elastic agents that are currently running.

To configure your Amazon Web Services (AWS) account details:

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.
3. Click the 'Edit' button to edit your Elastic Bamboo configuration (see screenshot above).
4. Enter or update your 'AWS Access Key ID'.
5. Check the 'Change AWS Secret Access Key?' checkbox, if you wish to enter or update your AWS Secret Access Key.
6. Enter or update 'AWS Secret Access Key' as desired (see Notes section below).
7. Click the 'Save' button to update your Account Details. If the details you have provided are invalid, a validation message will display and you will be required to update your account details to correct the error.

### Elastic Bamboo Global Settings

Elastic Bamboo provides you with a number of global configuration options to help you optimise EC2 usage for your Bamboo Job 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.
To configure Elastic Bamboo to use EC2 Spot Instances:

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.
3. Click the 'Edit' button to edit your Elastic Bamboo configuration.
4. In the 'Spot instances configuration' section, update these settings as desired:
   - 'Enable support for spot instances' — Tick this checkbox to enable support for spot instances.
   - 'Fallback to a regular instance after' — Enter the time (in minutes) after which Elastic Bamboo will fall back to using a regular instance, if a spot instance has not become available.
   - 'Your current bid levels (per hour)' — Fill out this table with your bids. The bids are categorised by EC2 instance type and operating system.
5. Click the 'Save' button.

Elastic Bamboo AWS Settings

These settings allow you to specify your AWS configuration settings in Bamboo so that Bamboo can operate elastic instances through your AWS account. This section includes settings that are used to configure elastic instances to work with the Amazon Elastic Block Store (EBS).

Using EBS with your elastic instances can significantly reduce the amount of data transfer required to run a Job 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 the 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.
3. Click the 'Edit' button to edit your account details.
4. In the 'Elastic Bamboo AWS Settings' section of the 'Elastic Bamboo Configuration' screen, 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 Job build information in a snapshot. 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.

5. 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.

Automatic Elastic Instance Management

The Automatic Elastic Instance Management feature allows Bamboo to start and shut down elastic instances automatically (based on build queue demands), so that you do not have to perform these actions manually. This feature reduces Bamboo administration overhead and can help minimise your overall elastic instance usage costs.

If a Job's requirements cannot be met by any available online agents, this feature will start any elastic instance whose elastic agent has the
capabilities to execute the Job, so that the Job's build can be generated. Regardless of how an elastic instance was started, all elastic instances will be shut down based on the settings specified below.

To configure Bamboo’s 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.
3. Click the 'Edit' button to edit your account details.
4. In the 'Automatic Elastic Instance Management' section of the 'Elastic Bamboo Configuration' screen, choose one of Bamboo’s elastic instance management settings:
   - **Preset Options** — You can choose from the following elastic instance management presets. Each of these presets sets predefined values for the five criteria described in the 'Custom' user-defined options (below). (Bear in mind that both the 'Aggressive' and 'Passive' presets have trade-offs.)
     - 'Default' — Balances build queue clearance rates with elastic instance usage costs.
     - 'Aggressive' — Favours higher build queue clearance rates but with higher elastic instance usage costs.
     - 'Passive' — Favours lower instance usage costs but with lower build queue clearance rates.
   - **User-defined Options** — If you want more control over this feature than what’s provided by the presets (above), or you want to disable elastic instance management, choose the appropriate option.
     - 'Custom' — Choose this option to individually specify the following five criteria, which are used to manage Bamboo’s elastic images.
       - 'Idle Agent Shutdown Delay' — Specify the number of minutes that an elastic agent must be idle before Bamboo shuts down the elastic instance running that agent.
       - 'Maximum Number of Instances to Start at Once' — Specify the maximum number of elastic instances Bamboo can start in one go. Bamboo only starts this maximum number of elastic instances on a 'per minute' basis.
       - The remaining three criteria are thresholds for starting new elastic instances. New elastic instances will only be started when all of the following thresholds have been reached:
         - 'Maximum Number of Builds in Queue Threshold' — Specify the maximum threshold for the total number of Job builds in the queue.
         - 'Number of Elastic Builds in Queue Threshold' — Specify the threshold for the maximum number of Job builds in the queue that can be executed on elastic instances.
         - 'Average Queue Time Threshold' — Specify the maximum threshold for the average number of minutes that Job builds have been waiting in a queue.
   - 'Disabled' — Choose this option to completely disable Bamboo’s elastic instance management feature.
5. Click the 'Save' button to update your elastic instance settings.

Notes

- **AWS Access Key ID and AWS Secret Access Key** — Your AWS Access Key ID and AWS Secret Access Key are used together to identify yourself when accessing Amazon EC2 services. If you are unsure what your AWS Account ID and AWS Secret Access Key are, please refer to the Amazon documentation on AWS access identifiers.

Related Topics

- Generating your AWS Private Key File and Certificate File
- Configuring Elastic Instances to use the EBS
- Managing Elastic Bamboo
- Disabling Elastic Bamboo

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
- Notes

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 above: Downloading the generated AWS private key file and certificate file](image)

**Notes**

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.

**Related Topics**

**Configuring Elastic Bamboo**

**Configuring Elastic Instances to use the EBS**

The Amazon Elastic Block Store (EBS) provides 'EBS volumes' which can attach to EC2 instances. EBS volumes (and the 'EBS snapshots'
created from these volumes) provide persistent storage for your elastic instances.

If you have relatively static resources required for building your Bamboo Jobs (such as, source code checkouts and Maven repository artifacts), you can add these to an EBS volume. From this volume, you can create an EBS snapshot, which effectively records the 'state' of an EBS volume at a given point in time.

After setting up an EBS snapshot, you can then associate it with an elastic image configuration. When this elastic image is started:

- its elastic instance will be started, along with the EBS volume (derived from the EBS snapshot associated with the elastic image) and
- this EBS volume will be attached to this elastic instance

Any build resources (added to the EBS volume prior to creating its snapshot) will be available to this elastic instance.

Why should I use the EBS with Elastic Bamboo?

Because an elastic instance is stateless, so is the elastic agent that runs on it. Hence, every time an elastic instance is restarted from the same image:

- Any resources that its elastic agent must retrieve externally (for example, Maven repository artifacts), must be downloaded in their entirety.
- Full checkouts must be performed by elastic agents when new Jobs are built.

Therefore, you can use the EBS to store these external resources in an EBS volume and snapshot so that they do not have to be downloaded or source code checked out each time you start up an elastic instance from an image. If your Jobs rely heavily on downloading such resources and/or you are not performing clean builds each time, the EBS may significantly improve your build times.

Additionally, the EBS provides an easy mechanism for customising elastic agents, rather than you having to create a custom elastic image from scratch (with your own elastic agent capabilities). For example, you could upload files and scripts to your EBS volume that would install resources such as PostgreSQL databases for your elastic agents, which will be available when the agent's elastic instance is started.

### Notes

#### Important EBS Directories and Files

- **Account Certificate File**
  - Upload AWS account identifiers to new elastic instances.
- **Account Private Key File**
  - Certificate file to your Bamboo server yet, please see Generating your AWS Private Key File and Certificate File for instructions.
- **EBS Directories and Files**
  - directories and files for this standard volume structure are detailed in the [ Important EBS Directories and Files section](#)

#### 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).
3. Start a single elastic instance via Bamboo. See [Starting an Elastic Instance](#) for instructions.
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-eb <filename>` 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. The EBS volume is attached to the elastic instance, so accessing your elastic instance via SSH will give you full access to the EBS volume (see [ Important EBS Directories and Files](#) below).

   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 snapshoted.

   f. Run `generateSnapshot.sh` — This script unmounts and detaches the volume, before creating a snapshot based on the EBS 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 started from that image will have EBS volumes attached to them. You can associate different snapshots with different elastic image configurations.

**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.

![Screenshot above: AWS Console (click to view full-size image)](image-url)

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 'Manage Elastic 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).

![Screenshot above: Adding EBS Snapshot details to an Elastic Image Configuration (click to view full-size image)](image-url)

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:**

1. Start a single elastic instance via Bamboo. See **Starting an Elastic Instance** for instructions.

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:
   - All the scripts described below are bundled with Bamboo.
   - 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.
     - The device can not unmount if any terminals are currently in the mounted 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/.m2/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.

**Notes**

Related Topics

Configuring Elastic Bamboo

Populating your EBS volume

This page is intended to complement the instructions for Configuring Elastic Instances to use the 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 for Elastic Agents
- Setting up Selenium on Elastic Agents
- Notes

**Uploading Maven 2 Repository Data**

You can upload Maven 2 repository data to your EBS volume, so that it does not have to be downloaded every time an elastic agent (running on an instance which 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 the 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 does not have to be downloaded every time an elastic agent (running on an instance which 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 the EBS). This is a faster and more reliable method of populating your volume, if you are using Ant.

**Setting Up PostgreSQL for Elastic Agents**

You can upload scripts to your EBS volume so that the elastic agent started on any elastic instances which use this EBS volume, will have PostgreSQL automatically installed.

> These elastic instances must be started from an elastic image which is associated with an EBS snapshot derived from this EBS volume.

To set up the automatic installation of PostgreSQL on your EBS volume for elastic agents, you will need to create the following script:

```bash
setupPostgreSQL.sh
```
This script uses the package management tools provided by Fedora to install and configure PostgreSQL on the agent when it's started.

1. **Uses `yum` to install the PostgreSQL server packages.** Details on the yum tool can be found in the [Fedora Software Management Guide](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](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](Configuring Elastic Agent Capabilities) for detailed instructions.

**Setting up Selenium on Elastic Agents**

You can upload scripts to your EBS volume so that the elastic agent started on any elastic instances which use the EBS volume, will be able to run Selenium tests.

> These elastic instances must be started from an elastic image which is associated with an EBS snapshot derived from this EBS volume.

To set up elastic agents to support Selenium test, you will need to create the following script:

```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

cat /mnt/bamboo-ebs/bin/thunderbird-2.0.0.20.tar.gz | tar -xvf -C /opt

grep "^"LD_LIBRARY_PATH /home/bamboo/.bashrc
if [ "$?” = “1” ] then
echo "LD_LIBRARY_PATH=/opt/firefox" >> /home/bamboo/.bashrc
fi
```

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](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`)

256
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.

Notes

Related Topics

Configuring Elastic Instances to use the EBS

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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operating system.

Each elastic image in the Amazon S3 has its own unique identifier, known as an AMI ID.

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 Jobs that an image can build, see Viewing an Elastic Image.
- To associate an elastic image with your Bamboo installation, see Managing your Elastic Image Configurations.
- 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.

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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operating system.

Each elastic image in the Amazon S3 has its own unique identifier, known as an AMI ID.

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 Jobs that an image can build — you can also view the Jobs that an image is capable of building (via the elastic agent created from the image), similar to how you view the Jobs that an agent is capable of viewing. Read more about viewing the Jobs that an agent can build and determining which agents can build which Jobs.
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 be displayed.
3. Click the name of the image that you want to view, e.g. ‘Default’, or the image’s ‘View’ link in the Operations column. The elastic image configuration will be displayed (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 the 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.
Manage Elastic Image Configurations - Maven 2.1 Image

Elastic Image Configuration

- **Name:** Maven 2.1 Image
- **Contains:** Maven 2.1 and the necessary bits for Selenium 2
- **AMI ID:** ami-0ab54563
- **EBS Snapshot ID:** snap-68204c00
- **Instance Type:** High-CPU Medium
- **Availability Zone Preference:** Default (chosen by EC2)
- **Active Instances:** 13

Start Instances | Disable | Edit

Elastic Image Capabilities

- **Add Capability | Revert to Default Capabilities**

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom. You can use this page to view, add and delete capabilities associated with this Elastic Image Configuration. Any existing elastic instances will need to be restarted to pick up these changes.

The following capabilities exist on Elastic Agents running on an instance of this Image:

### Custom

Custom capabilities are key-value pairs that define particular characteristics of an agent (e.g., `operating_system=WindowsXP, Test build=true`). For an agent to be able to build a job, both the 'Key' and 'Value' must match the job'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>Builder 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 2 (Maven 2.x)</td>
<td>/opt/maven-2.0</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x)</td>
<td>/opt/maven-2.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x)</td>
<td>/opt/maven-2.2</td>
<td>View</td>
</tr>
</tbody>
</table>

### JDK

JDK capabilities define the JDKs which are available to your build plans.

<table>
<thead>
<tr>
<th>JDK 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 1.8</td>
<td>/opt/jdk-6</td>
<td>View</td>
</tr>
</tbody>
</table>

### Mercurial

The path to the Mercurial executable (e.g., `C:\Program Files (x86)\Mercurial\hg.exe` or `/usr/local/bin/hg`)

<table>
<thead>
<tr>
<th>Executable</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercurial</td>
<td>/usr/bin/hg</td>
<td>View</td>
</tr>
</tbody>
</table>
**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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image in the Amazon S3 has its own unique identifier, known as an **AMI ID**.

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 Custom Elastic Images with Bamboo

**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 a Ubuntu operating system-based elastic image 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 be displayed.
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 a description for your image, which is used in Bamboo only.
   - **AMI ID** — Here, you should enter the AMI ID obtained as an output from step 6 of the 'Creating a Custom Elastic Image' instructions. You can also view the AMI IDs of elastic images via the AWS console.
   - **Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances** (optional) — Select this option if you want the elastic instances started from this image to use the EBS. Read more about Configuring Elastic Instances to use the EBS.
     - **EBS Snapshot ID** (displayed if 'Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances' is selected) — If you selected 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.
You can customise the agent capabilities of an elastic image that is already associated with Bamboo. The initial process is similar to that of associating a custom elastic image with Bamboo (above). Here, however, you use the AMI ID of an image already associated with Bamboo — most commonly the default image.

To create an elastic image with customised agent capabilities:

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 be displayed.
3. Click the name of the image that you want to view, e.g. ‘Default’, or the image’s ‘View’ link in the Operations column. The elastic image configuration will be displayed (see screenshot above).
4. Enter the details of your custom elastic image in the panel under the Create Elastic Image Configuration header:
   - ‘Name’ — Enter a name for your elastic image whose agent capabilities you wish to customise.
   - ‘Description’ (optional) — Enter a description for your image, which is used in Bamboo only.
   - ‘AMI ID’ — Copy and paste the AMI ID of an image already associated with Bamboo (table at the top of the page). The default image is the most common elastic image used for configuring custom agent capabilities because for each Bamboo release, the default image:
     - is updated with the correct elastic agent version and newer default capabilities, and
     - is already associated with Bamboo.
   - ‘Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances’ (optional) — Select this

Screenshot above: Manage your Elastic Image Configurations
option if you want elastic instances started from this image to use the EBS. Read more about Configuring Elastic Instances to use the EBS.

- "EBS Snapshot ID" (displayed if 'Automatically attach an Amazon Elastic Block Store (EBS) volume to new elastic instances' is selected) — If you selected 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.

5. You now have a new elastic image configuration based off an existing elastic image. Follow the procedure on Configuring Elastic Agent Capabilities to customise this elastic image's agent capabilities.

Notes
Related Topics

Managing your Elastic Images

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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image in the Amazon S3 has its own unique identifier, known as an AMI ID.

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 Amazon Machine Images (AMIs) 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.

Before you begin:

- This is not a trivial procedure. 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 the EBS. This is a much simpler option. If you are having problems, please don’t hesitate to contact us for further help.

- Please note, Atlassian does not support custom elastic images. Consider customising the elastic agents started from your existing image instead.

- 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 instance). 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
- 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:

```bash
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
1. Registering Key Pair

- **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 follow the command to do so:

```
$EC2_HOME/bin/ec2-add-keypair <key_pair_name>
```

The content of the private key will be displayed 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:

- **Atlassian's AMI** — 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.
- **Amazon’s AMI** — this is the base Fedora 8 image provided by Amazon. It does not have any Bamboo prerequisites installed.
- **Alestic’s AMI** — 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 and you should consult the Alestic site for the name of this AMI.

The names of Atlassian's and Amazon's AMIs (and hence, their IDs) change with each release of Bamboo, including both major and minor (or point) releases.

Therefore, to find out the AMI ID for your version of Bamboo:

1. Open the following URL:
   
   https://maven.atlassian.com/content/repositories/atlassian-public/com/atlassian/bamboo/atlassian-bamboo-agent-elastic-image-default/

2. On the resulting directory page, click on the link that represents the version of Bamboo you are currently running. For example, if you are running Bamboo 2.6.1, click on the '2.6.1' link. Another directory page opens, listing a .ami, a .pom and some additional checksum files.

   Do not click on a version number link that contains 'mX', 'rcX' or 'betaX' (where 'X' is a number), since these relate to publicly available developmental releases of Bamboo.

3. Do one of the following to obtain the name of the AMI you require:

   - **Atlassian’s AMI** — Open the atlassian-bamboo-agent-elastic-image-default-A.B.C.ami file (where A.B.C relate to numbers of your version of Bamboo).
   - The content of this file is the name of this AMI.
   - **Amazon’s AMI** — Open the atlassian-bamboo-agent-elastic-image-default-A.B.C.pom XML file (where A.B.C relate to numbers of your version of Bamboo).
   - Locate the ec2ImageID element in this file. The text content of this element is the name of this AMI.

   To quickly access Atlassian’s AMI ID for a currently-running version of Bamboo, open that Bamboo site in a web browser and access its Image Configurations page (see Managing your Elastic Image Configurations for more information). The AMI ID of Atlassian’s AMI is listed in the table at the top of this page, in the row whose image configuration contains ‘(default)’ in its name.

3. Starting an Instance

After you have selected an existing AMI to customise, the next step is to start an instance of the AMI.

3.1 Starting an instance of Atlassian's default AMI

If you chose to customise Atlassian's default AMI, you will have to start the instance from the admin section of Bamboo. See Starting an Elastic Instance.

On start up, the Bamboo agent on Atlassian's AMI checks to see if it was started from a Bamboo server, and immediately shuts itself down if it was not. Because of this, Atlassian's default AMI cannot be started using the command line ec2 tools.
Once started, see Accessing an Elastic Instance for details on how to access the running instance.

3.2 Starting an instance from the command line

Use the `ec2-run-instances` command to start your instance, as follows:

```bash
$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:

```bash
$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

If you started the instance from Bamboo, see Accessing an Elastic Instance for details on how to access the running 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:

```bash
e2-describe-instances <instance_name>
```

For example, if you wanted to find the address of instance `i-25b86743`, you would enter:

```bash
e2-describe-instances i-25b86743
```

This command would produce the following command-line output similar to this:

```
RESERVATION r-790f7210 121852097033 default
INSTANCE i-25b86743 ami-e55bbd8c ec2-174-129-94-241.compute-1.amazonaws.com
  domU-12-31-39-04-38-87.compute-1.internal running elasticbamboo 0 m1.small
  2009-06-24T12:36:20+0000 us-east-1c aki-a71cf9ce ari-a51cf9cc
  monitoring-disabled
```

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.ppk` 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
   | `wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools-1.3-30349.zip` | Download Amazon EC2 API tools
   | `unzip ec2-api-tools-1.3-30349.zip` | Unzip the downloaded file
   | `mv ec2-api-tools-1.3-30349 /usr/local/ec2/ec2-api-tools` | Move the tools to the EC2 directory

2. Java 6 and Java 5

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
   | `# get Java 6` | Install Java 6
   | `wget "http://download.java.net/dlj/binaries/jdk-6u11-dlj-linux-i586.bin"` | Download Java 6
   | `mkdir unbundle-jdk-6` | Create a directory to unbundle the Java file
   | `cd unbundle-jdk-6` | Navigate to the directory
   | `sh ../jdk-6u11-dlj-linux-i586.bin --accept-license` | Unbundle and accept the license
   | `cd ..` | Return to the parent directory
   | `# get Java 5` | Install Java 5
   | `wget "http://download.java.net/dlj/binaries/jdk-5.0u17-dlj-linux-i586.bin"` | Download Java 5
   | `mkdir unbundle-jdk-5` | Create a directory to unbundle the Java file
   | `cd unbundle-jdk-5` | Navigate to the directory
   | `sh ../jdk-5.0u17-dlj-linux-i586.bin --accept-license` | Unbundle and accept the license
   | `cd ..` | Return to the parent directory
   | `# Build the JDKs` | Build the JDKs
   | `curl *https://jdk-distros.dev.java.net/source/browse/jdk-distros/trunk/utils/construct.sh?content-type=text/plain&rev=148" > construct.sh` | Download the construct script
   | `chmod +x construct.sh` | Make the script executable
   | `./construct.sh unbundle-jdk-6 /opt/jdk-6 /opt/jre-6` | Unbundle and build Java 6
   | `./construct.sh unbundle-jdk-5 /opt/jdk-5 /opt/jre-5` | Unbundle and build Java 5

3. Ant

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
   | `wget "http://www.apache.org/dist/ant/binaries/apache-ant-1.7.1-bin.tar.bz2"` | Download Ant
   | `tar xjC /opt -f apache-ant-1.7.1-bin.tar.bz2` | Extract the Ant archive

4. Maven 2

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
   | `wget "http://www.apache.org/dist/maven/binaries/apache-maven-2.0.10-bin.tar.bz2"` | Download Maven 2
   | `tar xjC /opt -f apache-maven-2.0.10-bin.tar.bz2` | Extract the Maven 2 archive
   | `mv /opt/apache-maven-2.0.10 /opt/maven-2.0` | Move the Maven 2 directory
   | `ln -fs /opt/maven-2.0 /opt/apache-maven-2.0.10` | Link the directories

5. (optional) Maven 1.1

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
   | `wget "http://archive.apache.org/dist/maven/binaries/maven-1.1.tar.bz2"` | Download Maven 1.1
   | `tar xjC /opt -f maven-1.1.tar.bz2` | Extract the Maven 1.1 archive

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.

   **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').**

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...
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 -m bamboo
```

After you have created the 'bamboo' user, you need to download the Bamboo agent elastic assembly. The latest version of the Bamboo agent elastic assembly can be downloaded from the Bamboo download center (choose the desired tab and click 'Show all'). If you need an earlier version of the Bamboo agent elastic assembly, you can download it from the Bamboo Archive Downloads page or for a more complete list, our maven repository.

Once you have downloaded the Bamboo agent elastic assembly, you need to copy it onto your instance by using the following commands:

```
cd <BAMBOO_AGENT_ELASTIC_ASSEMBLY_DOWNLOAD_DIRECTORY>/webapp
scp -i <private_key_file> atlassian-bamboo-agent-elastic-assembly-<x.x.x>.tar.gz root@<instance_address>:/mnt
```

where `<private_key_file>` is the private key file generated in '1. Requirements', `<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. **Bamboo Agent binaries**

   ```
tar xzC /opt -f atlassian-bamboo-agent-elastic-assembly-<x.x.x>.tar.gz
ln -s /opt/bamboo-elastic-agent-<x.x.x> /opt/bamboo-elastic-agent
```

   where `<x.x.x>` is the Bamboo version you are running (e.g. 2.4.1)

   `atlassian-bamboo-agent-elastic-assembly-2.3.tar.gz` is available in the `webapp` sub-directory of the Bamboo installation directory.

2. **Bamboo Agent default capabilities definition file**

   ```
   mv bamboo-elastic-agent/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:

   ```
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:

   ```
   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:

   ```
   chown -R bamboo:bamboo /home/bamboo/
   ```
4. Configure path variables
Create a file `profile.sh` in your instance's `/mnt` directory. This file contains the default Elastic 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:$ANT_HOME/bin:$PATH
```

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` is correctly customised for your instance, 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 /opt/bamboo-elastic-agent/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:

```bash
echo bamboo-<x.x.x>  >> /opt/bamboo-elastic-agent/motd
mv /opt/bamboo-elastic-agent/motd /etc/motd
rm -f /root/firstlogin /etc/ssh/ssh_host_dsa_key /etc/ssh/ssh_host_rsa_key.pub /etc/ssh/ssh_host_rsa_key /root/.ssh/authorized_keys
touch /root/firstrun
```

where `<x.x.x>` is the Bamboo version you are running (e.g. 2.4.1)

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:

```bash
/usr/local/bin/ec2-bundle-vol -c $EC2_CERT -k $EC2_PRIVATE_KEY -u <amazon_account_number> -p <elastic_image_name> --batch --debug
```

where `<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:

```bash
/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:
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.
- **Bamboo Developer Forums** — please feel free to discuss any useful tips or issues regarding this process in the Bamboo Developer Forums.

**Upgrading the Agent for your Custom Elastic Image**

If you are using a custom elastic image with your Bamboo installation, you need to ensure that the agent running on your custom elastic image is the same version as your Bamboo server (down to the point version).

For example, if you are running a Bamboo 2.6 server, the agents running on your custom elastic image also need to be version 2.6.

**On this page:**

- Download the Bamboo Agent
- Start Up an Instance of your Custom Image
- Replace the Existing Agent with the New Agent
- Create an Image of your Customised Instance
- Associate the new Custom Image with Bamboo
- Notes

**Download the Bamboo Agent**

The latest version of the Bamboo agent elastic assembly can be downloaded from the Bamboo download center (choose the desired tab and click ‘Show all’). If you need an earlier version of the Bamboo agent elastic assembly, you can download it from the Bamboo Archive Downloads page or for a more complete list, our maven repository.

**Start Up an Instance of your Custom Image**

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 previously (see ‘1. Requirements of ‘Creating a Custom Elastic Image’). Please note, 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:
This command would produce the following command-line output:

```
INSTACE 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 (see 'Accessing your Instance' in 'Creating a Custom Elastic Image') 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.

### Replace the Existing Agent with the New Agent

Once you have started an instance of your custom image, you can replace the existing agent with a new agent. To do this, copy the Bamboo agent elastic assembly that you previously downloaded onto your instance, by using the following commands:

```
cd <BAMBOO_AGENT_ELASTIC_ASSEMBLY_DOWNLOAD_DIRECTORY>
scp -i <private_key_file> atlassian-bamboo-agent-elastic-assembly-<x.x.x>.tar.gz root@<instance_address>:/mnt
```

where `<private_key_file>` is the private key file generated when you created your custom image, `<instance_address>` is the address of your instance (see 'Accessing your Instance' in 'Creating a Custom Elastic Image') and `<BAMBOO_STANDALONE_DIRECTORY>` is your Bamboo Installation directory.

You then need to unzip the agent tarball by using ssh to access your instance and then running the following command:

```
tar -zxvf file.tar.gz
```

### Create 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

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 $EC2_CERT -k $EC2_PRIVATE_KEY -u <amazon_account_number> --batch --debug
```

where `<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.

Assume the new Custom Image with Bamboo

Finally, 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.

Notes
Related topics
Managing your Elastic Image Configurations

Updating Elastic Images for Bamboo Upgrades

Various updates to default packages and capabilities are made to the default image with each major release of Bamboo.

Therefore, if you are using either a:

- custom elastic image, or
- an elastic image with customised agent capabilities

then to ensure this elastic image acquires these package/capability updates, use the flow chart below to update your elastic image.

⚠️ Use this flowchart only after Bamboo has been upgraded. For each elastic image you wish to update, follow this flow chart from the start.
Elastic Images with Customised Capabilities:

This flow chart assumes that all elastic images with customised agent capabilities are based off the default image. Please check the default image page to identify the packages and related capabilities available in the default image for Bamboo 3.1.x.

Managing your Elastic Instances

An elastic instance is a running instance of an elastic image. One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one elastic agent is created on that instance.

The following list directs you to details on managing elastic instances manually in Bamboo. However, you can configure Bamboo to automatically manage your elastic instances. Please refer to Automatic Elastic Instance Management for more information.

- 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 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 a running instance of an elastic image. One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one elastic agent is created on that instance.

Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down:

- Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist
- Any customisations to the instance itself will also be lost.

The Amazon Elastic Block Store can provide persistent storage for your elastic instances.

On this page:

- Viewing an Elastic Instance
- Notes

Viewing an Elastic Instance

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. Click the 'Elastic Agent on <instance_name>' link to view the elastic agent. If the agent is running a Job, the Job's Key will be shown in brackets after the elastic agent name.
  - **Current Availability Zone** — the availability zone that your elastic instance is running in. 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.
  - **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.

- **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.
**Elastic Bamboo › Instances › i-07ec936b**

**Information**

This is an instance running on the Amazon EC2 compute cloud. You can get more extensive information about this instance from the Amazon AWS Web Console.

- **Current status**: ✔ Running
- **Public DNS**: ec2-184-72-81-22.compute-1.amazonaws.com
- **IP**: 184.72.81.22
- **Start Time**: 14/02/11 10:57 AM (41 minutes ago)
- **Elastic Agent**: Elastic Agent on i-07ec936b (Idle)
  
  This is the Bamboo agent that is running in this instance in EC2.

- **Current Availability Zone**: us-east-1c (chosen by EC2)

**Attached Volumes**

<table>
<thead>
<tr>
<th>Volume ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>vol-fe3-5396</td>
</tr>
</tbody>
</table>

**Configuration**

- **Configuration**: Maven 2.1 Image
  
  Contains Maven 2.1 and the necessary bits for Selenium 2

- **AMI ID**: ami-0ab54563
- **EBS Snapshot ID**: snap-58204c00
- **Instance Type**: High-CPU Medium

**SSH Access**

You can SSH into this instance in the EC2. To do this, simply execute the following command from the bamboo server home directory. Bamboo cannot find the elasticbamboo.pk file on the server. For more information on where to find your elasticbamboo.pk file see our online documentation.

```
ssh -i elasticbamboo.pk root@ec2-184-72-81-22.compute-1.amazonaws.com
```

**Accessing Logs**

You can use SCP to download the logs from this EC2 instance. To do this, simply execute the following command.

```
scp -i elasticbamboo.pk root@ec2-184-72-81-22.compute-1.amazonaws.com:/home/bamboo/bamboo-elastic-agent.out .`
```

**Amazon EC2 Console**

You can manage your EC2 instances using the Amazon EC2 Console. Once you've logged in through the console, you can access logs for this instance directly. Logs are reproduced below in the iframe below. (Note that the logs are usually a little delayed).
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.

On this page:
- Accessing your Elastic Instance via SSH
- Accessing your Elastic Instance via SSH
- Notes

Accessing your Elastic Instance via SSH

To access your elastic instance via SSH:

1. Navigate to the desired elastic instance, as described on Viewing an Elastic Instance.
2. Copy the command text listed under the 'SSH Access' section. It will be similar to the following example command text:
   
   ```bash
   ssh -i /opt/bamboo/home/xml-data/configuration/elasticbamboo.pk
   root@ec2-68-111-185-197.compute-1.amazonaws.com
   ```
3. 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.'

Accessing your Elastic Instance via SSH

Note, you can also use SCP to upload files to your elastic instance.

To access your elastic instance via SCP:

1. Navigate to the desired elastic instance, as described on Viewing an Elastic Instance.
2. Copy the command text listed under the 'Accessing Logs' section. It will be similar to the following example command text:
   
   ```bash
   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 ./
   ```
3. Execute the text in your terminal to download the logs from your elastic instance.

Notes
- 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.

Related Topics
- Managing your Elastic Instances

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 Job 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.
Starting an Elastic 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).
   a. Enter the number of new elastic instances you would like to start in the 'Number of instances' field.
   b. Select the elastic image configuration that you would like to start your instance from in the 'Elastic Image Configuration Name' dropdown.
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.
   b. Your elastic instances will then display with a status of 'Pending' while they start up. This generally takes a few minutes.
   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.
   d. Once the elastic agents have started, they will display a status of 'Online'.

Notes

* What if my elastic agent doesn’t start? Bamboo has a set period of time that it waits for the agent to start on an elastic instance. If no response is received by the end of this time period, Bamboo will shut down the elastic instance. You can configure this time period by modifying the following system property (default is 600):
  bamboo.agent.elastic.startupTimeoutSeconds
Read Configuring System Properties for instructions on how to set a system property.

Related Topics

Managing your Elastic Instances

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 Job builds during regular busy periods.

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.

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>Trigger disabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 Jul 2010, 1:00:00 PM (in 3 days)</td>
<td>Each Monday at 9:00 am</td>
<td>Default</td>
<td>0</td>
<td>Edit</td>
</tr>
<tr>
<td>Trigger disabled</td>
<td></td>
<td>Maven 2.1 Image</td>
<td>3</td>
<td>Edit</td>
</tr>
</tbody>
</table>

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.

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. For example, if you specify 'at least' in this field and '3' in the 'Active Instances' field, the schedule will ensure that at least three elastic instances are active when the schedule triggers (and will start instances to meet this
Shutting Down an Elastic Instance

Before you begin:

- Please ensure that the agent on an elastic instance is not running a Job build, before shutting down the instance. Any Job builds running on the agent will be abandoned when you shut down the 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.

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**

Before you begin:

- Please ensure that the agent on an elastic instance is not running a Job build, before shutting down the instance. Any Job builds running on the agent will be abandoned when you shut down the elastic instance.

**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.

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.

**Notes**

**Related Topics**

**Managing your Elastic Instances**

**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 elastic instance of an 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 configure your elastic agent's capabilities, see Configuring Elastic Agent Capabilities.
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 elastic instance of an elastic image. An elastic agent inherits its capabilities from the elastic image that it was created from.

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.

On this page:
- Viewing Your Elastic Agents
- Notes

Viewing Your Elastic Agents

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'
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.

Select All, None, Idle, Disabled Action: Delete | Disable | Enable

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Agent</td>
<td>Building - SANDBOX-RFIN-JOB1-22</td>
<td>View</td>
</tr>
<tr>
<td>Reporting agent</td>
<td>Idle (Disabled)</td>
<td>View</td>
</tr>
</tbody>
</table>

Add Local Agent

Remote Agents

Remote agents run on computers other than the Bamboo server.

Select All, None, Idle, Disabled Action: Delete | Disable | Enable

<table>
<thead>
<tr>
<th>Agent</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Agent on i-01bcc86d</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-03bcc86f</td>
<td>Idle</td>
<td>View</td>
</tr>
<tr>
<td>Elastic Agent on i-1fbc873</td>
<td>Building - BTS-FUNC-JOB1-7815</td>
<td>View</td>
</tr>
</tbody>
</table>

There are currently 3 remote agents online (3 elastic). Start elastic agents here. A maximum of 25 agents are supported by your license.

Install Remote Agent | Disable Remote Agent Support

Notes

Related Topics
Managing your Elastic Agents

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 startup time' — the last time that the elastic agent was started. This is based on the Bamboo server time.
   - 'Last shutdown 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 Job builds run by the elastic agent and information about the Job build, such as the status, duration, test results, etc. You can access the full results by clicking the build number.

![Screenshot above: Elastic Agent History](image)

Related Topics

Managing your Elastic Agents

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 elastic instance of an 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 Job 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.

Please note, 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.

On this page:
- Configuring the Capabilities on an Elastic Image
- Notes

Configuring the Capabilities on an Elastic Image

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 be displayed.
3. Click the name or 'View' link of the elastic image whose capabilities you want to configure. The configuration screen will be displayed, 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 Executable
   - 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 Jobs 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.
## Elastic Image Capabilities

A capability is a feature of an agent. There are 3 types of capabilities: builders, JDKs and custom. You can use this page to view, add and delete capabilities associated with this Elastic Image Configuration. Any existing elastic instances will need to be restarted to pick up these changes.

The following capabilities exist on Elastic Agents running on an instance of this image:

### Custom

'Custom' capabilities are key-value pairs that define particular characteristics of an agent (e.g., 'operating system=Windows XP', 'fast builds=true'). For an agent to be able to build a job, both the 'Key' and 'Value' must match the job'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>Builder 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 2 (Maven 2.x)</td>
<td>/opt/maven-2.0</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x)</td>
<td>/opt/maven-2.1</td>
<td>View</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x)</td>
<td>/opt/maven-2.2</td>
<td>View</td>
</tr>
</tbody>
</table>

### JDK

'JDK' capabilities define the JDKs which are available to your build plans.

<table>
<thead>
<tr>
<th>JDK 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-5</td>
<td>View</td>
</tr>
</tbody>
</table>

### Mercurial

The path to the Mercurial executable (e.g., 'C:\Program Files (x86)\Mercurial\hg.exe' or '/usr/local/bin/hg')

<table>
<thead>
<tr>
<th>Executable</th>
<th>Path</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercurial</td>
<td>/usr/bin/hg</td>
<td>View</td>
</tr>
</tbody>
</table>

## Add Capability

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Screenshot above: Configuring elastic agent capabilities

Notes

Related Topics
Managing your Elastic Agents

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 elastic instance of an 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 Job build it is running and prevent it from running any further Job 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.

On this page:
- Disabling an Elastic Agent
- Notes

Disabling an Elastic Agent

To disable an elastic agent:

1. Navigate to the desired elastic agent, as described in Viewing your Elastic Agents.

<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</td>
<td>i-2776e24e</td>
<td>✅ Running</td>
<td>5 minutes</td>
<td>View</td>
</tr>
</tbody>
</table>

2. 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>Status</th>
<th>Up Time</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>✅ Running</td>
<td>6 minutes</td>
<td>View</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elastic Agent on</th>
<th>Status</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-2776e24e</td>
<td>✅ Online (Disabled)</td>
<td>Enable</td>
</tr>
</tbody>
</table>

Notes

Related Topics

Managing your Elastic Agents

Running Job 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 Job builds can I run on Elastic Bamboo?
- How do I run a Plan build and its Jobs on an elastic agent?
- How do I automatically start or shut down elastic instances for Job builds?
- How do I know whether my Job 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 Job builds?

What Job builds can I run on Elastic Bamboo?
You can run any of your Job builds on any elastic agent (which in turn runs on an elastic instance), provided that the elastic agent's capabilities meet the Job's requirements. An elastic agent inherits the capabilities of the elastic image it was created from. Hence, you can see which of your Jobs can run on elastic agents by checking that your Job's requirements match your elastic image's capabilities.

✔️ You view your elastic image and the Job builds that meet its requirements on the Agents and Plans matrix.

How do I run a Plan build and its Jobs on an elastic agent?

An elastic agent operates in a similar fashion to a non-elastic agent. The Bamboo server will determine if any Job builds in the queue can be built on any of the available agents (including elastic agents), based on whether or not the capabilities of these agents meet the requirements of these Jobs.

If an available elastic agent (like any other available agent) has capabilities which meet the requirements of a build in the build queue, the Bamboo server will assign that Job build to this elastic agent.

If you do not have any free elastic agents running, you can configure Bamboo to automatically start up elastic instances whose elastic agents are capable of running Job builds in the queue, or you can start up an appropriate elastic instance manually. (When an elastic instance is started, its elastic agent is also started, automatically.) For more information about starting elastic instances manually, refer to Starting an Elastic Instance.

If you do not use Bamboo's Automatic Elastic Instance Management feature and prefer to manage your elastic instances manually, then we strongly recommend that you shut down any elastic instances (running your elastic agents), when they are not in use. Minimising unutilised elastic instance uptime will help reduce costs. Read Shutting Down an Elastic Instance for instructions on how to shut down an elastic instance.

How do I automatically start or shut down elastic instances for Job builds?

Bamboo can automatically start elastic instances based on demand from the build queue and shut them down once the elastic agents running on them have been idle for a specified period of time. For more information, please refer to the Automatic Elastic Instance Management section of the Configuring Elastic Bamboo topic.

While Bamboo's Automatic Elastic Instance Management feature is the easiest and most effective method of managing elastic instances in Bamboo, you can also manage elastic instances 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 Job builds.

How do I know whether my Job build was run on an elastic agent?

The name of the image and elastic agent that ran a Job 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 certain Jobs in your Plan(s). For example, if you want to force certain Job 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 these Jobs.

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 Job 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 Job build by configuring Elastic Bamboo to use Automatic Elastic Instance Management and Amazon's Elastic Block Store (EBS).

What is EBS and how does it affect my Job 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.
Building in the cloud presents you with new opportunities to optimise your builds in Bamboo. There are also a pitfalls that you can avoid by following the appropriate build strategies.

We have identified a number of common Elastic Bamboo usage patterns and anti-patterns on this page. We recommend that you take these into consideration when adapting your build strategies for Elastic Bamboo.

On this page:

- Scenario: Per-Module Builds
  - Problems to Consider
  - Recommended Solutions
  - Key Concepts
- Scenario: Pre-populated Image
  - Problems to Consider
  - Recommended Solutions
  - Key Concepts

Scenario: Per-Module Builds

Problems to Consider

- Elastic agents are disposable. Unless they are permanently occupied, it is likely that they will be destroyed and replaced regularly, requiring a full check-out.
- Elastic agents are remote, so large check-outs can take a long time.

Recommended Solutions

- Avoid large check-outs.
- Build and unit test each module as a separate build.
- Where possible, run these builds in parallel, to save time.
- Have dependant builds download pre-built dependency artifacts from the Bamboo server, a Maven repository, or similar.

Key Concepts

- Storing a base checkout inside a custom image, which is occasionally rebuilt.
- Storing a base checkout in S3, for much faster access.
- A better mechanism for supplying dependency artifacts to a build.
- Better support for graphs of related builds (e.g. parallelism, multiple dependency builds triggering a single dependent build, a unified changeset-centric view of related builds, etc.).
- Checkout replication between elastic agents.

Scenario: Pre-populated Image

Problems to Consider

- Elastic agents are disposable.
- Some builds have many dependencies outside version control. (e.g. the contents of Maven repositories)
- Each time an agent is launched, these dependencies have to be downloaded.
- This makes builds slower and more costly.

Recommended Solutions

- Create a custom image, and pre-populate it with dependencies that seldom change.

Key Concepts

- Pre-populate the default image with the core Maven plug-ins and their dependencies.

Disabling Elastic Bamboo

If you do not want to execute Plan builds and their Jobs 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.
Disabling Elastic Bamboo

Before you begin:

- 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

You can configure your Amazon Web Services (AWS) account details on this page, as well as global settings for Elastic Bamboo. Please read the documentation on configuring Elastic Bamboo for more information.

Configuration Details

**AWS Access Key ID**

- GLFOUJ44ADSREYQ3Z1IO

- □ Change AWS Secret Access Key?

Elastic Bamboo Global Settings

- **Maximum Number of Elastic Instances**

  - 25

  - Set the maximum number of elastic instances that can be running concurrently. This must be less than or equal to the number of remote agents allowed by your license.

  - ✔ Automatically shut down elastic instance when elastic agent process ends.

- **Shutdown Delay**

  - 60

  - Time to wait before the instance shuts down automatically.

Elastic Bamboo AWS Settings

- ✔ Upload AWS account identifiers to new elastic instances (mandatory if EBS Snapshot ID specified for Elastic Image Configuration)

  - /Users/atlassian/keys/elastic_bamboo_dev/pk-FOSPE2FK7PX6DRNCHXZRC32ZG7

  - This is the location on your Bamboo server of the Account Private Key File, generated by Amazon to access your AWS account.

  - /Users/atlassian/keys/elastic_bamboo_dev/cert-FOSPE2FK7PX6DRNCHXZRC32ZG7

  - This is the location on your Bamboo server of the Account Certificate File, provided by Amazon to access your AWS account.

Automatic Elastic Instance Management

- **Elastic Instance Management**

  - Custom

  - This allows you to specify your own automatic elastic instance management configuration below.

- **Idle Agent Shutdown Delay**

  - 30

  - The number of minutes an elastic agent must be idle before Bamboo shuts down its instance (e.g. Enter 10 for 10 minutes).

- **Maximum Number of Instances to Start at Once**

  - 10

  - The maximum number of elastic instances that can be started in one go.

- **Number of Builds in Queue Threshold**

  - 5

  - The total number of builds in a queue. When this and all other thresholds have been reached, new elastic instances will be started.

- **Number of Elastic Builds in Queue Threshold**

  - 3

  - The number of builds in a queue that are executable on elastic instances. When this and all other thresholds have been reached, new elastic instances will be started.

- **Average Queue Time Threshold**

  - 1

  - The average number of minutes that builds have been waiting in a queue. When this and all other thresholds have been reached, new elastic instances will be started.

3. Click the 'Disable' button. Elastic Bamboo will disabled and a confirmation message will display.
Configuring Email and Instant Messaging Notifications

The following pages contain information on configuring email and instant messaging notifications in Bamboo.

- Configuring Notifications for a Plan and its Jobs
- Modifying Notification Templates
- Configuring Bamboo to send SMTP Email
- Configuring Bamboo to use Instant Messaging

Configuring Notifications for a Plan and its Jobs

Notifications in Bamboo are triggered by a range of events involving a Plan and its Jobs, 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.

For each Plan or Job, you can specify different recipients for each type of event notification. Also be aware that these recipients do not require Bamboo user accounts.

Adding Notifications for a Plan or Job

Before you begin:
- You must have the 'Edit' permission for a Plan, to add or remove notifications for it.
- You need to configure Bamboo's SMTP email and/or instant messaging capabilities before Bamboo can send notifications. If you have not configured either or both of these, a note will display on the page prompting you to set up the appropriate server(s):
  - To configure an email server for Bamboo, click the 'Add an Email Server' link in the note and enter the email server details in the window that displays. See Configuring Bamboo to send SMTP Email for more information.
  - To configure an instant messaging server for Bamboo, click the 'Add an Instant Messaging Server' link in the note and enter the instant messaging server details in the window that displays. See Configuring Bamboo to use Instant Messaging for more information.

To add notifications for a Plan and/or its Jobs:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the 'Notifications' tab to display the Plan's current Notifications settings.
3. 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 type of Event you want to be notified about. Refer to the list of events (above) for details.
   - Specify the Recipient:
     - 'User' — Type the username of the appropriate Bamboo user; or click the following icon to select from a list of users:
     - 'Group' — Type the name of the appropriate Bamboo group(s).
     - '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.
     - 'IM Address' — This is useful if you need to send Instant Messenger (IM) notifications to a person who is not a Bamboo user. Type the appropriate IM address.
   - If you specify a broadcast address (eg. 'project-x@broadcast.chat.mycompany.com'), Bamboo will not know the context of related IM responses.
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.

- **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.

4. Click the 'Add' button.
5. Repeat steps 5 and 6 until you have added all the build notifications that you wish to enable for this plan.
6. Click the "Save" button to save your changes.
<table>
<thead>
<tr>
<th>Notification Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All Jobs Completed</strong></td>
<td>Bamboo will send a notification whenever a Job build of the Plan finishes, regardless of the Job build's result. This notification is recommended if the latest build activity of all Jobs in this Plan are critical for people to be informed about. This is a good Job-based notification to use if you are new to Bamboo. You can change it to a less obtrusive notification option as you become more confident with continuous integration and Bamboo's build processes.</td>
</tr>
</tbody>
</table>
| **Failed Jobs And First Successful** | Bamboo will send a notification whenever:  
- a build of this Job fails.  
- the Job is “fixed” (that is, the Job's latest build is successful and the previous Job build failed). |
| **Change of Job Status**             | Bamboo will send a notification only when there has been a change in build activity status of the Jobs within this Plan over consecutive Plan builds — for example, only whenever the latest build of any Job in this Plan changes from successful to failed or vice versa (i.e. “fixed”).  
This notification option is less obtrusive than the other Job notifications mentioned above. |
| **First Failed Job For Plan**        | If multiple Jobs fail in a Plan, Bamboo will only send a notification for the first failing Job detected by the Bamboo system.  
This is a less obtrusive notification option that informs about a failing Job (and hence, Plan) in the shortest possible time. |
| **Job Error**                        | Bamboo will send a notification whenever an error occurs in one of the Plan's Job build processes (i.e. the activities that Bamboo performs to run a Job build). This event is not related to failures of the actual build itself (see the ‘Failed Jobs And First Successful’ and ‘Failed Builds And First Successful’ events above). For example, a notification will be sent if Bamboo encounters an error when connecting to the repository, or detecting changes. |
| **Job Hung**                         | Bamboo will send a notification whenever it determines that one of the Plan's Job builds has hung, according to the hung Job build criteria (read more about configuring your hung Job build settings).  
Use this notification to ensure that the relevant people are informed when a Job build becomes unresponsive. |
| **Job Queue Timeout**                | Bamboo will send a notification whenever one of the Plan's Job builds has been waiting in the queue for longer than the build queue timeout criteria (read more about configuring your Job's Build Queue Timeout settings).  
Use this notification to ensure that the relevant people are informed when a Job build is stuck in the build queue for too long. |
| **Job Queued Without Capable Agents**| Bamboo will send a notification whenever one of the Plan's Job builds is queued and there are no agents capable of building it.  
Use this notification to ensure that people are notified when changes to agents adversely affect your Job's builds. |

Removing Notifications from a Plan or Job

Before you begin:

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

To remove notifications for a Plan and/or its Jobs:

1. Navigate to the configuration for the desired Plan, as described on Editing a Plan.
2. Click the ‘Notifications’ tab to display the Plan's current Notifications settings.
3. Locate the notifications that you wish to remove and click the corresponding ‘Remove’ link in the ‘Operations’ column for each of these notifications.

Notes

Related Topics

- Editing a Plan
- Configuring Projects, Plans, Stages and Jobs
- Configuring the Job Build Queue Timeout Event
- Changing your Notification Preferences
- Granting Plan Permissions to Users or Groups
- Configuring Bamboo to send SMTP Email
- Configuring Bamboo to use Instant Messaging

Modifying Notification Templates

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.

Some content in notifications can also be configured via system properties, such as, the number of log lines to include in email notifications that display log information.
Modifying a Notification Template

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`
3. The filename will have formatted as: `<event name><notification type>.ftl`, e.g. `AfterXFailedHTMLMail.ftl`
4. Modify the copied template, as desired. Please see the section on Working with Freemarker below for tips on updating templates.
5. 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.

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.

Configuring Notifications Content via System Properties

The following system properties can be configured to control some of the content that is included in notifications (e.g. the number of log lines to include in email notifications that display log information). For instructions on how to configure a system property, please refer to the [Configuring System Properties](#) page.

Before you begin:

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>bamboo.notifications.logLinesToInclude</td>
<td>Specifies the number of log lines to include in email notifications that display log information.</td>
<td>100</td>
</tr>
</tbody>
</table>

Notes

- **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.

Related Topics

- Configuring Email and Instant Messaging Notifications

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.
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.

```
${commit.comment?html} // for data to be encoded to be displayed as html
${commit.author?url} // for data to be encoded for a url
```

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
Provide Test Error Details

Configure Bamboo to Send SMTP Email

Bamboo can send email notifications about its build results. There are two steps to setting this up:

1. Configure Bamboo to send SMTP email (see instruction below).
2. Configure a plan to send SMTP email notifications about build results (see Configuring Notifications for a Plan and its Jobs).

Notes

Configuring Email and Instant Messaging Notifications

Configuring Bamboo to send SMTP Email
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 'Subject Prefix' 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. 'java:comp/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.

![Screenshot above: Email Server Details]

Configuring Email Notifications for Gmail
Gmail.com uses TLS (SSL). A JNDI connector needs to be configured. Unfortunately Bamboo Standalone does not yet support JNDI with TLS.

To enable gmail as your mail server:

1. Install Bamboo as war on Tomcat
2. Add the following configuration yo your apache-tomcat-xxx/conf/Catalina/localhost/bamboo.xml

```xml
<Context path="/bamboo" docBase="${catalina.base}/bamboo.war" reloadable="true">
  <Resource name="mail/GmailSmtpServer" auth="Container" type="javax.mail.Session">
    <mail.smtp.host="smtp.gmail.com"/>
    <mail.smtp.port="465"/>
    <mail.smtp.auth="true"/>
    <mail.smtp.user="<your-name>@gmail.com"/>
    <mail.smtp.password="<your-pw>"/>
    <mail.smtp.starttls.enable="true"/>
    <mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"/>
  </Resource>
</Context>
```

3. Ensure that files mail-X.X.jar and activation-X.X.jar will exist in apache-tomcat-xxx/lib folder only. You can move the ones shipped at <Bamboo-Install>/WEB-INF/lib to apache-tomcat-xxx/lib if they don't exist there yet. If they are already there, you can delete the ones shipped with Bamboo.
4. Configure mail JNDI as in Bamboo.

Notes

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.

Related Topics

Now that you have configured Bamboo's SMTP email capability, you can configure notifications for a Plan and its Jobs.

Configuring Bamboo to use Instant Messaging

Bamboo can send Instant Messaging (IM) notifications about its build results. There are two steps to setting this up:

1. Configure Bamboo to use Instant Messaging (IM)
2. Configure a Plan to send IM notifications about its build results (see Configuring Notifications for a Plan and its Jobs).

Please note, Bamboo supports XMPP protocol for messaging. This means Bamboo can be used with Gtalk or your enterprise XMPP server.

On this page:
- Configuring Bamboo to use Instant Messaging (IM)
- Notes

Configuring Bamboo to use Instant Messaging (IM)

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 a TLS/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.
Now that you have configured Bamboo's IM capability, you can specify notifications for a Plan and its Jobs.

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:

- 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.
- 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
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'.
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.
7. In the 'Password' field, type the password for the account specified in step 6.
8. Type a test IM user's address (e.g. atlassianbamboo@gmail.com) in the 'Test Recipient Address' box. (Note: use a different user to the one you specified in step 6.)
9. Click the 'Test' button, and verify that the message is successfully received.
10. Click the 'Save' button.

Notes

Related Topics

Configuring Email and Instant Messaging Notifications

Managing Users and Permissions

The information on this page describes how users and permissions are configured in Bamboo. Please note, this documentation does not relate to application-level security for Bamboo. If you are looking for information on security of the Bamboo application, please refer to the Managing Bamboo Security page.

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 Plan and its Jobs. 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.

Related Topics

For more information on configuring users, groups and security, see the following topics:

- 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

Global Security and Permission Properties

- Enabling or Disabling Public Signup
- Enabling or Disabling Contact Details Display
- Enabling or Disabling Captcha for Failed Logins

Working with External User Repositories

- Integrating Bamboo with Crowd
- Integrating Bamboo with LDAP
  - Configuring the Caching of your LDAP Repository
  - Testing LDAP or Active Directory connectivity with Paddle

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.

Related Topics

For more information on configuring users, see the following topics:

- 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 anyone who can log in 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.

On this page:

- Creating a Bamboo User
- Notes

Creating a Bamboo User

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.
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.

If you don’t know the user’s login name for their source-code repository, they can specify it themselves later.
Changing a User's Password or Details

The instructions on this page describe how to change a user's password or details (e.g. email or full name) in Bamboo.

On this page:
- Changing a User's Password
- Changing a User's Details
- Notes

Changing a User's Password

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.

Changing a User's Details

To change a user's details:

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.

Notes

- 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.

Related Topics
Managing Users

Deleting or Deactivating a User

The instructions on this page describe how to delete a user or how to deactivate a user.

Deleting a user removes their Bamboo user account. Deactivating a user revokes their permission to log in to Bamboo.

On this page:

- Deleting a Bamboo User
- Deactivating a Bamboo User
- Notes

Deleting a Bamboo User

Before you begin:

- 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.
- 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.
Deactivating a Bamboo User

To deactivate a Bamboo user, please see Deactivating a Bamboo user.

Notes

Related Topics

Managing Users
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.

On this page:
- Granting Global Administration Rights to a User
- Granting Plan Administration Rights to a User
- Notes

Granting Global Administration Rights to a User

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).

Granting 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).

Notes
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

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, 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.
**Notes**

**Related Topics**

**Managing Groups**

**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.

---

**Deleting a Group**

Before you begin:

- 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.
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.

On this page:
- Adding Users to a Group
- Removing Users from a Group
- Notes

Adding Users to a Group

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.
Removing Users from a Group

Before you begin:

- You cannot remove a user from the **bamboo-admin** group if they are the only member.

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.

Notes

Related Topics

Managing Groups

Managing Permissions for Users and Groups

About permissions

A **Plan permission** is the ability to perform a particular operation in relation to a Plan and its Jobs. 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 Global Permissions to Users or Groups
- Allowing Anonymous Users to access Bamboo

Granting Plan Permissions to Users or Groups

A **Plan permission** is the ability to perform a particular operation in relation to a Plan and its Jobs. For each Plan, different permissions can be granted to particular groups and/or users.
Please note, anyone with the 'Admin' global permission automatically has all plan permissions for every plan. 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  
  **People who don't have the 'View' permission will not know that the plan exists.** | - 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 Configuring 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.

**On this page:**

- Granting Multiple Plan Permissions to a User
- Granting Multiple Plan Permissions to a Group
- Granting Multiple Plan Permissions to All Bamboo Users
- Granting Multiple Plan Permissions to Anonymous Users
- Revoking Multiple Plan Permissions

**Granting Multiple Plan Permissions to a User**

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.

**Granting Multiple Plan Permissions to a Group**

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.

**Granting Multiple Plan Permissions to All Bamboo Users**

- 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.

**Granting Multiple Plan Permissions to Anonymous Users**

Anonymous users are people who are not logged in to Bamboo.

- 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.

**Revoking Multiple Plan Permissions**

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. If you deselect all permissions for a user or group, that user or group will disappear from the screen.
  7. Click the 'Save' button.
### Bulk Edit Plan Permissions Wizard

![Warning]

*All existing permissions defined for the plans you select will be deleted and replaced with the new permissions you specify.*

**Select Plans**

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.

Select: All, None

**Artifact Sharing Dogfooding**

- [ ] Artifact sharing
- [ ] CI Tests
- [ ] Maven Sharing CI Tests
- [ ] Maven Sharing CI Tests with Maven 3

**Bamboo Main**

- [ ] CI Tests
- [ ] Extras
- [ ] Stable CI Tests
- [ ] Stable Extras

Select: All, None

[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>
<tbody>
<tr>
<td>‘Access’</td>
<td>Permission to view the Bamboo system. <img src="https://example.com" alt="Warning" /> The ability to view build plans and build results is subject to individual plan permissions.</td>
<td>- a particular user&lt;br&gt;- a particular group&lt;br&gt;- all logged-in users&lt;br&gt;- anonymous users¹</td>
</tr>
<tr>
<td>‘Create Plan’</td>
<td>Permission to create new build plans.</td>
<td>- a particular user&lt;br&gt;- a particular group&lt;br&gt;- all logged-in users</td>
</tr>
</tbody>
</table>
Admin  
Permission to:  
- access the Bamboo 'Administration' menu  
- create plans  
- delete plans  

The 'Admin' global permission also includes all plan permissions, for every plan.

The 'Restricted Admin' global permission used in JIRA Studio is equivalent to this permission but has the following restrictions:

- Local agents cannot be created because JIRA Studio only uses Elastic Agents
- Users and groups are managed by JIRA Studio

† i.e. people who are not logged in to Bamboo.

The processes for granting and revoking global permissions are described below.

On this page:
- Granting Global Permissions to a User
- Granting Global Permissions to a Group
- Granting Global Permissions to All Bamboo Users
- Granting Global Permissions to Anonymous Users
- Revoking Global Permissions
- Notes

Granting Global Permissions to a User

To grant global permissions to a user:

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 'User'.
5. 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').
6. Click the 'Add' button.
7. 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.
8. Click the 'Save' button.

Granting Global Permissions to a Group

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.
   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').
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.

Granting Global Permissions to All Bamboo Users

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.

Granting Global Permissions to Anonymous Users

Before you begin:
The ‘Access’ global permission is the only global permission that can be granted to anonymous users.

To grant global permissions 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.

**Revoking Global Permissions**

Before you begin:

- If you deselect all permissions for a user or group, that user or group will disappear from the ‘Permissions’ tab for this plan.

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.
6. Click the ‘Save’ button.

![Global Permissions](image)

### Notes

#### Related Topics

Managing Permissions for Users and Groups
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'.

Global Security and Permission Properties

The Global Security and Permission Properties page allows a Bamboo system administrator to configure security- and permission-related properties, which apply to Bamboo at a site-wide level.

Further Information

Read more about configuring Bamboo's global security and permission properties:

- Enabling or Disabling Public Signup
- Enabling or Disabling Contact Details Display
- Enabling or Disabling Captcha for Failed Logins

Enabling or Disabling Public Signup

If you enable signup for your Bamboo system, visitors can create their own Bamboo user accounts. Public signup is enabled on your Bamboo site if you see the 'Signup' link at the top-right of the Bamboo user interface.
Enabling Signup

To enable signup:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Select the 'Enable Signup?' check box. An additional option 'Enable Captcha On Signup' becomes available.
5. If you wish to enable Captcha on signup as an additional security measure to prevent brute force attacks, ensure that the 'Enable Captcha On Signup' option is selected.
6. Click the 'Save' button.
7. Log out of Bamboo and verify that the top navigation bar now contains a 'Signup' link (see screenshot below).

Disabling Signup

To disable signup:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Clear the 'Enable Signup?' check box.
5. Click the 'Save' button.

Screenshot above: Security settings for Bamboo, including signup

Notes

- For more information about the purpose of Captcha and its ability to prevent brute force attacks, please refer to the introduction of the Enabling or Disabling Captcha for Failed Logins topic.

Related Topics

- Global Security and Permission Properties

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.
Enabling Contact Details Display

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 to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Select the 'Enable contact details to be displayed?' check box.
5. Click the 'Save' button.

Disabling Contact Details Display

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 to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Clear the 'Enable contact details to be displayed?' check box.
5. Click the 'Save' button.

Notes

Related Topics

Global Security and Permission Properties

Enabling or Disabling Captcha for Failed Logins

Captcha is a tool that prevents brute force attacks on the Bamboo login screen. A brute force attack occurs when an attacker uses malicious code to make automated, repeated login attempts on a Bamboo site with the aim of gaining access to that Bamboo site.

If you are a Bamboo system administrator, you can configure Bamboo to block these automated login attempts. Once a certain number of failed login attempts has been reached (the default is three) Bamboo's Captcha feature will be activated. When Captcha is activated, users will need to recognize a distorted picture of a word and must type the word into a text field. This is easy for humans to do, but very difficult for computers.

On this page:

- Enabling Captcha
- Disabling Captcha

Enabling Captcha

To enable Captcha for Bamboo:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Ensure that the 'Enable Captcha' check box has been selected.
5. If you wish to change the number of failed login attempts permitted by Bamboo before Captcha is activated, specify the number of failed attempts in the 'Login Attempts' field. (This field is mandatory and requires a value of 1 or more.)
6. Click the 'Save' button.

Disabling Captcha

To disable Captcha:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'Security Settings' link in the left navigation column to open the 'Global Security and Permission Properties' page.
3. Click the 'Edit' button on this page.
4. Clear the 'Enable Captcha' check box.
5. Click the 'Save' button.
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
  - Testing LDAP or Active Directory connectivity with Paddle

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>
<tr>
<td>Bamboo integrated with — Crowd with authentication-only delegated to 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
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:

```plaintext
Dn: CN=Sales and Marketing,CN=Users,DC=ad,DC=atlassian,DC=com
objectClass: top; 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 first repository definition in your `atlassian-user.xml` file will take precedence.

- **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.

```plaintext
<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):
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>
   </ldap>

Filters: `<userSearchFilter>` and `<groupSearchFilter>` may use the AD specific filter syntax. Operators such as `&` need to be escaped.

   ```xml
   <userSearchFilter>(objectClass=inetorgperson)(memberOf=cn=bamboo-usr,ou=groups,DC=atlassian,dc=com)</userSearchFilter>
   ```

Step 4 — Optional LDAP Settings

The following settings do not appear in the default atlassian-user.xml file. Their default values are as follows:

```xml
<poolingOn>true</poolingOn>
<maxSize>0</maxSize>
<preSize>10</preSize>
<debugLevel>none</debugLevel>
<securityProtocol>plain ssl</securityProtocol>
<authentication>simple</authentication>
<timeout>0</timeout>
<InitialContextFactory>com.sun.jndi.ldap.LdapCtxFactory</InitialContextFactory>
<batchSize>100</batchSize>
<timeToLive>0</timeToLive>
<userSearchAllDepths>true</userSearchAllDepths>
<groupSearchAllDepths>true</groupSearchAllDepths>
```

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 `</ldap>` line shown in 'Stage 3' above:

```xml
<initSize>20</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).

**Notes**

- 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.

**Related Topics**

Working with External User Repositories

**Configuring the Caching of your LDAP Repository**

The instructions on this page describe how to configure the caching of your LDAP repository.

On this page:

- Disabling the Caching of Users
- Enabling the Caching of Users
- Configuring the LDAP Caches
- Notes

### 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** 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"/>
```

- **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"/>
```

- **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"/>
```
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** 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.LDAPUserManagerReadOnly.myLdapRepository.repository"
      maxElementsInMemory="500"
      eternal="false"
      timeToIdleSeconds="300"
      timeToLiveSeconds="300"/>
```

- **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"
      maxElementsInMemory="500"
      eternal="false"
      timeToIdleSeconds="300"
      timeToLiveSeconds="300"/>
```

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.groups_hasMembership"
      maxElementsInMemory="500"
      eternal="false"
      timeToIdleSeconds="300"
      timeToLiveSeconds="300"/>
```

```xml
<cache name="com.atlassian.user.impl.ldap.LDAPGroupManagerReadOnly.myLdapRepository.groups_getGroupsForUser"
      maxElementsInMemory="500"
      eternal="false"
      timeToIdleSeconds="300"
      timeToLiveSeconds="300"/>
```

- **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"/>
```

Notes

Related Topics

Integrating Bamboo with LDAP

Testing LDAP or Active Directory connectivity with Paddle

Paddle is a tool that will test the LDAP or Active Directory settings in your atlassian-user.xml.
Using Paddle

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>java -jar paddle-x.x.jar debug</td>
<td>Prints DEBUG messages to the console as well as paddle.log.</td>
</tr>
<tr>
<td>limit</td>
<td>java -jar paddle-x.x.jar 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:

```
# Support Tool version 1.1
# 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

User: CN=Justin Koke
Member of:
(1) CN=Domain Controllers
(2) CN=Enterprise Admins

Found more than 10 results.

-----------------------------------------------------------------

TEST 2: Search and list 10 groups

-----------------------------------------------------------------

Group: CN=HelpServicesGroup
Members:
(1) CN=SUPPORT_388945a0,CN=Users,DC=ad,DC=atlassian,DC=com

Group: CN=TelnetClients
No members in this group.

Group: CN=IIS_WP
Members:
(1) CN=S-1-5-20,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
(2) CN=S-1-5-6,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
(3) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com
(4) CN=IWA_MALTSHOVEL,CN=Users,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005SQLBrowserUser$MALTSHOVEL
Members:
(1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSSQLServerADHelperUser$MALTSHOVEL
Members:
(1) CN=S-1-5-20,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005SQLAgentUser$MALTSHOVEL$MSSQLSERVER
Members:
(1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005SQLUser$MALTSHOVEL$MSSQLSERVER
Members:
(1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSFTEUser$MALTSHOVEL$MSOLAP\MSSQLSERVER
Members:
(1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005MSOLAPUser$MALTSHOVEL$MSSQLSERVER
Members:
(1) CN=S-1-5-18,CN=ForeignSecurityPrincipals,DC=ad,DC=atlassian,DC=com

Group: CN=SQLServer2005NotificationServicesUser$MALTSHOVEL
Related Topics

Integrating Bamboo with LDAP

Managing Data and Backups

Related Topics

For information on managing data and backups, see the following topics:

- 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
- Configuring Global Expiry of Job Build Results
- Specifying a Backup Schedule
- Exporting Data for Backup
- Importing Data from Backup

Locating Important Directories and Files

The information on this page describes how to find important Bamboo directories and files.

On this page:

- Bamboo Server
- Bamboo Agent

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

- `artifacts/PLAN_KEY/shared/build-BUILD_NUMBER/` — This is a folder shared by all the stages of a certain plan. Stages will place Artifacts here so that other stages from the same plan can have access to them. The `BUILD_NUMBER` will always be composed with 5 digits, having the number completed with zeros. For instance, for build “42” the number will be “00042”.
- `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/JOB_KEY` — 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, and 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/` — Known as the Build Directory. This is where Bamboo stores build results (note that they will be deleted as described in Configuring Global Expiry of Job 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/JOB_KEY/results` — Contains the build results for all the builds belonging to the ‘JOB_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/JOB_KEY/download-data` — Contains the logs for each build belonging to the ‘JOB_KEY’ plan.
  - `xml-data/configuration/` — 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
**Bamboo 3.1 Documentation**

### 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.
- **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, unless you have used the Standalone Installer for Windows. (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.)

\[ If you have used the Standalone Installer for Windows, log files will be located at %USERPROFILE%\bamboo.log. For Bamboo running as a Windows service it can be found at %WINDIR%\System32\Config\systemprofile\bamboo.log. \]

- **webapp/** — This directory contains all the Bamboo server application files.
- **webapp/WEB-INF/lib/** — 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 xml-data/build-dir/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">/home/Bamboo-home/xml-data/build-dir</property>
   ......
   ```

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:
Using Bamboo’s embedded HSQL database

Once Bamboo is running, you can view the database configuration details as follows.

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘Database Configuration’ link in the left navigation column, under the ‘System’ section.

### Viewing your Database Connection Details

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, under the ‘System’ section.

### Moving Bamboo Data to a Different Database

When you installed Bamboo, you would have set up a database connection by following one of these processes:

- 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.

### Related Topics

Managing Data and Backups

## Moving Bamboo Data to a Different Database

To move your Bamboo data to a different database:

1. Back up 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:

   - PostgreSQL 8.2
How do I connect Bamboo to an unsupported database?

At Step 3 of the Setup Wizard (see screenshot below), select 'Import existing data' and specify the export file created in Step 1 above.

Wait while Bamboo imports your data. (You will not need to complete any more steps of the Setup Wizard.)

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

Notes

Related Topics

Managing Data and Backups

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).

On this page:

- Re-indexing Bamboo's Build Results Data
- Optimising Bamboo's Build Results Data
- Notes

Re-indexing Bamboo's Build Results Data

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.

Optimising Bamboo's Build Results Data

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.

Notes

Related Topics

Managing Data and Backups

Configuring Global Expiry of Job Build Results

By enabling global build expiry, you can choose what Job build result data* will be kept for all Plans in your Bamboo system and for how long this data will be kept (e.g. for reporting purposes), before Bamboo automatically deletes it.

* 'Job build result data' refers to all Job builds of any given Plan's build.

If you disable global build expiry, all of your Plan's Job build result data will never be automatically deleted from your Bamboo server. This could lead to a large portion of your Bamboo server's storage space being consumed to store this data.

You can enable/disable build expiry for:
• all plans (i.e. "global" as described below). This is generally the easiest way to manage build expiry in Bamboo. Your settings will apply to all Plans that do not have individual build expiry settings.
• individual plans (see Configuring Expiry of a Plan's Job Build Results). You would generally only do this if there is a specific reason to keep/delete a particular Plan's Job build result data.

You can also delete the results of a Plan build manually — see Deleting the Results of a Plan Build.

On this page:

- Enabling and Configuring Global Expiry of Job Build Results
- Disabling Global Expiry of Job Build Results
- Notes

Enabling and Configuring Global Expiry of Job Build Results

Before you begin:

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

To enable and configure global expiry of Job build result 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 fields listed below will be displayed.

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 Configuring Expiry of a Plan's Job Build Results for details on how to override the global build expiry settings.

- 'What should be expired?': — select:
  - 'Build result' — if you want to delete all build results data (including artifacts and build logs). If you choose this option, the following two options are automatically selected.
  - 'Artifacts' — if you want to selectively delete all user-defined artifacts of Jobs in this Plan but keep all other Job build result data (such as build logs).
  - 'Build Logs' — if you want to selectively delete all build logs of Jobs in this Plan but keep all other Job build result data (such as artifacts).

- In the Expiry criteria section, use one of the follow three criteria to determine how much Job build result data to keep, by modifying the 'Expiry period' and 'Minimum builds to keep' fields accordingly:

  - To keep all Job build results of this Plan up to a certain age, and a minimum number of these Job build results, 

    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 Job build results of this Plan. E.g. specify '24 months' to keep all Job build results for the last two years.
    - In the 'Minimum builds to keep' field, specify '0'.
    - To keep a specified (or minimum) number of Job build results of this Plan,
      - In the 'Expiry period' field, specify '0'.
      - In the 'Minimum builds to keep' field, specify the number of Job build results of this Plan you want to keep. E.g. specify '50' to keep the latest 50 Job build results.
    - To keep all Job build results of this Plan up to a certain age and a minimum number of these Job build results, 
      - In the 'Expiry period' field, specify the number of months/weeks/days for which you want to keep your Job build results of this Plan. E.g. specify '24 months' to keep all Job build results for the last two years.
      - In the 'Minimum builds to keep' field, specify the number of Job build results of this Plan you want to keep. E.g. specify '50' to keep the latest 50 Job build results. (Hence, even if all Job builds of this Plan are over two years old, the last 50 Job build results will not be deleted.)

- If you wish to keep Job build results with particular labels, select the 'Don't expire builds with certain labels' check box.
  The 'Labels to keep' field will be displayed:
  - In the 'Labels to keep' field, specify any labels which match the labels applied to Job builds you want to keep. (If you want to specify more than one label, use spaces to separate them.) For any label(s) specified, all Job builds of this Plan which have a matching label will never be deleted, regardless of the Minimum builds to keep and Labels to keep settings.

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 Configuring Automatic Labelling of Job Build Results in the Bamboo Administrator's Guide.
Disabling Global Expiry of Job Build Results

To disable global expiry of Job build result 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 Configuring Expiry of a Plan’s Job Build Results for details on how to override the global build expiry settings.

---

### Build Expiry

The build expiry schedule determines when build results and/or artifacts will be deleted from your Bamboo system. On this page, you can configure when the build expiry schedule will run and what will happen to the build results globally when it runs. You can override what will happen to build results at the build plan level.

#### Build Expiry Settings

Bamboo will check for expired data based on the cron expression defined below.

```
Cron Expression: 0 0 0 * *
```

Cron is a time-based job scheduler. Refer to the documentation for help constructing cron expressions.

#### Global Configuration

Bamboo will remove expired data based on the settings below. This global configuration can be overridden for individual plans by updating the Post Actions for a plan. Refer to the documentation for help configuring the settings below.

- **Disable Global Build Expiry**
  - Check this box to prevent the automatic deletion of build results from Bamboo.

#### What should be expired?

- **Build results**
  - The entire result will be removed (including artifacts)
  - **Artifacts**
    - User defined artifacts will be expired
- **Build logs**
  - Build log will be expired

#### Expiry criteria

<table>
<thead>
<tr>
<th>Expiry period</th>
<th>1</th>
<th>months</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Expire builds that completed before the above time period. Use 0 to ignore this option.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Minimum builds to keep</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Minimum number of builds per plan to keep (e.g. 1 will keep the last build only), the rest will expire. Use 0 to ignore this option.</td>
</tr>
</tbody>
</table>

- **Don't expire builds with certain labels**
  - **release**
    - The labels (separated by spaces) that you don't want to expire
Specifying a Backup Schedule

You can configure Bamboo to automatically create a backup each night, rather than doing a manual export every time.

On this page:
- Specifying a Backup Schedule
- Disabling a Backup
- Notes

Specifying a Backup Schedule

Before you begin,

- Bamboo will be unavailable while the backup process completes. The export itself may take a long time to complete, depending on the number of builds and test. We recommending running your backups at a time of day or night when usage is low.
- Backups may require large amounts of disk space, depending on the number of builds and tests. Please make sure you have enough disk space in your desired backup location 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).
- 'Backup Path' setting — Bamboo restricts the editing of certain file path settings for security reasons (see Bamboo Security Advisory 2010-05-04). If so, you must configure Bamboo to permit modification to its file path settings, by starting Bamboo with the system property bamboo.paths.set.allowed=true. The procedure for configuring a Bamboo system property is described on Configuring System Properties.

Once you have configured your file path setting, we recommend removing or disabling the bamboo.paths.set.allowed system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.

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. The 'Scheduled Backup Details' page will be displayed, showing details about the status of scheduled backups or any currently configured backup.
3. Click the 'Edit' button to edit the current 'Scheduled Backup Details'.
4. Ensure that the 'Disable scheduled backups' check box is cleared to ensure that your automatic backups will be performed.
5. Select the 'Backup Artifacts' check box if you want to include build artifacts in your scheduled backups.
6. In the 'Backup path' field, specify the directory where you want to store your backups. Each backup will be stored as a single file.
7. In the 'Backup file prefix' field, specify the first part of the filename for all your backup files.
8. 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'.
9. In the 'Schedule' field, choose one of the following frequencies at which backups will be executed:
   - 'Daily' — creates a backup every day. If you choose this option, select the time 'Interval' with which Bamboo will execute backups:
     - 'once per day' — If you choose this option, select the time of day at which the backup will be conducted.
     - Any other 'Interval' option — select the segment of time in a day during which the backups will be conducted.
   - 'Days per Week' — creates a backup on specific days of a week. If you choose this option, select the check boxes next to the days of the week on which you want the backups conducted. Also, specify the time 'Interval' with which Bamboo will execute backups on these days. (Refer to the 'Daily' interval options above for details.)
   - 'Days per Month' — creates a backup on a specific day of a calendar month. If you choose this option, select either:
     - The calendar date of the month on which the backup will run.
     - The day of the week in the calendar month on which the backup will run.
   - 'Advanced' — creates a backup whose schedule is dependent on a Cron expression. For more information on creating Cron expressions, please refer to the documentation on creating Cron expressions in Bamboo.
10. Click the 'Save' button. Your first backup will run when your server's clock matches the specified time.

Disabling a Backup

If you disable schedule backups, your schedule details will be retained but no automatic backups will be performed.

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. The 'Scheduled Backup Details' page will be displayed, showing details about the status of scheduled backups or any currently configured backup.
3. Click the 'Edit' button to edit the current 'Scheduled Backup Details'.
4. Select the 'Disable scheduled backups' check box.
5. Click the 'Save' button.

### Scheduled Backups

Configures Bamboo to generate a backup at specific time intervals.

#### Scheduled Backup Details

```
Please choose the schedule for your backups carefully. Bamboo will be unavailable for the duration of the backup process.

```

- **Disable scheduled backups**
  - Switch scheduled backups on and off.
- **Backup Artifacts**
  - Should Bamboo backup build artifacts

**Backup path**

```
/Volumes/Pharlap/opt/docfood/panda/home/backups/
```

**Backup file prefix**

```
Bamboo_backup
```

**Backup file date pattern**

```
yyyy_MM_dd
```

**Schedule**

```
Each Saturday at 12:00 am
```

[Save] [Cancel]

### Exporting Data for Backup

The instructions on this page describe how to export Bamboo data for backup.

#### Notes

- Managing Data and Backups
- Exporting Data for Backup
- Importing Data from Backup

### Exporting Data for Backup

Before you begin:

- Bamboo will be unavailable while the backup process completes. The export itself may take a long time to complete, depending on the number of builds and test. We recommending running your backups at a time of day or night when usage is low.
- Backups may require large amounts of disk space, depending on the number of builds and tests. Please make sure you have enough disk space in your desired backup location 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).
- 'Export Directory Path' setting — Bamboo restricts the editing of certain file path settings for security reasons (see Bamboo Security Advisory 2010-05-04). If so, you must configure Bamboo to permit modification to its file path settings, by starting Bamboo with the...
system property `bamboo.paths.set.allowed=true`. The procedure for configuring a Bamboo system property is described on Configuring System Properties.

Once you have configured your file path setting, we recommend removing or disabling the `bamboo.paths.set.allowed` system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.

- 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.

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. Specify the absolute 'Export Directory Path' to which Bamboo will export its data. For example, 'C:\Documents and Settings\<me>\bamboo-home\my-backups' for a Windows-based operating system. You would typically use forward-slashes (without drive letter specification) on UNIX-based operating systems.
4. Type the 'File Name' of the file to which Bamboo will export its data. For example, 'export.zip'.
5. Select the 'Export Artifacts' check box if you want to backup your build artifacts.
6. Click the 'Export' button.

Screenshot above: Exporting data for backup

Notes

Related Topics

Managing Data and Backups
Specifying a Backup Schedule
Importing Data from Backup

Importing Data from Backup

The instructions on this page describe how to import data from a Bamboo backup.

On this page:

- Importing Data from Backup
- Notes

Importing Data from Backup

Before you begin:

- The import process will delete your Bamboo installation and restore data from a previous export of Bamboo. This includes login...
data, hence you will need to know an administration login in the Bamboo data to be imported.

- Bamboo will be unavailable until the import process is complete, which may take some time.
- 'File Path' setting — Bamboo restricts the editing of certain file path settings for security reasons (see Bamboo Security Advisory 2010-05-04). If so, you must configure Bamboo to permit modification to its file path settings, by starting Bamboo with the system property bamboo.paths.set.allowed=true. The procedure for configuring a Bamboo system property is described on Configuring System Properties.

Once you have configured your file path setting, we recommend removing or disabling the bamboo.paths.set.allowed system property and restarting Bamboo. If your Bamboo instance is accessible to anyone outside your organisation, then this will minimise the risk of Bamboo being compromised by security-related attacks.

- Bamboo cannot import backups files that are larger than 200MB on 32-bit JVMs (see BAM-1737). A 64-bit JVM with a maximum heap space of 4G (-Xmx4096m) can handle these imports.
- If you manage users externally (LDAP or Crowd) and the Bamboo internal user repository (in the backup file) contains user names that duplicate user names in the external repository, you will not be able to import from the backup file.

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/bamboo-home/export.zip" on UNIX-based operating systems.
4. Select the 'Backup data' check-box. (This is HIGHLY RECOMMENDED.)
5. Specify the absolute 'Backup Directory Path' to which Bamboo will backup its data. This must be different from the 'File Path' specified above. For example, 'C:\Documents and Settings\<me>\bamboo-home\my-backups' for a Windows-based operating system. You would typically use forward-slashes (without drive letter specification) on UNIX-based operating systems.
6. Type the 'File Name' of the file to which Bamboo will export its data. For example, 'backup.zip'.
7. Click the 'Import' button.
8. After the import is complete,
   - check the paths of your builders and JDK.
   - index your data.

---

**Notes**

**Related Topics**

- Managing Data and Backups
- Specifying a Backup Schedule
- Exporting Data for Backup
Configuring System Settings

Related Topics

For information on configuring system settings, see the following topics:

- Viewing Bamboo's System Information
- Updating your Bamboo License Details
- Specifying Bamboo's Title
- Specifying Bamboo's URL
- Logging in Bamboo
- Enabling GZIP Compression
- Enabling Bamboo's Remote API
- Configuring System Properties
- Finding Your Bamboo Support Entitlement Number (SEN)
- Configuring Gravatar Support

Viewing Bamboo's System Information

When you installed Bamboo, you provided information about how the system should be configured. You can view the system information via your administration console in Bamboo.

The system information contains useful data for you to send to Atlassian when requesting support.

On this page:

- Viewing your Bamboo System Information
- Notes

Viewing your Bamboo System Information

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.
System Information

System Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Date</td>
<td>Tuesday, 15 Feb 2011</td>
</tr>
<tr>
<td>System Time</td>
<td>15:48:22</td>
</tr>
<tr>
<td>Up Time</td>
<td>1 day, 31 minutes, 14 seconds (since Mon Feb 14 15:18:08 EST 2011)</td>
</tr>
<tr>
<td>Username</td>
<td>panda</td>
</tr>
<tr>
<td>User Timezone</td>
<td>Australia/Sydney</td>
</tr>
<tr>
<td>User Locale</td>
<td>English (United States)</td>
</tr>
<tr>
<td>System Encoding</td>
<td>MacRoman</td>
</tr>
<tr>
<td>Operating System</td>
<td>Mac OS X 10.8.5</td>
</tr>
<tr>
<td>Operating System</td>
<td>x86_64</td>
</tr>
<tr>
<td>Available Processors</td>
<td>8</td>
</tr>
<tr>
<td>Application Server</td>
<td>Apache Tomcat 6.0.18</td>
</tr>
</tbody>
</table>

Java / JVM Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Version</td>
<td>1.6.0_17</td>
</tr>
<tr>
<td>Java Vendor</td>
<td>Apple Inc.</td>
</tr>
<tr>
<td>JVM Spec. Version</td>
<td>1.0</td>
</tr>
<tr>
<td>JVM Spec. Vendor</td>
<td>Sun Microsystems Inc</td>
</tr>
<tr>
<td>JVM Version</td>
<td>14.3-b01-101</td>
</tr>
<tr>
<td>JVM Vendor</td>
<td>Apple Inc.</td>
</tr>
<tr>
<td>JVM Name</td>
<td>Java HotSpot(TM) 64-Bit Server VM</td>
</tr>
<tr>
<td>JRE Version</td>
<td>1.6.0_17-b04-243</td>
</tr>
<tr>
<td>JRE Name</td>
<td>Java(TM) SE Runtime Environment</td>
</tr>
</tbody>
</table>

Network

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>panda <a href="mailto:sydney@atlassian.com">sydney@atlassian.com</a></td>
</tr>
<tr>
<td>IP Address</td>
<td>172.20.6.108</td>
</tr>
</tbody>
</table>

Memory Statistics

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Memory</td>
<td>292 MB</td>
</tr>
<tr>
<td>Free Memory</td>
<td>63 MB</td>
</tr>
<tr>
<td>Used Memory</td>
<td>229 MB</td>
</tr>
</tbody>
</table>

Bamboo Version Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>3.0</td>
</tr>
<tr>
<td>Build Number</td>
<td>2301</td>
</tr>
<tr>
<td>Build Date</td>
<td>14/02/11</td>
</tr>
</tbody>
</table>
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.

On this page:
- Updating your Bamboo License Key
- Notes

Updating your Bamboo License Key

To update your Bamboo license key:

1. Click the 'Administration' link in the top navigation bar.
2. Click the 'License Details' link in the left navigation column.
3. This will display your existing Bamboo license details, and an empty box called 'License Key'. Paste your new license into this box.
4. Click the 'Save New License' button.

Notes
- Licensing Questions? Please see the Licensing FAQ.

Related Topics
Configuring System Settings

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.

On this page:
- Specifying Bamboo's Title
- Notes

Specifying Bamboo’s Title

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.

Notes
- Related Topics

Configuring System Settings

Specifying Bamboo's URL

This is the base URL of this installation of Bamboo. All links created (for links in Bamboo email notifications etc.) will be prefixed by this URL.
Specifying Bamboo’s 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.

Notes

- **Accessing Bamboo from Outside a Firewall** — When accessing Bamboo through a web browser, most Bamboo URL links (which provide navigation throughout the product) will use the base URL that was originally entered into your browser’s URL field. For example, to access Bamboo through a web browser on the same machine running Bamboo itself, you may have entered the base URL http://localhost:8085/... into your browser’s URL field. Consequently, most Bamboo URL links will use the base URL http://localhost:8085/... However, URL links to a Bamboo instance that are provided in Bamboo email notifications and by some Bamboo plugins, will use the base URL set on this ‘General Configuration’ page. Hence, if you configure the ‘Base URL’ field above to one that can only be accessed internally, behind a firewall, then you may have problems accessing this Bamboo instance externally.

Related Topics

Configuring System Settings

Logging in Bamboo

The information on this page relates to the **Bamboo server (atlassian-bamboo) logs only**. You cannot change the logging configuration for the build logs.

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 `atllassian-bamboo-agent.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.
  - **atlassian-bamboo Logs for Elastic Agents** — Elastic agent activity is logged inside the elastic instance where the elastic agent runs. To access the elastic agent logs (atlassian-bamboo.log and bamboo-elastic-agent.out) use ssh to log in to your elastic instance as described in Viewing an Elastic Instance and retrieve the logs.

See Locating Important Directories and Files for information on where to find other important files in Bamboo.

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.

Before you begin:

• Note, you will not need to restart your Bamboo server for any logging changes to take effect.

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.
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 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` file and update the value of `log4j.rootLogger` to the desired logging level:
Configure the Location of the atlassian-bamboo Logs

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.appenders.fileLog.file=/my/path/to/atlassian-bamboo.log```

Note that the new log file location applies to both the server and remote agents. If using an absolute path this may result in aggregated logs.

Notes

Related Topics

Configuring System Settings
Locating Important Directories and Files
Viewing a Build’s Artifacts

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.

On this page:

- Enabling GZIP Compression
- Notes

Enabling GZIP Compression

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.

Notes

Related Topics

Configuring System Settings

Enabling Bamboo’s Remote API

You can 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.

Please note, 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 (as described below), if you want to allow access to the deprecated remote API. Access to the new REST API is enabled by default.
Enabling the Remote API

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 under the 'System' section.
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.

Notes

Related Topics

- 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.

Bamboo on UNIX-based operating systems (such as Solaris, Linux or Mac OS X) can be started by either executing the bamboo.sh script or using the Java Service Wrapper packaged with Bamboo.

Bamboo on Windows-based operating systems can be started by running the startup.bat file from the command line (which is the same as running the 'Start in Console' option from the Windows Start menu) or as a Windows Service. Both approaches start Bamboo using the Java Service Wrapper.

Configuring a Bamboo System Property (UNIX-based OS)

Before you begin:

- Bamboo must be shut down before modifying any of its system properties. Once you have modified one or more system properties, they will come into effect when Bamboo is restarted.
  - 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.

To configure a system property via the bamboo.sh file:

1. Open the Bamboo start-up script bamboo.sh in a text editor. (This is usually located at the root of your Bamboo installation directory.)
2. Locate the variable RUN_CMD in bamboo.sh and add the system property as a parameter to the java command string value of RUN_CMD, by adding the '-D' prefix to the system property.
   - For example, if you wanted to set the bamboo.agent.heartbeatInterval system property to 10 (seconds), you would add the parameter -Dbamboo.agent.heartbeatInterval=10 to the java command string value of RUN_CMD such that the RUN_CMD variable assignment in bamboo.sh might look like:
   ```
   RUN_CMD="java -server -Xms256m -Xmx512m -XX:MaxPermSize=256m -Dbamboo.agent.heartbeatInterval=10 -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true -Djetty.port=8085 com.atlassian.bamboo.server.Server 8085 ./webapp /
   ```
3. Save your changes to the bamboo.sh file and (re-)start Bamboo.

Configuring a Bamboo System Property (Windows-based OS)
Before you begin:

- Bamboo must be shut down before modifying any of its system properties. Once you have modified one or more system properties, they will come into effect when Bamboo is restarted.
  - 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.

To configure a system property via the Java Service Wrapper `wrapper.conf` configuration file:

1. Open the Bamboo Wrapper configuration file `wrapper.conf` in a text editor. (This is usually located in the `conf` subdirectory of your Bamboo installation directory.)
2. Locate the set of variables beginning `wrapper.java.additional.X`, where `X` is a series of consecutive numbers starting from `1`. After the final `wrapper.java.additional.X` variable in this set, add a new variable `wrapper.java.additional.Y`, where `Y` is the next consecutive number in this set of variables.
   - For example, if you wanted to set the `bamboo.agent.heartbeatInterval` system property to 10 (seconds), you would add a new variable `wrapper.java.additional.4` to `wrapper.conf` and assign it the value `-Dbamboo.agent.heartbeatInterval=10`, such that this section of the `wrapper.conf` file might look like:

   ```
   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
   # And now for the new variable:
   wrapper.java.additional.4=-Dbamboo.agent.heartbeatInterval=10
   ```

3. Save your changes to the `wrapper.conf` file and (re-)start Bamboo.

Notes

- For general information on configuring Bamboo parameters on start up, please refer to Configuring Bamboo on start-up.

Related Topics

Configuring System Settings

Finding Your Bamboo Support Entitlement Number (SEN)

Your Support Entitlement Number (SEN) is required when raising a support request in our support system: [http://support.atlassian.com](http://support.atlassian.com).

The three ways of finding you SEN are described below.

On this page:

- Method 1 - Check the Bamboo Administration Interface
- Method 2 - Check my.atlassian.com
- Method 3 - Check your Atlassian Invoice
- Notes

Method 1 — Check the Bamboo Administration Interface

To find your SEN via the Bamboo administration interface:

1. Click the 'Administration' link in the top navigation bar of Bamboo.
2. Click the 'License Details' link in the left navigation column under the 'System' section. The SEN is shown, as per the screenshot below.
Method 2 — Check my.atlassian.com

To find your SEN via my.atlassian.com:

1. Log into my.atlassian.com as the Account Holder or Technical Contact for your Bamboo product.
2. The SEN will be shown, as per the screenshot below.
Method 3 — Check your Atlassian Invoice

Your Support Entitlement Number (SEN) appears on the third page of your Atlassian Invoice.

Notes

Related Topics

See Finding Your Support Entitlement Number in the support space for more general information about how Atlassian Support uses this number.

Configuring Gravatar Support

Bamboo is configured to support Gravatars by default. This means that Bamboo will attempt to use user’s emails to retrieve profile pictures from the Gravatar service. The profile pictures will be displayed against user activity, e.g. comments, in Bamboo.

On this page:
- Configuring Gravatar Support
- Notes

Configuring Gravatar Support

To enable/disable Gravatar support:

1. Click the ‘Administration’ link in the top navigation bar.
2. Click the ‘General Configuration’ link in the left navigation column.
3. Enable/Disable the ‘Enable Gravatar Support’ checkbox, as desired.
4. Click the ‘Save’ button.

Notes

Related Topics

Configuring System Settings
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

Other Security Resources

- Page: How to Report a Security Issue
- Page: Securing your Remote Agents
- Page: Security Advisory Publishing Policy
- Page: Elastic Bamboo Security
- Page: Managing Users and Permissions
- Page: Securing your repository connection
- Page: Configuring Tomcat to Use HttpOnly Session ID Cookies
- Page: Managing Bamboo Security
- Page: Severity Levels for Security Issues
- Page: Security Patch Policy
- Page: Bamboo Security Advisories
- Page: Configuring a Plan's Permissions

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
- Notes

Adding an OAuth consumer

Before you begin:

- 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 'Gadgets', 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 (see screenshot below).
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:
     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.
# OAuth Administration

An OAuth consumer is a web site or application that has permission to access Bamboo’s data on behalf of the logged-in user. Typically, the consumer accesses the data via gadgets that support OAuth for authorization. Learn more.

**Enter the consumer’s base URL so we can retrieve its information automatically for you.**

<table>
<thead>
<tr>
<th>Consumer Base URL (required)</th>
<th><a href="http://jira.example.com:8085/bamboo">http://jira.example.com:8085/bamboo</a></th>
</tr>
</thead>
</table>

*Use base URLs for Atlassian applications only. For example, ‘http://jira.mycompany.com’.*

[Add] Return to Consumer List

**Enter the consumer’s information manually.**

- **Consumer Key** (required)
  
  The key supplied by the consumer application. The format of this key is determined by the consumer.

- **Name** (required)
  
  A short name for the consumer site, to help users identify the consumer when granting it access to Bamboo’s data.

- **Description**
  
  For example, the consumer application name and URL, such as ‘JIRA at http://jira.mycompany.com’.

- **Public Key** (required)
  
  The public key, or self-signed certificate, supplied by the consumer application.

- **Callback URL**
  
  The URL supplied by the consumer application (optional). This is the default address in the consumer application that we will go to after the user has approved the OAuth request.

[Add] Return to Consumer List
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.

Notes

Related Topics

Managing Bamboo Security

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.

The sections below explain the default access rules for remote agent instances and how to change these rules, if desired.

On this page:
- Default EC2 Access Rules
- Changing the Default EC2 Access Rules
- Setting up your VCS for Elastic Bamboo
- Notes

Diagram above: Elastic Bamboo security architecture

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').
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.

Notes

Related Topics

Configuring Elastic Bamboo

Managing Bamboo Plugins

A plugin is an add-on to the core Bamboo code, used to extend the Bamboo functionality. Some plugins are shipped with Bamboo, others are available for you to install yourself.

Bamboo comes with the following preinstalled plugins:

- AutoFavourites plugin
- JIRA plugin
- NAnt Builder plugin
- Clover plugin

You can develop additional plugins (see the Bamboo Plugin Guide) or download existing plugins from the Atlassian Plugin Exchange, and install them into your Bamboo system.

On this page:

- Managing your Plugins
- About the Universal Plugin Manager (UPM)
- Notes

Managing your Plugins

- Uninstalling a Plugin
- Upgrading your Existing Plugins
- Checking Plugin Compatibility for Bamboo Upgrades
- Configuring Plugins
- Installing a Plugin
- Disabling or Enabling a Plugin
- Viewing the Plugin Audit Log
- Viewing your Installed Plugins
- Plugin Blacklist

About the Universal Plugin Manager (UPM)

The Universal Plugin Manager (UPM) provides you with a powerful and user-friendly interface to manage your plugins. The Universal Plugin Manager itself is a plugin, which contains a number of modules that are implementations of the Atlassian REST plugin module type. It allows you to perform common plugin tasks, such as:

- Enabling/disabling plugins and their plugin modules.
- Installing new plugins.
- Configuring advanced plugin options.
- Finding out-of-date plugins and updating them.
- Checking the compatibility of your installed plugins against newer versions of the application.

The Universal Plugin Manager also interfaces with the Atlassian Plugin Exchange, so you can browse the wide range of plugins available for your application from within your application. You can install any of these plugins with a single click, or upload your own plugins using the Universal Plugin Manager as well.
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 ‘System’, click the ‘Plugins’ link.

To uninstall a plugin from Bamboo:

1. Click the ‘Manage Existing’ tab. You will see a list of the plugins installed in your application.
2. Click the name of the plugin that you wish to uninstall. The plugin details will appear.
3. Click the ‘Uninstall’ button. The information summary will display an ‘Uninstalling’ message and the plugin will be uninstalled from your application.

Related Topics

Managing Bamboo Plugins

Upgrading your Existing Plugins

Plugins are often developed separately from Bamboo. You may wish to upgrade your plugins to more recent versions to allow them to work with your Bamboo version or simply to take advantage of new features in a plugin version. The Universal Plugin Manager (UPM) provides

Notes

- **Plugin Safety.** Plugins are very powerful. They can change the behaviour of almost any part of the Bamboo server. This makes it very important that you trust a plugin before you install it. Always be aware of where and who a plugin comes from.
- **Troubleshooting.** Having problems with the Universal Plugin Manager? Try the Universal Plugin Manager FAQ. (The link will direct you to the Universal Plugin Manager documentation. Use the back button on your browser to return the Bamboo documentation).

Uninstalling a Plugin

If you wish to completely remove a plugin from Bamboo, you can uninstall it via the Universal Plugin Manager (UPM). If you only want to temporarily remove it, you may wish to disable the plugin instead.

To access the Universal Plugin Manager in Bamboo:

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 ‘System’, click the ‘Plugins’ link.

To uninstall a plugin from Bamboo:

1. Click the ‘Manage Existing’ tab. You will see a list of the plugins installed in your application.
2. Click the name of the plugin that you wish to uninstall. The plugin details will appear.
3. Click the ‘Uninstall’ button. The information summary will display an ‘Uninstalling’ message and the plugin will be uninstalled from your application.

Related Topics

Managing Bamboo Plugins

Upgrading your Existing Plugins

Plugins are often developed separately from Bamboo. You may wish to upgrade your plugins to more recent versions to allow them to work with your Bamboo version or simply to take advantage of new features in a plugin version. The Universal Plugin Manager (UPM) provides
you with a list of plugins that have available upgrades and allows you to upgrade each plugin individually or in bulk.

On this page:

- Upgrading a Plugin
- Upgrading All your Plugins
- Notes

Upgrading a Plugin

To access the Universal Plugin Manager in Bamboo:

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 "System", click the 'Plugins' link.

To upgrade a plugin in Bamboo:

1. Click the 'Upgrade' tab. The plugin upgrades page will appear.
   - If there is a later version of a plugin that you have already installed, this page will show the latest compatible version of the plugin.
   - You can click the plugin name to expand the row and see more information about the plugin.
   - You can filter your list by entering keywords in the 'Filter plugins' text box.
2. Click the 'Upgrade Now' button next to the relevant plugin to update it to the plugin version shown.

Upgrading All your Plugins

To access the Universal Plugin Manager in Bamboo:

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 "System", click the 'Plugins' link.

To upgrade all available plugins in Bamboo:

1. Click the 'Upgrade' tab. The plugin upgrades page will appear.
   - If there is a later version of a plugin that you have already installed, this page will show the latest compatible version of the plugin.
   - You can click the plugin name to expand the row and see more information about the plugin.
   - You can filter your list by entering keywords in the 'Filter plugins' text box.
2. Click the 'Upgrade All' button to update every plugin to the plugin versions shown. **Note:** Some plugins cannot be installed via the Universal Plugin Manager. You must install these plugins manually. These plugins will not be upgraded automatically.

Notes

- If you are considering upgrading Bamboo, you can use the Universal Plugin Manager to check the compatibility of your plugins with your desired Bamboo version. See Checking Plugin Compatibility for Bamboo Upgrades.

Related Topics

Managing Bamboo Plugins

Checking Plugin Compatibility for Bamboo Upgrades

You can use the plugin manager's 'Upgrade Check' to verify that your plugins will still work after a Bamboo upgrade.

For example, if you are thinking of upgrading from Bamboo 3.1 to Bamboo 3.2, the upgrade check can tell you the following:

- Installed plugins that are compatible with Bamboo 3.1 and Bamboo 3.2.
- Installed plugins that are not compatible with Bamboo 3.2 but will be compatible with Bamboo 3.2 if you upgrade them.
- Installed plugins that are not compatible with Bamboo 3.2, even if you upgrade them to their latest version.

To access the Universal Plugin Manager in Bamboo:

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 "System", click the 'Plugins' link.

To check the compatibility of your plugins against different Bamboo versions:

1. Click the 'Upgrade Check' tab.
2. In the 'Check compatibility for' dropdown menu, select the version of your application to check the plugins against.
3. Click the 'Check' button.
4. The page displays any of your installed plugins that are not compatible with the selected application version. The compatibility checker will also check the compatibility of the latest available version of each plugin (if not already upgraded). You can click on the name of any of the plugins to view more information about the plugin.

The plugins are grouped into sections under the following headings:

- **Incompatible** – The installed versions of these plugins are not compatible with the selected application version. There are currently no plugin upgrades available that are compatible with the selected application version.
- **Compatible, if upgraded** – The installed versions of these plugins are not compatible with the selected application version. However, the plugins will be compatible if you upgrade them. There are buttons allowing you to upgrade these plugins.
- **Compatible if both <application> and the plugin are upgraded** – The installed versions of these plugins are not compatible with the selected application version. There is a plugin compatible with the newer application version, but it is not compatible with the application version you are currently running. You must upgrade the application and then upgrade the plugin. There are buttons allowing you to disable these plugins before proceeding with the upgrade.
- **Compatible** – The currently installed versions of these plugins are compatible with the selected application version.
- **Unknown** – These plugins may or may not be compatible with the selected application version. If a plugin is not registered with the Atlassian Plugin Exchange, the Universal Plugin Manager cannot check its compatibility with different application versions.

**Notes**

**Related Topics**

**Managing Bamboo Plugins**

**Configuring Plugins**

A number of Bamboo plugins offer advanced configuration options. If you have one of these plugins installed in your application, you can view and update these configuration options via the Universal Plugin Manager (UPM).

If you would like to disable or enable a plugin, please refer to [Disabling or Enabling a Plugin](#).

---

**Configuring a Plugin**

To access the Universal Plugin Manager in Bamboo:

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 'System', click the 'Plugins' link.

To configure a plugin in Bamboo:

1. Click the 'Manage Existing' tab.
2. Locate the plugin that you want to configure and click its title. The plugin details section will expand.
3. Click the 'Configure' link for that plugin.
   - The link will be disabled if the plugin is disabled.
   - If there is no 'Configure' link, then there are no advanced configuration options available for that plugin.
4. The advanced configuration options for the plugin will appear. Update the configuration settings as desired and save your changes.

**Note:** The advanced configuration screens are provided by the plugin. If you encounter any problems after you click the 'Configure' link, the plugin is responsible for the issue, not the Universal Plugin Manager.

**Configuration Notes for Specific Plugins**

Please refer to the following pages about configuring specific plugins:

- [Enabling the Auto-Favourite Plugin](#)
- [Enabling the Clover Plugin](#)

**Related Topics**

**Managing Bamboo Plugins**

**Enabling the Auto-Favourite Plugin**

If the Auto-Favourite plugin is enabled, then a plan will be automatically added to a user's 'My Bamboo' tab when the user checks code in to the plan's source repository. The user will become a 'watcher' of that plan. Depending on how the plan's notifications are configured, the user may receive notifications about the plan's build results.
Enabling the Auto-Favourite Plugin

To enable the Auto-Favourite plugin:

1. Click the 'Administration' link in the top navigation bar.
2. Click the link 'Autofavourite Plugin' in the left-hand column.
3. Tick the box 'Enable Auto-Favourite Plugin'.
4. Click the 'Save' button.

Notes

- If a user removes the plan from their favourites, it will not be automatically added again.

Related Topics

Configuring Plugins

Enabling the Clover Plugin

This page contains instructions on enabling and configuring the Clover plugin for a Job of a Plan in Bamboo.

Enabling the 'Clover' Plugin on a Plan

To enable the 'Clover' plugin on a plan:

1. Navigate to the desired Job, as described on Configuring a Job.
2. Click the 'Configuration' tab.
3. Click the 'Configure Job' dropdown, then click 'Builder'. You will need Administrator rights to do this.
4. 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.
     - You will also need to insert a Clover license (evaluation licenses are available) into the field provided.
     - '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.

5. Click the 'Save' button.
Installation a Plugin

This page describes how to install a plugin in Bamboo. You can use plugins to customise and extend the functionality of your application.

A number of plugins are available from the Atlassian Plugin Exchange. You can also create your own as described in the Bamboo Plugin Guide.

Adding a plugin from the Atlassian Plugin Exchange

To access the Universal Plugin Manager in Bamboo:

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 ‘System’, click the ‘Plugins’ link.

To find a plugin in the Atlassian Plugin Exchange and add it to Bamboo:

1. Click the ‘Install’ tab in the UPM. You will see a list of featured plugins.
2. Search for your plugin as follows:
   - Enter some keywords that describe the plugin in the ‘Search the Plugin Exchange’ search box and press ‘Enter’.
   - Alternatively, browse to the desired plugin in the list. You can choose ‘Featured’, ‘Popular’, ‘Supported’ (by Atlassian) or ‘All available’ from the ‘Plugins to show’ dropdown to see a different list of plugins.
3. Click the ‘Install’ button for the desired plugin to add it to your application. A confirmation message and the plugin details will appear when the plugin is installed successfully.

Note: You may need to restart your application for your change to take effect. The Universal Plugin Manager will inform you if this is the case.

Note: Not all plugins can be automatically installed. Some required manual installation. These plugins will have a ‘Download’ button instead of an ‘Install’ button. In these cases, you should read and follow the plugin’s installation instructions.
To restart Bamboo with the plugin installed:
1. Shut down Bamboo and then start it up again.

Uploading your own plugin

To access the Universal Plugin Manager in Bamboo:
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 'System', click the 'Plugins' link.

To upload your own plugin to Bamboo:
1. Click the 'Install' tab in the UPM. You will see a list of featured plugins.
2. Click the 'Upload Plugin' link. The 'Upload Plugin' window will appear.
3. Specify the location of your plugin:
   - If the plugin you want to install is on your computer, use the 'Browse' dialogue to choose the plugin JAR file.
   - If you want to install a plugin from a remote location, enter the URL of the plugin JAR file in the 'From this URL' text box.
4. Click the 'Upload' button to upload and enable your plugin. A confirmation message will appear when the plugin is successfully installed.
   
   Note: You may need to restart your application for your change to take effect. The Universal Plugin Manager will inform you if this is the case.

To restart Bamboo with the plugin installed:
1. Shut down Bamboo and then start it up again.

Notes

- In Bamboo, you can install and uninstall both version 1 and version 2 plugins using the Universal Plugin Manager. You will see an 'Install' or an 'Uninstall' button.
- Some entries that you find listed in the Universal Plugin Manager are not actually plugins. These entries will show a 'Download' button which allows you to download the application to your desktop and run it.

Related Topics

Managing Bamboo Plugins

Disabling or Enabling a Plugin

The Universal Plugin Manager (UPM) allows you to disable a plugin in Bamboo without permanently removing it. You can also enable any plugins that have been previously disabled. If you want to add or remove a plugin from your Bamboo site, please refer to Installing a Plugin or Uninstalling a Plugin respectively.

You can also disable all user installed plugins in Bamboo, by enabling safe mode. This may help you to diagnose a plugin-related problem more easily.
Disabling a Plugin

To access the Universal Plugin Manager in Bamboo:

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 'System', click the 'Plugins' link.

To disable a plugin in Bamboo:

1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application. Enabled plugins will have this icon:
2. Locate the plugin that you want to disable and click the title to expand the plugin details section.
3. Click the 'Disable' button.
4. Once a plugin has been disabled, you may need to restart your application for your change to take effect. If so, you will see a message for the plugin, 'Disabled, requires restart'.
   Once the plugin is fully disabled, you will see an 'Enable' link for the plugin.

Enabling a Plugin

To access the Universal Plugin Manager in Bamboo:

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 'System', click the 'Plugins' link.

To enable a plugin in Bamboo:

1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application. Disabled plugins will have this icon:
2. Locate the plugin that you want to enable and click the title to expand the plugin details section.
3. Click the 'Enable' button.
4. Once a plugin has been enabled, you may need to restart your application for your change to take effect. If so, you will see a message for the plugin, 'Enabled, requires restart'.
   Once the plugin is fully enabled, you will see an 'Disable' link for the plugin.

Disabling/Enabling all User Installed Plugins (Safe Mode)

Running your application in safe mode disables all user installed plugins at once. By "user installed plugins", we mean plugins that were not shipped with Bamboo but were installed later via the UPM.

All plugins that were disabled when you entered safe mode will be re-enabled when you exit safe mode.

To access the Universal Plugin Manager in Bamboo:

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 'System', click the 'Plugins' link.

To enable safe mode in Bamboo:

1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application.
2. Click the 'Enable Safe Mode' button.
3. Click the 'Continue' button in the confirmation window. All user installed plugins will be disabled and your application will now be running in 'Safe Mode'.
4. You can now make changes to your installed plugins as desired. For example, you may want to enable/disable specific plugins or plugin modules.
5. Exit safe mode by clicking one of the links in the Safe Mode banner:
   - Click 'Exit Safe Mode and restore the previous configuration' to restore your plugin configuration to its state before you entered Safe Mode.
   - Click 'Exit Safe Mode and keep the current configuration' to keep all changes made to your plugin configuration during Safe Mode.

Related Topics

Managing Bamboo Plugins
Viewing the Plugin Audit Log

The Universal Plugin Manager (UPM) keeps a log of all plugin activity for your Bamboo site. Such activities may be adding plugins, enabling plugins, and so on. You can adjust the period of time for which log entries are kept.

To access the Universal Plugin Manager in Bamboo:
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 'System', click the 'Plugins' link.

To view the plugin audit log:
1. Click the 'Audit Log' tab. The plugin audit log will appear, showing the 25 most recent entries.
2. Use the arrows if you want to view older entries.
3. Click the orange RSS icon if you want to receive the audit log activity in an RSS feed.

Configuring the Plugin Audit Log

To access the Universal Plugin Manager in Bamboo:
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 'System', click the 'Plugins' link.

To configure the length of time for which log entries are kept:
1. Click the 'Audit Log' tab. The plugin audit log will appear.
2. Click the 'Configure purge policy' link.
3. In the 'Purge audit log after' field, specify the number of days for which you wish to keep the logs.
4. Click the 'Confirm' button.

Related Topics

Managing Bamboo Plugins

Viewing your Installed Plugins

Using the Universal Plugin Manager (UPM) you can see a list of plugins installed on your Bamboo site. These include plugins that are bundled with Bamboo as well as any plugins that you have installed. Both enabled and disabled plugins are displayed.

To view your installed plugins:
1. Click the 'Manage Existing' tab. You will see a list of the plugins installed in your application.
   - The plugins are grouped into 'User-installed Plugins' and 'System Plugins'.
   - You can filter your list by entering keywords in the 'Filter visible plugins' text box.
   - The list of 'System Plugins' will be hidden by default. Click the 'Show System Plugins' link to see them.
   - Enabled plugins will have this icon: 📌
   - Disabled plugins will have this icon: 🔒
   - Click the name of a plugin to view the plugin's details.
   - Click 'Enable Safe Mode' to run your application in safe mode. This mode disables all user installed plugins.

Viewing a Plugin's Details
You can view the details for a plugin by clicking the name of a plugin in the list of installed plugins. The summary contains a short description of the plugin as well as links to plugin operations and related information.

- **Plugin key** – A unique key that identifies each plugin in the system.
- **Developer** — The name of the plugin developer and a link to the developer’s home page, if provided by the plugin developer.
- **Plugin version** – The version of the plugin currently installed.
- **Manage plugin modules** — Click this link to display the plugin modules below the plugin summary. A module is a component of the plugin. This link will only appear if the plugin has modules. If you want to enable or disable a plugin module, hover your mouse over the module and click the 'Enable/Disable' button for that module.
- **Configure** – Click this link to display the configuration settings for the plugin. This link is disabled if the plugin is disabled. Please note that not all plugins have settings that can be configured through the Universal Plugin Manager.
- **Enable** – Click this button to enable the plugin in your application. This button will only appear if the plugin is disabled.
- **Disable** – Click this button to disable the plugin in your application. This button will only appear if the plugin is enabled.
- **Uninstall** – Click this button to uninstall the plugin from your application. This button will only appear for user installed plugins.

**Notes**

- **What is the difference between a 'system plugin' and a 'user installed plugin'**?
  - System plugins are shipped with the application. These plugins are essential for the functioning of the system. Although you can disable some of them, you should not do so unless instructed by an Atlassian support engineer. Note: Not every system plugin can be disabled. You cannot uninstall any system plugins.
  - User installed plugins are those which have been installed in the application after it was set up. You can install a plugin either by uploading a JAR file or via the Universal Plugin Manager. You can uninstall these plugins.

**Related Topics**

Managing Bamboo Plugins

**Plugin Blacklist**

Outdated plugins may break certain functionality in Bamboo. If Bamboo detects a presence of a non-working plugin it will print a warning to its logs during startup and ask you to refer to this page.

For more information about why Bamboo printed a particular warning, please refer to a section below that is relevant to the plugin in question.

**Experimental Bamboo Git Plugin**

Since version 3.0 Bamboo is distributed with a fully supported version of the Bamboo Git plugin.

The experimental Bamboo Git Plugin that was available before Bamboo 3.0 (and was not distributed with Bamboo) does not work with Bamboo 3.0 and later.

If you were using the experimental Git Plugin, please remove the plugin from your Bamboo installation, and manually reconfigure each plan that was using it to use the Bamboo Git Plugin that is distributed with Bamboo.

**Using Bamboo with Other Applications**

**Related Topics**

For information on using Bamboo with other applications, see the following topics:

- Embedding Bamboo into Other Applications
- Integrating Bamboo with Other Applications

**Embedding Bamboo into Other Applications**

**Related Topics**

For information on embedding Bamboo into other applications, see the following topics:

- 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
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

   <script type="text/javascript"
   src="<bamboo-base-url>/js/jsBuildSummaryAll.action" >
   </script>

   - For your favourite plans

   <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/

   <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

   <link rel="stylesheet" type="text/css" href="<bamboo-base-url>/styles/bamboo-widget.css" >

2. Place the following script tag in your 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.
suit your needs.

**Example**

Live example from [http://opensource.bamboo.atlassian.com/](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>
```

**RELATED TOPICS**

**Bamboo Documentation**

**Latest Status of a Plan**

This widget allows you to view the current status of a particular plan.

**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-status.css" >
   ```

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/)

   ```html
   <link rel="stylesheet" type="text/css" href="http://opensource.bamboo.atlassian.com/styles/bamboo-status.css" >
   <script type="text/javascript" src="http://opensource.bamboo.atlassian.com/js/showLatestBuildStatus.action?buildKey=STRUTS-MAIN">
   </script>
   ```

**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

```javascript
<script type="text/javascript"
src="<bamboo-base-url>/js/myChanges.action?os_username=<your-user-name>&os_password=<your-password>"></script>
</script>
```

3. Replace `<bamboo-base-url>` with the base url for your bamboo instance.

4. Replace `<your-use-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

   ```javascript
   <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


### Table of Sample Builds

<table>
<thead>
<tr>
<th>Build</th>
<th>When</th>
<th>Comments</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCBUILD-DEF-88</td>
<td>8 minutes ago</td>
<td>20th time lucky!</td>
<td>[details] 2 passed</td>
</tr>
<tr>
<td>SUCBUILD-DEF-83</td>
<td>14 minutes ago</td>
<td>Trying to stop build from failing (TST-11a)</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF-82</td>
<td>17 minutes ago</td>
<td>Altered test files (TST-11a)</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF-81</td>
<td>32 minutes ago</td>
<td>Added failing test to original test suite</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF-75</td>
<td>1 hour ago</td>
<td>*** empty log message ***</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF-74</td>
<td>3 days ago</td>
<td>Bynie's change is related to TST-1a</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF-73</td>
<td>3 days ago</td>
<td>Trying to break build</td>
<td>[details] 4 out of 6 failed!</td>
</tr>
<tr>
<td>AMP-DEF-10</td>
<td>1 week ago</td>
<td>*** empty log message ***</td>
<td>[details] 3 passed</td>
</tr>
<tr>
<td>TEST-DEF-60</td>
<td>1 week ago</td>
<td>*** empty log message ***</td>
<td>[details] 2 out of 4 failed!</td>
</tr>
<tr>
<td>SUCBUILD-DEF2-14</td>
<td>1 week ago</td>
<td>*** empty log message ***</td>
<td>[details] 2 passed</td>
</tr>
</tbody>
</table>
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
   * LAST_30_DAYS
   * LAST_90_DAYS
   * ALL_BUILDS

Example

Live example from http://opensource.bamboo.atlassian.com/

```javascript
<script type="text/javascript"
src="http://opensource.bamboo.atlassian.com/js/viewCombinedByBuildNumberChart.action?buildKey=STRUTS-MAIN&filterController.selectedFilterKey=LAST_25_BUILDS">
</script>
```

Integrating Bamboo with Other Applications

Related Topics

For information on integrating Bamboo with other applications, see the following topics:

- 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.

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 integrations (which are bundled with JIRA 4.0 and Bamboo 2.4). We strongly advise you to upgrade JIRA to version 4.0 or later and Bamboo to version 2.4 or later, 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 integration on the Bamboo Server
3. Add your JIRA Server as an OAuth Consumer in Bamboo

On this page:

- Before You Begin
- 1. Configure the Bamboo plugin on the JIRA Server
- 2. Configure the JIRA integration on the Bamboo Server
- 3. Add your JIRA Server as an OAuth Consumer in Bamboo
- Try your New Configuration
- Need help?

Before You Begin
Security Considerations

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 integration 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.

Configuring an upgraded version of the Bamboo Plugin for JIRA

This note applies only if you have downloaded and installed a later version of the JIRA Bamboo plugin on your JIRA site. Please ignore this note if you are using the same plugin as was shipped with JIRA. If you have installed version 4.2 or later of the JIRA Bamboo plugin, then you will need to configure the plugin via Application Links in JIRA.

1. Follow the JIRA instructions to configure the application link in JIRA.
2. Follow the instructions in step 2 below to configure the JIRA integration on the Bamboo server.
3. That's it. Try out your new configuration. See below.

Known Issues

- Deploying multiple Atlassian applications in a single Tomcat container is not supported. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see this FAQ for more information).

We also strongly recommend that you do not deploy multiple Atlassian applications in a single Tomcat container for a number of practical reasons. Firstly, you will need to shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in the Tomcat container will be inaccessible.

Tips JIRA and Bamboo cannot run in the same Tomcat instance due to a known issue with the Bamboo plugin for JIRA (see JIRA-19662).

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).

Before you begin:

- 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.

To configure your Bamboo plugin in JIRA:

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.

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, you need to configure the JIRA integration on your Bamboo server so that you can view JIRA information in Bamboo.
2. Configure the JIRA integration on the Bamboo Server

The JIRA integration in Bamboo is shipped with Bamboo, so you do not need to download and install anything. 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 in Bamboo:

1. Launch your Bamboo site, 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.

Try your New Configuration

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 and 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)
  - Connecting Bamboo to an external database
  - Hardware sizing considerations
  - Running the Setup Wizard

Bamboo Upgrade Guides

- Bamboo Upgrade Guides
  - Bamboo Generic Upgrade Guide
Bamboo Installation Guide

On this page:

1. Check the System Requirements
   - Supported Platforms
   - Hardware Requirements
   - Servlet Container Requirements
2. Choose your Bamboo ‘Distribution’:
   - Standalone Distribution
   - EAR-WAR Distribution
3. Next Steps
   - Next Steps for Standalone Distribution
   - Next Steps for EAR-WAR Distribution
4. Related Topics
5. Checking for Known Issues and Troubleshooting the Bamboo Installation

1. Check the System Requirements

Supported Platforms

Please read the Supported Platforms page before you install Bamboo. The Supported Platforms page lists the applications servers, databases, operating systems, web browsers and JDKs that we have tested Bamboo with and recommend.

Hardware Requirements

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.

Servlet Container Requirements

If you are using the Bamboo EAR-WAR distribution, you will need a servlet container that supports the Servlet 2.4 specification. Most modern containers should comply to this.

2. Choose your Bamboo ‘Distribution’:

Bamboo is available in two ‘distributions’ — Standalone or EAR-WAR. The Standalone distribution is recommended (even for organisations with an existing application server environment).

Standalone Distribution
Bamboo 3.1 Documentation

- Pre-packaged with the Jetty application server
- Requires virtually no setup
- Recommended for all users

**EAR-WAR Distribution**

- Deploys into an existing application server
- Requires manual configuration
- Suitable only for system administrators

### 3. Next Steps

#### Next Steps for Standalone Distribution

- Standalone Installation Guide — Windows, or
- Standalone Installation Guide — Linux, or
- Standalone Installation Guide — Mac

#### Next Steps for EAR-WAR Distribution

- EAR-WAR Installation Guide

### Related Topics

- Running the Setup Wizard
- Upgrade Guide
- Bamboo Remote Agent Installation Guide
- Release Notes

### Checking for Known Issues and Troubleshooting the Bamboo Installation

If something is not working correctly after you have completed the steps above to install Bamboo, please check for known Bamboo issues and try troubleshooting your upgrade as described below:

- **Check for known issues.** Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases we publish information about the known issues in the Bamboo Knowledge Base. Please check the known issues in the Bamboo Knowledge Base and follow the instructions to apply any necessary patches if necessary.

- **Did you encounter a problem during the Bamboo installation?** Please refer to the guide to troubleshooting upgrades in the Bamboo Knowledge Base.

- If you encounter a problem during the upgrade and cannot solve it, please create a support ticket and one of our support engineers will help you.

### Bamboo Remote Agent Installation Guide

This page describes how to install the Bamboo Remote Agent manually.

Before you begin:

- **Not sure whether to install a remote agent?** See Configuring Agents and Capabilities 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
- Step 3. Configure the Remote Agent's Capabilities
- Step 4. (Optional) Rename the Remote Agent
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:

Installing a Remote Agent

To install a remote agent, please follow these instructions:

1. Ensure that you have Java Runtime Environment 5.0 or later installed on the agent machine.
2. Download the remote agent JAR file to a directory on the agent machine.

Running a Remote Agent

Once installed, you can run the remote agent by executing the following command line from the directory containing the remote agent JAR file:

```
java -jar atlassian-bamboo-agent-installer-3.0-SNAPSHOT.jar
http://bamboo-host-server:8085/agentServer/
```

This will start a service wrapper for your agent, which will automatically restart in case of failure. You may also add extra system properties like -Dbamboo.home... to customise the agent's home location. For more information, see our Bamboo Remote Agent Installation Guide

Running the agents without the service wrapper

For customers wanting to run the Bamboo agent without the service wrapper, the direct agent JAR is available at bamboo-agent-3.0-SNAPSHOT.jar. For more information about using the JAR directly please consult our documentation.

Screenshot above: Installing a remote agent

6. Click the 'DOWNLOAD Remote Agent JAR' button and save the JAR file to the directory you created in step 1.1.
7. Note the command under the heading 'Running a Remote Agent' for use in step 2 below.

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 -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.

⚠️ If you are having issues bringing up the Agent, then take a look at our troubleshooting guide.

Changing where the remote agent stores its data
By default, the remote agent will store its data in a USER_HOME/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.](image)

### Changing 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 [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.](image)

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.](image)

### Running the remote agent with different start-up commands
### The Remote Agent Supervisor

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 -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 -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.

### Configuring Remote Agent Capabilities via bamboo-capabilities.properties

You can define the capabilities for a specific remote agent by configuring a `bamboo-capabilities.properties` file on the agent machine. When the bamboo agent starts up, it will look in the current runtime directory (i.e. `<bamboo-agent-home>/bin`) for a file named `bamboo-capabilities.properties`. The capabilities defined in that file will then be published for the bamboo agent after registering.
We are aware of an issue that prevents a remote agent capability from being updated once it has been added via the `bamboo-capabilities.properties` file. If you choose to add capabilities via the `bamboo-capabilities.properties` file, you will only be able to update them by deleting the capability via Bamboo and restarting the remote agent. Please see BAM-4213 for further details.

To configure a new agent-specific

1. Shut down the remote agent, if it is running.
2. Create a file named `bamboo-capabilities.properties` on the agent machine.
3. Edit the `bamboo-capabilities.properties` file and add the desired capabilities to the agent as desired. You need to follow the capability formats below in the file:
   - **Tip:** Use `\` to escape the spaces, periods and backslashes (`\`).
   - **JDK capabilities** — system.jdk.JDK\ `<jdk number>`=<`jdk location`>
     
     Examples:
     ```plaintext
     system.jdk.JDK\ 1.6=/System/Library/Frameworks/JavaVM.framework/Versions/1.6
     system.jdk.JDK\ 1.6=C:\\Program Files\\Java\\jdk6.0.17
     ```
   
   Note the double backslashes in the example above in the path.
   - **Builder capabilities** — system.builder.<`builder type`>.<`builder label`>=<`builder path`>
     
     Examples:
     ```plaintext
     system.builder.ant.Ant=/opt/apache-ant-1.7.1
     system.builder.maven.Maven\ 1=/opt/maven-1.0.2
     system.builder.mvn2.Maven\ 2=/opt/maven-2.0
     ```
   - **Perforce capabilities** — system.perforce=<`perforce executable location`>
     
     Example:
     ```plaintext
     system.perforce=/usr/bin/p4
     ```
   - **Custom capabilities** — <`custom capability name`>=<`custom capability value`>
     
     Example:
     ```plaintext
     system.os=osx
     ```
4. Save your changes to the `bamboo-capabilities.properties` file.
5. Start up your remote agent. The capabilities defined in the `bamboo-capabilities.properties` file will be configured for your agent.

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 and Enabling 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-2.0-SNAPSHOT.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 EAR-WAR Installation Guide

The Bamboo EAR-WAR distribution is intended for deployment into an existing J2EE application server. It is assumed that you already know how to deploy a web application on the application server of choice. If not, we recommend that you install the Bamboo Standalone distribution.

The following instructions are only indicative of the process and examples are based on installing the Bamboo WAR file on the Apache Tomcat application server. Deployment and configuration will differ from the procedure below if you choose to deploy the Bamboo EAR-WAR distribution on another (unsupported) application server.

On this page:

- Before You Begin
- 1. Download and Install Bamboo EAR-WAR
- 2. Set the Bamboo Home
- 3. Set jms.broker.uri
- 4. Set Java OPTs
- 5. Edit the Bamboo Context Descriptor (Tomcat)
- 6. Configure Bamboo

Before You Begin

- Please ensure that you have read the Requirements section of the Bamboo Installation Guide.
- Deploying multiple Atlassian applications in a single Tomcat container is not supported. We do not test this configuration and upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration (see this FAQ for more information).
- We also strongly recommend that you do not deploy multiple Atlassian applications in a single Tomcat container for a number of practical reasons. Firstly, you will need to shut down Tomcat to upgrade any application and secondly, if one application crashes, the other applications running in the Tomcat container will be inaccessible.

1. Download and Install Bamboo EAR-WAR

1. Download Bamboo WAR for your operating system. Bamboo WAR files for Linux, Mac OS X and Windows are available for download from the Bamboo Download Center. Click the tab for your operating system, click the ‘Show all’ link to display the WAR file and select the Bamboo WAR file to download.
2. Deploy onto your application server by using either of the following methods:
   - Place the WAR file directly into the webapps folder of Tomcat. Tomcat will perform all the necessary extractions when it starts.
   - Extract the WAR file to your chosen directory in the webapps folder. This directory is referred to as the installation directory (i.e. `<bamboo-install>`)

Extraction of the Bamboo WAR

- 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.

By default, the WAR file will extract to a folder called `Bamboo<version>`. The name of the directory in the webapps folder will form the URL required to access Bamboo (e.g. `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.

2. Set the Bamboo Home

Set your Bamboo Home Directory. You can do this in one of three ways:

- Set the `bamboo.home` property in the file `/WEB-INF/classes/bamboo-init.properties` to your chosen Bamboo home directory.
- Pass the Bamboo home directory to the application server as a java opt. (e.g. `-Dbamboo.home=C:/bamboo/bamboo-home`.
- Specify an environment variable ‘BAMBOO_HOME’ which specifies the absolute path to your [BAMBOO_HOME] directory.

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:

```plaintext
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.

4. Set Java OPTs

We recommend 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 are 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` — *(Unix systems only)* 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:

```bash
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:

```bash
JAVA_OPTS="-server -XX:MaxPermSize=256m -Dbamboo.home=/opt/bamboo/bamboohome -Xmx512m -Djava.awt.headless=true $JAVA_OPTS"
export JAVA_OPTS
```

5. Edit the 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 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, if you are using Tomcat 6 you must create the Catalina and localhost directories.**

```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.**

2. For `docBase`, specify the `<bamboo-install>` absolute path that you noted down earlier.

6. Configure Bamboo

1. Shut down and then restart your application server.
3. Configure Bamboo via the Setup Wizard which will display. Read Running the Setup Wizard for further instructions.

Configuring Tomcat to Use HttpOnly Session ID Cookies

Bamboo Standalone distributions from version 2.5.5 now enforce the `HttpOnly` flag on session ID cookies by default, as a means to minimise the risk of common XSS attacks. For more information about this feature, please refer to the Bamboo Security Advisory 2010-05-04.
If you are running the Bamboo EAR-WAR distribution on Tomcat (or another application server that is unsupported), it is likely that Bamboo's session ID cookies will not be transmitted with the HttpOnly flag. Hence, to mitigate the risk of common XSS attacks, we recommend that you configure your application server to transmit HttpOnly session ID cookies.

**Configuring Tomcat to use HttpOnly Session ID Cookies**

To configure your Bamboo EAR-WAR distribution running on Tomcat to use HttpOnly Session ID Cookies:

1. Shutdown the Bamboo service running on Tomcat and the Tomcat application server.
2. Open the `context.xml` file of the Tomcat installation running Bamboo in a text editor. This file is typically located in the `conf` subdirectory of the main Tomcat installation directory.
3. Add the following `Manager` element within the `Context` element of this file:

   ```xml
   <Manager useHttpOnly="true"/>
   ```

   To disable HttpOnly Session ID cookies, either remove this `Manager` element or change the value of its `useHttpOnly` parameter to `false`.
4. Save your changes to the `context.xml` file and restart Bamboo.

**Bamboo Standalone Installation Guide (Linux)**

This page contains instructions to help you install Bamboo Standalone on Linux.

On this page:
- 1. Download and Install Bamboo Standalone
- 2. Launch Bamboo Standalone on Linux
   - Launch via `bamboo.sh` startup script
   - Launch via Java Service Wrapper
- 3. Configure Bamboo

1. Download and Install Bamboo Standalone

   **Before you begin**
   Please ensure that you have read the Requirements section of the Bamboo Installation Guide.

   1. Download Bamboo Standalone for Linux. Bamboo Standalone for Linux is available for download from the Bamboo Download Center (click the 'Linux' tab).
   2. Extract the files from the Linux archive (atlassian-bamboo-x.x-standalone.tar.gz) to a Bamboo installation directory of your choice. By default, the root directory of the tar file is "Bamboo".
   3. Set up your Bamboo home directory — this is the directory where Bamboo will store its configuration data. To do this, open the file `<Bamboo installation directory>/webapp/WEB-INF/classes/bamboo-init.properties` and insert the property "bamboo.home", with an absolute path to your Bamboo home directory. Your file should look something like this:

   ```properties
   bamboo.home=/test/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.

   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.

2. Launch Bamboo Standalone on Linux

   There are two ways you can launch Bamboo on Linux — via a startup script or via a Java Service Wrapper:
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.

Launch 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 `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

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.

3. Configure Bamboo

   1. Access your running Bamboo instance by going to your web browser and entering the address: `http://localhost:8085/`
   2. Configure Bamboo via the Setup Wizard which will display. Read Running the Setup Wizard for further instructions.

Bamboo Standalone Installation Guide (Mac)

This page contains instructions to help you install Bamboo Standalone on Mac OS X.

On this page:

- 1. Download and Install Bamboo Standalone
- 2. Launch Bamboo on Mac OS X
- 3. Configure Bamboo

1. Download and Install Bamboo Standalone

   Before you begin
   Please ensure that you have read the Requirements section of the Bamboo Installation Guide.

You can choose to install Bamboo via a Mac OS X Installer (.dmg) or a TGZ Archive (.tgz):

Download and Install Bamboo Standalone for Mac OS X (Mac OS X Installer)

1. Download Bamboo Standalone for Mac OS X. Bamboo Standalone for Mac OS X is available for download from the Bamboo Download Center (click the 'Mac OS X' tab). Choose the Mac OS X Installer (.dmg) download.
2. Launch the Bamboo Mac OS X installer (atlassian-bamboo-x.x-standalone.dmg). This will mount the Atlassian Bamboo installation volume.
3. Launch the Bamboo Continuous Integration Server Installer.app to begin the installation wizard. 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.

**Download and Install Bamboo Standalone for Mac OS X (TGZ Archive)**

1. **Download** Bamboo Standalone for Mac OS X. Bamboo Standalone for Mac OS X is available for download from the Bamboo Download Center (click the 'Mac OS X' tab). Choose the TGZ Archive (.tgz) download (click the 'Show all' link to show the TGZ Archive' download link).
2. **Extract** the files from the Mac OS X archive version (atlassian-bamboo-x.x-standalone.tgz) to a Bamboo installation directory of your choice. By default, the root directory of the tgz file is "Bamboo".
3. **Set up 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.

4. **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 X**

There are two ways you can launch Bamboo on Mac OS X:

**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. `.bash/bamboo.sh 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.

**Launch 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.

**Step 3. Configure Bamboo**

1. **Access** your running Bamboo instance by going to your web browser and entering the address: `http://localhost:8085/`.
2. **Configure** Bamboo via the Setup Wizard which will display. Read Running the Setup Wizard for further instructions.

**Bamboo Standalone Installation Guide (Windows)**
This page contains instructions to help you install Bamboo Standalone on Windows.

On this page:

- **1. Download and Install Bamboo Standalone**
  - Download and Install Bamboo Standalone for Windows (Windows Installer)
  - Download and Install Bamboo Standalone for Windows (ZIP Archive)
- **2. Launch Bamboo**
  - Launch via the Start Menu
  - Launch via batch file
- **3. Configure Bamboo**

## 1. Download and Install Bamboo Standalone

Before you begin:

- Please ensure that you have read the [Requirements section](#) of the Bamboo Installation Guide.
- Note, you can choose to install Bamboo via a **Windows Installer(.exe)** or a **ZIP Archive (.zip)**.

### Download and Install Bamboo Standalone for Windows (Windows Installer)

**Note, if you wish to run Bamboo on a Windows x64 platform, make sure that you download the 64-bit version of Bamboo distribution.**

1. Download Bamboo Standalone for Windows. Bamboo Standalone for Windows is available for download from the [Bamboo Download Center](#). Choose the **Windows Installer (.exe)** download.
2. Launch the Bamboo Windows installer ([atlassian-bamboo-x.x-standalone.exe](#)) to begin the installation wizard.
3. 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**.

### Download and Install Bamboo Standalone for Windows (ZIP Archive)

1. Download Bamboo Standalone for Windows. Bamboo Standalone for Windows is available for download from the [Bamboo Download Center](#). Choose the **ZIP Archive (.zip)** download (click the 'Show all' link to show the 'ZIP Archive' download link).
2. Extract the files from the ZIP Archive ([atlassian-bamboo-x.x-standalone.zip](#)) to a **directory of your choice**. By default, the root directory in your zip file is named "Bamboo".

**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).

3. Set up 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.

4. 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.
2. Launch Bamboo

There are two ways you can launch Bamboo on Windows:

Launch via the Start Menu

If you have used the 'Windows Installer' to install Bamboo, you can start Bamboo via the Start Menu in Windows (generally under the 'Bamboo' folder by default). The following options will be available in your Start Menu:

- 'Bamboo Continuous Integration Server Uninstaller' — uninstalls Bamboo from your computer
- 'Install Service' — installs Bamboo as a Windows service (note, this will not start Bamboo)
- 'Remove Service' — removes the Bamboo Windows service, if you have previously installed it (note, Bamboo will not be uninstalled from your computer)
- 'Start in Console' — starts Bamboo in a Windows console
- 'Start Service' — starts your installed Bamboo Windows service
- 'Stop Service' — stops your installed Bamboo Windows service

You can run Bamboo in two modes, either in a Windows console or as a Windows service:

- To run Bamboo in a Windows console, click the 'Start in Console' option.
- To run Bamboo as a Windows service, click the 'Install service' option. After the service is installed, click 'Start Service'. Once you have installed Bamboo as a service, Bamboo will start up automatically every time Windows restarts.

Launch via batch file

You can start Bamboo via the batch files that are shipped with Bamboo. If you have installed Bamboo via the ZIP Archive, you will need to use the batch files to start Bamboo. You can find the following batch files in your installation directory:

- '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.

You can run Bamboo in two modes, either in a Windows console or as a Windows service:

- To run Bamboo in a Windows console, run 'BambooConsole.bat'
- To run Bamboo as a Windows service, run 'InstallAsService.bat'. After the service is installed, run 'StartBamboo.bat'. Once you have installed Bamboo as a service, Bamboo will start up automatically every time Windows restarts.

3. Configure Bamboo

1. Access your running Bamboo instance by going to your web browser and entering the address: http://localhost:8085/

Configure Bamboo via the Setup Wizard which will display. Read Running the Setup Wizard for further instructions.

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.

When you launch Bamboo for the first time, the Bamboo setup wizard will display. The wizard will lead you through the Bamboo settings that you need to configure before you can start using it.

Step 1. License Details and Setup Method

You must have a valid Bamboo license (evaluation or commercial) to 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.

Once you have entered a valid license key, you can choose which setup method you prefer for your Bamboo installation:

- 'Express Installation' — the 'Express Installation' method only requires you to enter the bare minimum of configuration information, as it sets up Bamboo with default settings and an embedded database (HSQL). We recommend that you choose this method if you are evaluating or demonstrating Bamboo.
  - If you choose, the 'Express Installation' method you can skip to 'Step 6. Set Up Administrator User'.
- 'Custom Installation' — the 'Custom Installation' method is much longer, but allows you to configure Bamboo with an external database, customise the default settings, and/or initialise the server with your own data. We recommend that you choose this method if you are setting up a production instance of Bamboo.
  - If you choose, the 'Custom Installation' method, proceed to 'Step 2. General Configuration' below.

Step 2. General Configuration

This step applies to the 'Custom Installation' method only.

On this page, you need to specify a number of Bamboo server settings, such as the address of the server, where data is stored and the message broker used to communicate with remote agents.

- 'Name' — for more details please see Specifying Bamboo’s Title.
- 'Base URL' — for more details please see Specifying Bamboo’s URL.
- '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 field will only display if you are permitted remote agents under your Bamboo license. The Broker URL 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. You can update this file if you want to change your Broker URL.
  - 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.

Step 3. Choose a Database Configuration
This step applies to the 'Custom Installation' method only.

Screenshot: Choose Database Configuration (click to view larger image)

Picking a database configuration is an important choice. If you pick the 'Embedded Database' configuration, you do not have to set up a database. However, the embedded HSQL database is only suitable for evaluation purposes. You will need to move to an external database, if you decided to deploy Bamboo in production at a later stage (as described in Moving your Bamboo Data to a Different Database).

- 'Embedded Database' — Choose this for quick and easy first-time installation of Bamboo. This option is suitable for evaluation purposes only.
  - If you choose, the 'Embedded Database' method you can skip to 'Step 5. Starting Data'.
- 'External Database' — Choose this if you wish to use an external database.
  - If you choose, the 'External Database' method, proceed to 'Step 4. Database Configuration' below.

Step 4. Database Configuration

Screenshot: Database Configuration (click to view larger image)

If you selected 'External Database' in the previous 'Choose a Database Configuration' screen, you will need to provide the configuration details for your database. Please see Connecting Bamboo to an external database for further instructions.

Step 5. Starting Data

Screenshot: Starting Data (click to view larger image)

On this page, you need to specify how Bamboo will populate the 'home directory' that you set up when you installed Bamboo.

- 'Create new Bamboo home' — choose this if you are performing a normal installation or upgrade.
- 'Import existing data' — only choose this under exceptional circumstances, e.g. if you are connecting Bamboo to a different database, or moving your pre-existing Bamboo installation to a different server.

Step 6. Set Up Administrator User

Screenshot: Set Up Administrator User (click to view larger image)

The final step of the setup wizard is to 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.

Once you have entered the details of your administrator user, click the 'Finish' button. The Bamboo dashboard will display.

Congratulations, you have successfully set up Bamboo!

Next Steps
If you are new to Bamboo, we recommend that you read the Bamboo 101 guide. The Bamboo 101 guide will walk you through the key features of Bamboo and how to set up and use them.

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.

Connecting Bamboo to an external database

Bamboo can be connected to an external database. For details and instructions please see:

- PostgreSQL 8.2
- MySQL 5.x
  - Tomcat and External MySQL Datasource Example
- Oracle 10g and 11g
- Microsoft SQL Server 2005 and 2008
- How do I connect Bamboo to an unsupported database?

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.

PostgreSQL 8.2

These instructions will help you connect Bamboo to a PostgreSQL 8.2+ database. (PostgreSQL 8.0 and PostgreSQL 8.1 are not supported for use with Bamboo.)

Please note, the JDBC driver for PostgreSQL 8.2 (PostgreSQL Driver 8.4.x) is bundled with Bamboo. You do not have to download and install the driver.

1. Configuring PostgreSQL
   • Accept remote TCP connections (remote PostgreSQL server only)
   • Creating a Bamboo database
2. Connecting Bamboo to PostgreSQL
   • Connecting via JDBC
   • Connecting via a datasource

Accept remote TCP connections (remote PostgreSQL server only)

If you are connecting Bamboo to a remote PostgreSQL server (i.e. if your PostgreSQL server is not installed locally on your Bamboo server host system), you will need to configure your data/postgresql.conf and data/pg_hba.conf files to accept remote TCP connections from your Bamboo server’s IP address.

The following PostgreSQL documentation contains information on the appropriate listen_addresses value in the postgresql.conf file as well as the pg_hba.conf file:

- PostgreSQL 8.2 documentation — Connections and Authentication

Once you have modified your data/postgresql.conf and data/pg_hba.conf files, you will need to restart PostgreSQL for your
changes to take effect.

Creating a Bamboo database

```
sudo -s -H -u postgres
# Create the Bamboo user:
/opt/PostgreSQL/8.3/bin/createuser -S -d -r -P -E bamboouser
# Create the bamboo database:
/opt/PostgreSQL/8.3/bin/createdb -O bamboouser bamboo
exit
```

Creating a completely empty bamboo database is recommended. Avoid using templates to create the database as some may insert default tables which can lead to conflicts when setting up Bamboo.

2. Connecting Bamboo to PostgreSQL

Bamboo provides two ways to connect to a PostgreSQL database — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

**Connecting via JDBC**

To connect Bamboo to a PostgreSQL database, via JDBC,

1. Run the **Setup Wizard** and choose the 'Custom Installation' method.
2. On the **Choose a Database Configuration** page, choose 'External Database', select 'PostgreSQL 8.2 and above' from the list and click the 'Continue' button. The **Database Configuration** page appears.
3. Ensure that 'Direct JDBC connection' has been chosen and complete the following fields (as shown in the screenshot below):
   - 'Driver Class Name' — Type the following: `org.postgresql.Driver` (if different from the default).
   - 'Database URL' — Type the URL where Bamboo will access your database (if different from the default). For details about syntax, please refer to 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.
4. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click 'Continue' to finish specifying your connection settings.

**Screenshot 1: 'Setup JDBC Connection (PostgreSQL)'**

**Database Configuration**

Choose how you wish Bamboo to connect to your database

<table>
<thead>
<tr>
<th>Select Database Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct JDBC connection</td>
</tr>
<tr>
<td>Connect via a datasource</td>
</tr>
</tbody>
</table>

**Driver Class Name:** `org.postgresql.Driver`

The class name of the database driver. Ensure that this class is in your class path.

**Database URL:** `jdbc:postgresql:localhost:5432:bamboo`

The url to access the database.

**User Name:**

The username to access the database.

**Password:**

(Optional) Enter the password if the database configuration requires it.

- [ ] Overwrite existing data

If you wish Bamboo to overwrite any existing tables that may exist in the database.

**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 details about the syntax to use for the JDBC database URL, please see the Postgres JDBC driver documentation.
2. Run the Setup Wizard and choose the 'Custom Installation' method.
3. On the Choose a Database Configuration page, choose 'External Database', select 'PostgreSQL 8.2 and above' from the list and click the 'Continue' button. The Database Configuration page appears.
4. Choose 'Connect via a datasource (configured in the application server)' (as shown in the screenshot below).
5. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
6. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click 'Continue' to finish specifying your connection settings.

Screenshot 2: 'Setup Datasource Connection'

### MySQL 5.x

These instructions will help you connect Bamboo to a MySQL 5.x database. (MySQL 4.1 is not a supported database for use with Bamboo.)

Please note, the JDBC driver for MySQL 5.x (JDBC Connector/J 5.1) is bundled with Bamboo. You do not have to download and install the driver.

1. Creating and Configuring the MySQL database
2. Connecting Bamboo to the MySQL database
   - Connecting via JDBC
   - Connecting via a datasource

### 1. Creating and Configuring the MySQL database

To connect Bamboo to an external MySQL database, you must first create and configure it. This database must be configured to use:

- utf8 character set encoding, instead of latin1
- utf8_bin collation
- the InnoDB storage engine

If your MySQL database server is configured to use a storage engine other than InnoDB by default (such as MyISAM), then if possible change it to use InnoDB. Otherwise, you can configure Bamboo's JDBC connection to your MySQL database so that any tables which Bamboo creates in this database will be done using the InnoDB database engine.

A MySQL database administrator can easily create and configure a MySQL database for Bamboo by running the following MySQL commands:

```
mysql> CREATE DATABASE bamboo CHARACTER SET utf8 COLLATE utf8_bin;
mysql> GRANT ALL PRIVILEGES ON bamboo.* TO 'bamboouser'@'localhost' IDENTIFIED BY 'password';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```
This creates an empty MySQL database for Bamboo named bamboo.

Please Note:

- bamboouser — the user account name for the Bamboo MySQL database
- localhost — the host name of the MySQL database server
- password — the password for this user account
- If the MySQL database and Bamboo servers are on the same physical computer, you can use localhost and not set a password by omitting IDENTIFIED BY ‘password’ from the 2nd MySQL statement above (if you trust the security within this computer).

For more information about configuring character set encoding and collation for Bamboo MySQL databases, please refer to the MySQL 5 documentation — Specifying Character Sets and Collations.

2. Connecting Bamboo to the MySQL database

Bamboo provides two ways to connect to a MySQL database — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

Connecting via JDBC

To connect Bamboo to a MySQL database, via JDBC,

1. Run the Setup Wizard and choose the 'Custom Installation' method.
2. On the Choose a Database Configuration page, choose 'External Database', select 'MySQL 5.x' from the list and click the 'Continue' button. The Database Configuration page will appear.
3. Ensure that 'Direct JDBC connection' has been chosen and complete the following fields (as shown in the screenshot below):
   - 'Driver Class Name' — Type the following: com.mysql.jdbc.Driver (if different from the default).
   - 'Database URL' — Type the URL where Bamboo will access your database (if different from the default). Your URL must include the autoReconnect=true flag. If you intend to use non-Latin characters in Bamboo, ensure that your URL includes the useUnicode=true and characterEncoding=utf8 flags.
   - If your MySQL database server is configured to use a storage engine other than InnoDB by default, ensure that your URL includes the sessionVariables=storage_engine=InnoDB flag.
   - If you include all of these flags, your Database URL should look similar to: jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8&sessionVariables=storage_engine=InnoDB

   If the autoReconnect=true flag is not specified, 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. This is bamboouser defined in section 1 (above).
   - 'Password' — Type the password (if required) that Bamboo will use to access your database. This is password defined in section 1 (above). Leave this field blank if a password for the database user account was not specified.
4. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click 'Continue' to finish specifying your connection settings.

Screenshot 1: 'Setup JDBC Connection (MySQL)'
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).
   - Ensure that the JDBC URL which you configure in your application server includes the `autoReconnect=true, useUnicode=true and characterEncoding=utf8` flags, such that your database URL should look similar to:
     ```
jdbc:mysql://localhost/bamboo?autoReconnect=true&useUnicode=true&characterEncoding=utf8
```
   - If your MySQL database server is configured to use a storage engine other than InnoDB by default, also include the `sessionVariables=storage_engine=InnoDB` flag in this URL.
   - 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.
   - For more information on the URL syntax, please see the MySQL documentation.

2. Run the Setup Wizard and choose the 'Custom Installation' method.
3. On the Choose a Database Configuration page, choose 'External Database', select 'MySQL 5.x' from the list and click the 'Continue' button. The Database Configuration page appears.
4. Choose 'Connect via a datasource (configured in the application server)' (as shown in the screenshot below).
5. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
   - If `java:comp/env/jdbc/DataSourceName` does not work, try `jdbc/DataSourceName` (and vice versa).
6. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click 'Continue' to finish specifying your connection settings.

Datasource example
You can see an example of Tomcat with a MySQL database as a datasource in the following document: Tomcat and External MySQL Datasource Example.
Tomcat and External MySQL Datasource Example

Within the Context tags of your context descriptor (bamboo.xml), directly after the opening `<Context...>` line, insert the DataSource Resource tag:

```xml
_Context .... >
<Resource name="jdbc/bamboo" auth="Container" type="javax.sql.DataSource"
  username="yourusername"
  password="yourpassword"
  driverClassName="com.mysql.jdbc.Driver"
  url="jdbc:mysql://localhost:3306/bamboo?autoReconnect=true"
  maxActive="15"
  maxIdle="7"
  validationQuery="Select 1" />
</Context>
```

Oracle 10g and 11g

These instructions will help you connect Bamboo to an Oracle 10g or 11g database. Oracle 9i is no longer a supported database for use with Bamboo and the 11.2.x drivers from Oracle do not support 9i.

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.

Important

For JDBC or JNDI connections, please ensure that the user connecting to the database will have total permissions over it. This includes DBMS_LOB package and other resources available.

Please note, the JDBC driver for Oracle 10g/11g (Oracle 11.2.0.1.0) is bundled with Bamboo. You do not have to download and install the driver.

- Connecting via JBDC
- Connecting via a datasource

Connecting via JBDC

To connect Bamboo to a Oracle database, via JDBC,
1. Run the Setup Wizard and choose the 'Custom Installation' method.
2. At the 'Choose a Database Configuration' step, choose 'External Database' and select 'Oracle 10g/11g' 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, 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. Run the Setup Wizard and choose the 'Custom Installation' method.
3. At the 'Choose a Database Configuration' step, choose 'External Database' and select 'Oracle 10g/11g' from the list.
4. The 'Select Database Connection' screen will appear. Select 'Connect via a datasource (configured in the application server)'.
5. 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).
6. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click 'Continue' to finish specifying your connection settings.

Screenshot 2: 'Setup Datasource Connection'

Microsoft SQL Server 2005 and 2008

These instructions will help you connect Bamboo to a Microsoft SQL Server 2005 or a Microsoft SQL Server 2008 database. (Microsoft SQL Server 2000 is not a supported database for use with Bamboo.)
SQL Server Express 2005 and 2008 are not recommended databases due to CPU, memory, database size limitations (please see these pages for full details: SQL Server Express 2005 feature comparison, SQL Server Express 2008 feature comparison). However, the instructions below will allow you to connect Bamboo to SQL Server Express 2005/2008.

Please note, the JDBC driver for SQL Server 2005/2008 (JTDS 1.2.2) is bundled with Bamboo. You do not have to download and install the driver.

• 1. Configuring SQL Server
• 2. Creating Your Database
• 3. Connecting Bamboo to SQL Server
  • Connecting via JDBC
  • Connecting via a datasource
• Unicode Characters Not Supported By Default

1. Configuring SQL Server

Before you connect Bamboo to a SQL Server, you need to configure the SQL Server appropriately.

• Change server authentication to 'SQL Server and Windows Authentication mode' — On a typical SQL Server installation, 'Windows Authentication' mode is the default security mode. However, if you try to connect to the database with a database user using this authentication mode, SQL Server will throw an error. You need to change the server authentication mode to 'SQL Server and Windows Authentication mode' in SQL Server before you can connect Bamboo to SQL Server. Please see this MSDN article for instructions on how to do this.

  Screenshot: Changing the SQL Server authentication mode (click to view larger image)

• Configure your firewall to allow SQL Server access — If you need to access SQL server through a firewall, you will need to configure your firewall appropriately. The following MSDN article describes how to configure a Windows firewall to allow SQL Server access, however the instructions are applicable to other firewalls: Configuring the Windows Firewall to Allow SQL Server Access.

• Enable the TCP/IP protocol for your database instance — You must enable the TCP/IP protocol for your SQL Server database instance by following the instructions in this MSDN article.

2. Creating Your Database

After configuring the SQL Server, you need to create the SQL database.

• Create the database for Bamboo — see this MSDN article for instructions.

• Assign the 'db-owner' role on the database for the user that will access the Bamboo database — the 'db_owner' fixed database role allows the user to perform all configuration and maintenance activities on the database. You need to add this role to the Bamboo user used to access your database by updating the login properties for your database user in SQL Server. Read more about login properties for SQL Server.

  Screenshot: Adding the 'db_owner' database role to a database user in SQL Server (click to view larger image)
Please ensure that you use a SQL Server user account to log into your database, not a Windows user account.

- **Configure the database to use case-sensitive collation** — to make the SQL Server database respect case differences in the data it stores (which is required for Bamboo), ensure that you configure it using a case-sensitive collation option such as 'Latin1_General_CS_AS'. To access this feature in SQL Server Management Studio, right-click on the database name, select 'Properties' from the resulting menu, then select the 'Options' page.

  [Screenshot: Configuring the Bamboo database to use 'Latin1_General_CS_AS' collation (click to view larger image)]

- **Configure the database to use the correct isolation level** — Ensure that the new database was set to use Read Committed with Row Versioning as its isolation level. You can apply the new isolation by executing the following query:

  ```sql
  ALTER DATABASE <database name>
  SET READ_COMMITTED_SNAPSHOT ON
  WITH ROLLBACK IMMEDIATE;
  ```

  To verify the changes, use this query which should result in '1':

  ```sql
  SELECT sd.is_read_committed_snapshot_on
  FROM sys.databases AS sd
  WHERE sd.[name] = '<database name>';  
  ```

3. **Connecting Bamboo to SQL Server**

Bamboo provides two ways to connect to a Microsoft SQL Server database — via JDBC or via a datasource. JDBC is generally simpler and is the recommended method.

- If you are planning to support Unicode in Bamboo please enable unicode settings for SQL Server

  **Connecting via JDBC**

  To connect Bamboo to a Microsoft SQL Server database, via JDBC,
1. Run the Setup Wizard and choose the 'Custom Installation' method.
2. On the Choose a Database Configuration page, choose 'External Database', select 'Microsoft SQL Server 2005/2008' from the list and click the 'Continue' button. The Database Configuration page will appear.
3. Ensure that 'Direct JDBC connection' has been chosen and complete the following fields (as shown in the screenshot below):
   - 'Driver Class Name' — Type the following: net.sourceforge.jtds.jdbc.Driver (if different from the default)
   - 'Database URL' — Type the URL where Bamboo will access your database, e.g. jdbc:jtds:sqlserver://localhost:1433/<database>. For details about syntax, please refer to the Microsoft SQL Server documentation.
   - 'Username' — Type the username that Bamboo will use to access your database.
   - 'Password' — Type the password that Bamboo will use to access your database.
4. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
5. Click 'Continue' to finish specifying your connection settings.

Screenshot 1: 'Set Up JDBC Connection SQL Server 2005/2008'

### Database Configuration

Choose how you wish Bamboo to connect to your database

**Select Database Connection**

- Direct JDBC connection
- Connect via a datasource (configured in the application server)

<table>
<thead>
<tr>
<th>Driver Class Name:</th>
<th>net.sourceforge.jtds.jdbc.Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username:</td>
<td></td>
</tr>
<tr>
<td>Password:</td>
<td></td>
</tr>
<tr>
<td>Overwrite existing data</td>
<td></td>
</tr>
</tbody>
</table>

Click 'Continue' to finish specifying your connection settings.

Connecting via a datasource

To connect Bamboo to a MS SQL Server, via a datasource,

1. Configure a datasource in your application server (consult your application server documentation for details).
   - For details about the syntax to use for the SQL Server database URL, please refer to the Microsoft SQL Server documentation.
2. Run the Setup Wizard and choose the 'Custom Installation' method.
3. On the Choose a Database Configuration page, choose 'External Database', select 'Microsoft SQL Server 2005/2008' from the list and click the 'Continue' button. The Database Configuration page will appear.
4. Choose 'Connect via a datasource (configured in the application server)' (as shown in the screenshot below).
5. In the 'JNDI name' field, type the JNDI name of your datasource, as configured in your application server.
   - If java:comp/env/jdbc/DataSourceName does not work, try jdbc/DataSourceName (and vice versa).
6. Select the 'Overwrite existing data' check box if you wish Bamboo to overwrite any tables that already exist in the database.
7. Click 'Continue' to finish specifying your connection settings.

Screenshot 2: 'Setup Datasource Connection'
Unicode Characters Not Supported By Default

Problem
Non-ASCII characters will not be displayed by Bamboo.

Reason
The default SQL Server dialect uses column types that do not support Unicode, specifically the `char`, `varchar` and `text` column types. See CONF-4786 for details.

Solution
To add Unicode support, use the Unicode SQL Server dialect which uses `nchar`, `nvarchar` and `ntext` column types. Unicode SQL Server dialect has the downside of halving the maximum length of each column from 8000 characters to 4000, as every char is stored in two bytes.

Enable Unicode SQL Server dialect on a new setup, perform these steps prior to 'Step 3 - Database Connection Setup'.

1. Open the `<bamboo-install>/webapp/WEB-INF/classes/database-defaults/mssql.properties` file within your Bamboo installation folder.
2. Comment the line: `dialect=net.sf.hibernate.dialect.SQLServerDialect`
3. Uncomment the line: `#dialect=net.sf.hibernate.dialect.SQLServerIntlDialect`
4. Start the Bamboo Setup Wizard

How do I connect Bamboo to an unsupported database?

We strongly recommend that you use Bamboo with one of the databases that we support (see Supported Platforms for details). However, if you wish to connect Bamboo to an unsupported database, you can do so via the instructions below.

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 into your application server’s classpath:
   - For the Bamboo Standalone distribution, copy the 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. The Setup Wizard no longer offers "Unsupported Database" as a selectable option when choosing to connect to an external database. If you are using an unsupported database, you will need to set the following system property before starting your upgraded Bamboo server to enable "Unsupported Database" as a selectable option in the Setup Wizard:
   ```sh
   -Dbamboo.enable.unsupported.db=true
   ```
3. At Step 2 of the Bamboo Setup Wizard, choose ‘External Database’ and select ‘Unsupported Database’ from the list.
4. The ‘Select Database Connection’ screen will appear. Select ‘Direct JDBC connection’.
5. 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.
     ```
     | Database                  | Dialect                                      |
     |---------------------------|----------------------------------------------|
     | DB2                       | net.sf.hibernate.dialect.DB2Dialect          |
     | DB2 AS/400                 | net.sf.hibernate.dialect.DB2400Dialect       |
     | DB2 OS390                  | net.sf.hibernate.dialect.DB2390Dialect       |
     | Oracle 9/10g               | net.sf.hibernate.dialect.Oracle9Dialect      |
     | Oracle (other versions)    | net.sf.hibernate.dialect.OracleDialect       |
     | Sybase                     | net.sf.hibernate.dialect.SybaseDialect       |
     | Sybase Anywhere            | net.sf.hibernate.dialect.SybaseAnywhereDialect |
     | Microsoft SQL Server      | net.sf.hibernate.dialect.SQLServerDialect    |
     | SAP DB                     | net.sf.hibernate.dialect.SAPDBDialect        |
     | Informix                   | net.sf.hibernate.dialect.InformixDialect     |
     | Ingres                     | net.sf.hibernate.dialect.IngresDialect       |
     | Progress                   | net.sf.hibernate.dialect.ProgressDialect     |
     | Mckoi SQL                  | net.sf.hibernate.dialect.MckoiDialect        |
     | Interbase                  | net.sf.hibernate.dialect.InterbaseDialect    |
     | Pointbase                  | net.sf.hibernate.dialect.PointbaseDialect    |
     | FrontBase                  | net.sf.hibernate.dialect.FrontbaseDialect    |
     | Firebird                   | net.sf.hibernate.dialect.FirebirdDialect     |
   - ‘Overwrite existing data’ checkbox if you wish Bamboo to overwrite any tables that already exist in the database.
7. Go to Step 3 of the Setup Wizard.

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. By default, the Setup Wizard does not offer Unsupported Database as a selectable option when choosing to connect to an external database. If you are using an unsupported database (see Supported Platforms for more information), you will need to set the following system property before starting your upgraded Bamboo server to enable the Unsupported Database as a selectable option in the Setup Wizard:

   -Dbamboo.enable.unsupported.db=true

3. At Step 2 of the Bamboo Setup Wizard, choose External Database and select Unsupported Database from the list.

4. The 'Select Database Connection' screen will appear. Select 'Connect via a datasource (configured in the application server)'.

5. 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.

   ![Screenshot of Setup Datasource Connection]

   ![Warning: If java:comp/env/jdbc/DataSourceName doesn't work, try jdbc/DataSourceName (and vice versa).]

6. Select the 'Overwrite existing data' checkbox if you wish Bamboo to overwrite any tables that already exist in the database.

7. Go to Step 3 of the Setup Wizard.

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.

Bamboo Release Notes

Latest Version

Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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

Page: Bamboo 3.1 Release Notes
Page: Bamboo 3.0.3 Release Notes
Page: Bamboo 3.0.2 Release Notes
Page: Bamboo 3.0.1 Release Notes
Page: Bamboo 2.7.4 Release Notes
Page: Bamboo 3.0 Release Notes
Page: Bamboo 2.7.3 Release Notes
Page: Bamboo 2.7.1 Release Notes
Page: Bamboo 2.7.2 Release Notes
Page: Bamboo 2.6.3 Release Notes
Bamboo Release Summary

This page shows the highlights of the major Bamboo releases.

Current Release

For information about the latest release of Bamboo, please check the main Bamboo Release Notes page.

Bamboo 3.1 — 10 May 2011

- Tasks
- Parameterised Builds
- .Net Support
- Bitbucket Support
- GitHub Support
- New Plugin Manager
- Support for Amazon EC2 Spot Instances
- Gravatar Support
- Improved Windows process handling
- More in release notes

Bamboo 3.0 — 16 February 2011

- Artifact Sharing
- Git Support
- User Interface Overhaul — Redesigned Plan Summary, Job Summary and Build Results. New look and feel.
- Scheduled Repository Polling
- Configuration Changes Captured in Audit Logs
- More in release notes

Bamboo 2.7 — 9 Nov 2010

- Build Stages — Map Your Build Process, Parallel Builds, Enhanced Plan Structure
- Simplified Plan Creation
- Concurrent Builds
- Mercurial Support
- Improved Wallboards
- New Plan and Job Configuration Summaries
- Recent History on Plan and Job Summaries
- Other User Interface Enhancements — New Breadcrumb Trail, Build Histories, Improved Build Result Summary Tabs
- More in release notes

Bamboo 2.6 — 1 June 2010

- Support for up to 100 Remote Agents
- Revamped Dashboard Pages and Other Usability Enhancements
- Performance and Security Improvements
- Automatically Managed Elastic Instances
- Grails Integration
- More in release notes

Bamboo 2.5 — 4 January 2010

- Maven Dependency Management
Bamboo 3.1 Documentation

- Plan Import from a pom.xml
- Additional Bulk Actions
- Streamlined Plan Creation
- Express Setup Wizard
- More in 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
Bamboo 3.1 Documentation

• Dashboard loading has been improved
• More in release notes

Bamboo 1.0 — 20 February 2007

• All Plans tab
• More in release notes

Bamboo 3.1 Release Notes

10 May 2011

Atlassian presents Bamboo 3.1 with Tasks, parameterised builds and support for Bitbucket and GitHub.

Upgrading to Bamboo 3.1 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

• Tasks
• Plan Variables & Parameterised Builds
• .Net Support
• Bitbucket Support
• GitHub Support
• New Plugin Manager
• Support for Amazon EC2 Spot Instances
• Gravatar Support
• Improved Windows process handling
• Plus over 150 fixes and improvements
• The Bamboo 3.1 Team

Thank you for your feedback:

🌟 47 new features and improvements implemented
🌟 114 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!

<table>
<thead>
<tr>
<th>Upgrading to Bamboo 3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download Bamboo from the <a href="https://www.atlassian.com">Atlassian website</a>. If upgrading from a previous version, please read the Bamboo 3.1 Upgrade Guide.</td>
</tr>
</tbody>
</table>

Tasks

Tasks in Bamboo 3.1 provide developers and build engineers with another tool to design more flexible builds. Configure a Job with Tasks to build your application, execute a script, upload files to another server, create your documentation and much more. Bamboo allows you to add as many Tasks as needed for a Job, with each Task providing detailed log messages during the build. Tasks are executed against the same working directory, allowing you to perform actions like changing version numbers or copying files before a subsequent task is executed.

We've also implemented Final Tasks as part of the Tasks feature. No matter what happens in your previous tasks, Final Tasks will always be executed at the end of the build. This gives you the opportunity to clean up after your build, shutting down processes or services that you may have started as part of your build.
Plan Variables & Parameterised Builds

Parameterised Builds allow you to customise parts of your Build when the Build is run manually or via a script. We've introduced Plan variables in this release to complement the existing global variables. This allows you to change version numbers on the fly or change certain options you use within your script or commands for particular builds.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentStableVersion</td>
<td>3.0.2</td>
</tr>
<tr>
<td>finalReleaseVersion</td>
<td>3.1</td>
</tr>
</tbody>
</table>
.Net Support

We've completely re-written the .Net plugin for Bamboo to add Tasks for building and testing .Net projects. All Tasks take advantage of our improved Windows process handling. The following Tasks are included in the plugin:

- Visual Studio – Build Visual Studio projects with devenv.exe. The Task also allows you to switch between different architectures (x86, AMD64, IA32, IA64).
- MSBuild – Run MSBuild as part of your build.
- NAnt – Execute NAnt targets to build your project.
- MSTest Runner – Run your MSTest configuration and display the MSTest results.
- MSTest Parser – Parse and display MSTest test results.
- MBUnit Parser – Parse and display MBUnit test results.
- NUnit Parser – Parse and display NUnit test results.

The plugin is open-source. Feel free to fork it on Bitbucket.

More...

Bitbucket Support

Bamboo now supports Bitbucket. If you use Bitbucket for your source code hosting, you can use Bamboo to build any source code maintained in repositories on Bitbucket.

GitHub Support

We've extended our Git support to include GitHub. If you use the GitHub for your source code hosting, you can use Bamboo to build any source code maintained in repositories on GitHub.
New Plugin Manager

Managing plugins and performing Bamboo upgrades are now much easier with the brand new plugin manager. The Universal Plugin Manager (UPM) is already bundled with JIRA and Confluence, and has now been integrated into Bamboo. With the UPM you can:

- Perform a plugin compatibility check before upgrading Bamboo.
- Install new plugins from the Atlassian Plugin Exchange.
- Manage existing plugins.
- With just one click, upgrade all plugins that have updates available.
- View and track updates via the audit log.
Support for Amazon EC2 Spot Instances

Bamboo now supports Amazon EC2 Spot Instances. If you are using Elastic Bamboo to run builds in the Amazon Elastic Compute Cloud (EC2), you can now bid for and use EC Spot Instances. This allows you to run your builds at a much lower price, provided that your bid exceeds the current "spot price" (determined by EC2 customer demand).
Gravatar Support

Bamboo 3.1 adds a personal touch to your CI environment with the introduction of Gravatar support. If your users have signed up to the Gravatar service, Bamboo will attempt to retrieve their profile pictures and display them in Bamboo. You’ll see these profile pictures displayed against activities for the user, like code changes and comments on build results.

Improved Windows process handling

In previous versions of Bamboo, processes started from .bat scripts or a number of different methods were not shut down properly. We have improved the Windows process handling in Bamboo to ensure that the underlying processes and their children are stopped correctly.
Plus over 150 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

### JIRA Issues (10 issues)

<table>
<thead>
<tr>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Votes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-1410</td>
<td>Multiple builders per plan</td>
<td>33</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-811</td>
<td>On Windows, builds using a batch script or other method, can not be stopped</td>
<td>29</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2357</td>
<td>Link to artifacts from the latest successful build</td>
<td>12</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-2960</td>
<td>Stop build script feature - Something to run when a build is cancelled / stopped</td>
<td>6</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-1854</td>
<td>Multiple builders for a particular build configuration</td>
<td>4</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-5214</td>
<td>Allow Build REST API to receive custom parameters</td>
<td>4</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-8241</td>
<td>Make ImportExportManager available to plugins</td>
<td>4</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-5127</td>
<td>Improve the Bamboo startup script</td>
<td>3</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-6177</td>
<td>Allow upload of key and certificate for EBS</td>
<td>2</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td></td>
<td>BAM-1934</td>
<td>Externalize all configuration, plugins, and properties from the bamboo installation</td>
<td>2</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>

The Bamboo 3.1 Team

**Development**

**Core Team**

- Brydie McCoy
- James Dumay
- Jason Berry
- Marek Went
- Krystian Brazulewicz
- Przemek Bruski
- Marcin Gardias
- Piotr Stefan Stefaniak
- Ben Woskow

**Git**

- Slawek Ginter

**Team Lead**

- Mark Chaimungkalanont

**Project Manager**

- Anton Mazkovoi

**Support**

- Renan Battaglin
- Ajay Sridhar
- Zed Yap
- Gurleen Anand
- Felipe Kraemer
- Rene Verschoor
- Dylan Hansen
Bamboo 3.1 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.1 from a previous version of Bamboo. For details on the Bamboo 3.1 release, see the Bamboo 3.1 Release Notes.

Please follow the Bamboo 3.1-specific instructions on this page, in addition to the upgrade instructions in the Bamboo Generic Upgrade Guide.

Please read the Supported Platforms page for the full list of supported platforms for Bamboo.

On this page:
- Upgrade Notes
- Upgrading from Bamboo 3.0 to 3.1
- Upgrading from Bamboo prior to 3.0
- Developing for Bamboo 3.1
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

Upgrade Notes

Bamboo Compatibility with Subversion pre-1.6

We have changed the default behaviour of the SVNKit library in Bamboo. As a result, any source code checked out by Bamboo will be automatically upgraded to be compatible with Subversion 1.6. This does not adversely affect any pre-1.6 Subversion servers. However, if you use a pre-1.6 Subversion client to access code checked out by Bamboo, then any Bamboo builds on that code may fail.

If you want to prevent any checked out code from being automatically upgraded to SVN client format 1.6, you will need to run Bamboo with the following system property:

```
-Dbamboo.svn.wc.format=1.5
```

For more information, please see this FAQ: How do I manually set the version of new Subversion workspaces?.

End of Support for Java Platform 5 (JDK/JRE 1.5)

We are ending support for Java Platform 5 (JDK/JRE 1.5) in this release. Please see End of Support Announcements for Bamboo for further details.

Conversion of Builders to Tasks

The introduction of the Tasks feature in Bamboo 3.1 means that the following activities will occur during the upgrade to Bamboo 3.1:

- Builder capabilities will be renamed to Executable capabilities.
- Builders will be converted to Tasks. The Tasks will be linked to the Job that the Builders were a part of.
- If one of your Builders cannot be matched to a Task (e.g. you are using a custom plugin), it will be converted to a ‘Compatibility Task’. The configuration for your Builder will be transferred to this Task, and the Task will be linked to the Job that the Builder was a part of. You can view/update the configuration by navigating to the Task and clicking ‘Configure Legacy Executable’.

Changes to Bamboo Files/Directories for Windows Standalone Installations

If you are using the Windows Standalone Installation of Bamboo, please note that the location of the following files/directories have changed.

- All log files now located at %USERPROFILE%\bamboo.log, rather than in the logs folder of your installation directory. For Bamboo
running as a Windows service, log files are located at %WINDIR%\System32\Config\systemprofile\bamboo.log. Note, existing logs will not be migrated, however new logs will be written to the new location when running Bamboo after the upgrade.

- All temporary directories on windows are now by default in %WINDIR%\Temp, rather than in the user's temporary directory.

Gravatar Support Enabled by Default

The new Gravatar support feature is enabled by default in Bamboo 3.1. For more information, see Configuring Gravatar Support.

Upgrading from Bamboo 3.0 to 3.1

Before you begin, do the following:

1. Back up your existing installation of Bamboo

We strongly recommend that you do the following to back up your Bamboo installation:

- Back up your xml-data directory — See the Bamboo Generic Upgrade Guide for instructions.
- Export your Bamboo data for backup — See the Exporting Data for Backup for instructions. Please note, that this may take a long time to complete depending on the number of builds and tests in your system.

2. Ensure that your plugins work

If you are using plugins, ensure that your plugins are compiled against 3.1 before upgrading.

Before you upgrade, please read the following important points that relate to Bamboo 3.0.

Upgrading from Bamboo prior to 3.0

In addition to the notes below, please read the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Notes for upgrading from Bamboo 2.6.x

- You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.0. If you are not running Bamboo 2.6.3, we recommend that you upgrade to it before upgrading to Bamboo 2.7.4. Bamboo 2.6.3 can be downloaded from the Bamboo Archived Downloads page. Bamboo 2.7.x introduces a number of significant and irreversible changes, so a phased upgrade is recommended. Please see the Bamboo 2.7.x Upgrade Guide for more details.
- You will need to set aside time, as described in the Bamboo 2.7.x Upgrade Guide, for Bamboo to migrate existing Plans to the new Plan structure in Bamboo 2.7.4.

Notes for upgrading from Bamboo 2.5 or earlier

- If you are upgrading from Bamboo 2.5 or earlier, you will need to set aside time, as described in the Bamboo 2.6 Upgrade Guide for Bamboo to migrate its test result data (stored in XML files on the filesystem) into the database.

Notes for upgrading from a version of Bamboo prior to 2.0

- If you are upgrading from a version of Bamboo prior to 2.0, you must upgrade to Bamboo 2.0.6 first before upgrading to Bamboo 2.6. Please read the Bamboo 2.0 Upgrade Guide for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.1

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.1 guide, which outlines changes in Bamboo 3.1 that may affect Bamboo plugins compiled for Bamboo version 3.0.x or earlier.

Checking for Known Issues and Troubleshooting the Bamboo Upgrade

If something is not working correctly after you have completed the steps above to upgrade your Bamboo installation, please check for known Bamboo issues and try troubleshooting your upgrade as described below:

- Check for known issues. Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases we publish information about the known issues in the Bamboo Knowledge Base. Please check the Bamboo 3.1 Known Issues in the Bamboo Knowledge Base and follow the instructions to apply any necessary patches if necessary.
- Did you encounter a problem during the Bamboo upgrade? Please refer to the guide to troubleshooting upgrades in the Bamboo Knowledge Base.
- If you encounter a problem during the upgrade and cannot solve it, please create a support ticket and one of our support engineers will help you.

Bamboo 3.0 Release Notes

16 February 2011
With great pleasure, Atlassian presents **Bamboo 3.0** with artifact sharing, Git support and a revamped user interface.

Upgrading to Bamboo 3.0 is free for all customers with active Bamboo software maintenance.

**Highlights of this release:**

- Artifact Sharing
- Git Support
- User Interface Overhaul
- Scheduled Repository Polling
- Configuration Changes Captured in Audit Logs
- Plus over 400 fixes and improvements
- The Bamboo 3 Team

**Thank you for your feedback:**

🌟 40 new features and improvements implemented  
🌟 125 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!

---

### Upgrading to Bamboo 3.0

You can download Bamboo from the [Atlassian website](https://www.atlassian.com). If upgrading from a previous version, please read the **Bamboo 3.0 Upgrade Guide**.

---

### Artifact Sharing

Bamboo 3.0 allows artifacts produced from a Job to be shared with other Jobs in the same Plan, without being rebuilt every time. Build your artifacts in the first Stage and pass them through Unit and Acceptance testing Stages. When the build has completed, you will have every confidence that the final artifact has been thoroughly tested, works and is ready for further deployment.

Artifact sharing for Maven 2 is also supported, but in beta.
Git Support

Bamboo now supports Git. If you use this distributed version control system (DVCS) or are thinking of migrating to it, you can use Bamboo to build any source code maintained in Git repositories.
User Interface Overhaul

In our previous release, Bamboo 2.7, we introduced Stages and Jobs to improve the way your Builds are structured. The user interface wasn’t ideal for representing the new Plan structure though. In this release, we’ve taken the opportunity to completely overhaul the Bamboo user interface, including the Plans, Jobs and Build Results screens. If you are currently using Atlassian’s JIRA, you’ll feel right at home with the new look and feel!

Highlights include:

- A more neutral colour scheme to improve readability of the screens.
- Redesigned controls — slicker tabs, svelte forms and tables, Atlassian-standard headings and better buttons.
- Layout changes — functions moved into dropdowns to make key information more prominent.

Plan and Job Summary

As part of our drive to improve the Bamboo user experience, we’ve implemented better user interfaces for Plans and Jobs. The new Plan Navigator shows you the Stages and Jobs hierarchy in a Plan, as well as allowing quick navigation to Jobs. If you are looking at a Job, it will be highlighted in the Plan Navigator. Common functions have been moved into an ‘Actions’ menu for easy access. The interface also looks much cleaner, due to a better organised layout and the new colour scheme.

Build Results

The Build Results user interface for Plans and Jobs has also been improved in Bamboo 3.0. This includes a status ribbon that allows you to see whether a build was successful or not, at a glance. We’ve also added a history navigator that allows you to view the status of and navigate to, prior and subsequent build results. An updated layout and the new colour scheme complements these new features.
Scheduled Repository Polling

Bamboo now allows you to schedule when you want to poll your source repositories for changes. You can create a schedule using Cron expressions, but don’t worry if you can’t remember all that Cron lingo. Bamboo has an easy-to-use user interface that allows you to create your schedule without any expression magic required.

Configuration Changes Captured in Audit Logs
All configuration changes in Bamboo are now recorded in the audit logs. This allows you to track down whether a build failed because of an actual problem in your code, or whether a Plan configuration change was responsible.

6

Plus over 400 fixes and improvements

The top 10 issues by votes are shown below. For the full list of fixes and improvements, please refer to our public JIRA site to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Votes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2875</td>
<td></td>
<td>GIT support for BAMBOO</td>
<td></td>
<td>62</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1704</td>
<td></td>
<td>New Build Strategy: Polling the repository at a fixed time</td>
<td></td>
<td>22</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-2496</td>
<td></td>
<td>Conditional, cron-based scheduling</td>
<td></td>
<td>11</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1104</td>
<td></td>
<td>You should be allowed to edit build artifacts</td>
<td></td>
<td>7</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7218</td>
<td></td>
<td>Maven 3.0 or 3.x Builder</td>
<td></td>
<td>3</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7453</td>
<td></td>
<td>BuildState is not set before POST plugins</td>
<td></td>
<td>2</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7722</td>
<td></td>
<td>Bamboo fails to auto detect mercurial executable on remote agent.</td>
<td></td>
<td>2</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7717</td>
<td></td>
<td>Support self signed SSL certificates when accessing Git repositories</td>
<td></td>
<td>2</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7038</td>
<td></td>
<td>Maven 2 build processor fails to find parent projects</td>
<td></td>
<td>1</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-7771</td>
<td></td>
<td>Build plan failed to run - it was marked as queued but was not present in the queue</td>
<td></td>
<td>1</td>
<td>Resolved</td>
<td></td>
</tr>
</tbody>
</table>

The Bamboo 3 Team

Development

Core Team

Brydie McCoy
James Dumay
Jason Berry
Marek Went
Krystian Brazulewicz
Przemek Bruski
Marcin Gardias
Michael Truong

Git

Slawek Ginter
Piotr Stefan Stefaniak

Team Lead

Mark Chaimungkalanont

Project Manager

Anton Mazkovoi

Support

Renan Battaglin
Ajay Sridhar
Zed Yap

Others

Product Management
Jens Schumacher
Helen Hung

Product Marketing
Giancarlo Lionetti

Technical Writing
Andrew Lui

Operations
James Fleming

Bamboo 3.0.1 Release Notes

25 February 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.0.1.

We’ve fixed several bugs in this release. Please see the ‘Updates and Fixes in this Release’ section below for details.

Bamboo 3.0.1 is of course free to all customers with active Bamboo software maintenance.

Don’t have Bamboo 3.0 yet?

Take a look at all the new features in the Bamboo 3.0 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (4 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-8048</td>
</tr>
</tbody>
</table>

Bamboo 3.0.1 Upgrade Guide

Upgrading from Bamboo 3.0 to 3.0.1

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.0 to 3.0.1.

Upgrading from Bamboo 3.0 or earlier

In addition to the above, please read the Bamboo 3.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 3.0.2 Release Notes
The Atlassian Bamboo team is proud to announce the release of Bamboo 3.0.2.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.0.2 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.0 yet?

Take a look at all the new features in the Bamboo 3.0 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues</th>
<th>Issue 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-8231</td>
<td>Upgrade SVNKit to 1.3.5.7539 to fix invalid xml character and svn externals errors</td>
<td>James Dumay [Atlassian]</td>
<td>Renan Battaglin [Atlassian]</td>
<td>[Resolved]</td>
<td>Fixed</td>
<td>Mar 09, 2011</td>
<td>Mar 24, 2011</td>
<td></td>
</tr>
</tbody>
</table>

Bamboo 3.0.2 Upgrade Guide

Upgrading from Bamboo 3.0 to 3.0.2

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 3.0 to 3.0.2.

Upgrading from Bamboo 3.0 or earlier

In addition to the above, please read the Bamboo 3.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 3.0.3 Release Notes

18 April 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 3.0.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 3.0.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 3.0 yet?

Take a look at all the new features in the Bamboo 3.0 Release Notes and see what you are missing out on!
Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 3.0.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 3.0.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

### JIRA Issues (6 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-8216</td>
<td>Build hung never finish in the UI even after killing agent and all the related process</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Adrian Deccico [Atlassian]</td>
<td>⚡️ ⚡️</td>
<td>⚡️</td>
<td>Resolved</td>
<td>Mar 08, 2011</td>
<td>Apr 08, 2011</td>
</tr>
<tr>
<td></td>
<td>BAM-7833</td>
<td>Users can't reset passwords if the Global Anonymous Permission is disabled</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Renan Battaglin [Atlassian]</td>
<td>⚡️ ⚡️</td>
<td>⚡️</td>
<td>Resolved</td>
<td>Jan 26, 2011</td>
<td>Apr 20, 2011</td>
</tr>
<tr>
<td></td>
<td>BAM-8153</td>
<td>Bamboo does not remove temporary files if failure occurs during artifact transfer or deserialisation</td>
<td>Marcin Gardias [Atlassian]</td>
<td>Przemek Bruski [Atlassian]</td>
<td>⚡️ ⚡️</td>
<td>⚡️</td>
<td>Resolved</td>
<td>Mar 03, 2011</td>
<td>Apr 21, 2011</td>
</tr>
</tbody>
</table>

Bamboo 3.0.3 Upgrade Guide

**Upgrading from Bamboo 3.0.2 to 3.0.3**

Please follow the Bamboo Generic Upgrade Guide.

⚠️ No additional upgrade tasks are required to upgrade from Bamboo 3.0.2 to 3.0.3.

**Upgrading from Bamboo 3.0.1 or earlier**

In addition to the above, please read the Bamboo 3.0 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 3.0 Upgrade Guide

The instructions on this page describe how to upgrade to Bamboo 3.0 from a previous version of Bamboo. For details on the Bamboo 3.0 release, see the Bamboo 3.0 Release Notes.

Please following the Bamboo 3.0-specific instructions on this page, in addition to the upgrade instructions in the Bamboo Generic Upgrade Guide.

Please read the Supported Platforms page for the full list of supported platforms for Bamboo.

---

**On this page:**

- Upgrade Notes
- Upgrading from Bamboo 2.7 to 3.0
- Upgrading from Bamboo prior to 2.7
- Developing for Bamboo 3.0
- Checking for Known Issues and Troubleshooting the Bamboo Upgrade

**Upgrade Notes**

End of Support for Internet Explorer 7
We are ending support for Internet Explorer 7 (IE7) in this release. Please see End of Support Announcements for Bamboo for further details.

Advance Notice of End of Support for Java Platform 5

We are planning on ending support for Java Platform 5 (JDK/JRE 1.5) in Bamboo 3.1. Please see End of Support Announcements for Bamboo for further details.

All Bamboo versions using MS SQL 2005 and 2008 demand Read Committed with Row Versioning isolation level.

- Before starting the upgrade process ensure that your current Bamboo MS SQL database is set to use Read Committed with Row Versioning as its isolation level. If you are planning to restore a Bamboo Backup Zip, ensure that the new database will have this isolation level as well. For instructions on how to set this isolation level, please see this page.

Specifying Artifact Location

In Bamboo 2.7 and earlier, artifacts are stored under xml-data/builds under your ${bambooHome} (unless specified otherwise). An upgrade task for Bamboo 3.0 will move your artifacts out of ${bamboo.project.directory} into a separate artifacts directory under ${bambooHome}. If your artifacts are currently not located under your ${bambooHome}, i.e. you manually changed the location of your artifacts, you will need to do one of the following:

- make sure that there is enough room under ${bambooHome} to accommodate the artifacts in the new artifacts directory, or
- set the bamboo.artifacts.directory property (in bamboo.cfg.xml) to the preferred location for your artifacts. You must update this property before the upgrade. The upgrade task will use the location specified by this property, rather than moving artifacts to the new artifacts directory under ${bambooHome}.

If your ${bamboo.project.directory} currently points to a different physical disk to your ${bambooHome}, the upgrade process will copy (rather than move) data between locations, unless you set the bamboo.artifacts.directory property.

The new Default Path for Artifacts

- Bamboo 2.6 and earlier versions:
  xml-data/builds/PLAN_KEY/download-data

- Bamboo 2.7:
  <bamboo-home>/xml-data/builds/JOB_KEY/download-data/artifacts/build-BUILD_NUMBER

- Bamboo 3.0:
  <bamboo-home>/artifacts/PLAN_KEY/shared/build-BUILD_NUMBER/

In Bamboo 3.0, this is a folder shared by all the stages of a certain plan. Stages will place Artifacts here so that other stages from the same plan can have access to them. The BUILD_NUMBER will always be composed with a minimum of 5 digits, having the number completed with zeros. For instance, for build “42” the number will be “00042”.

Upgrade Exceptions

- If you experience the following exception during the Bamboo 3.0 upgrade, it means that the upgrade task has failed to fully migrate a directory, as part of the internal artifact storage migration. You will need to manually move the directory and restart the upgrade.

  Unable to move DIRECTORY_NAME_A -> DIRECTORY_NAME_B, destination directory already exists. This might indicate interrupted upgrade process. To continue upgrade, move directory manually.

Crowd Integration Authenticator

Bamboo 3.0 is using the new 2.4 version of the Seraph authenticator. Please, go through the Integrating Crowd with Atlassian Bamboo steps to ensure that the new necessary configurations will be applied.

Upgrading from Bamboo 2.7 to 3.0

Before you begin, do the following:

1. Back up your existing installation of Bamboo

We strongly recommend that you do the following to back up your Bamboo installation:

- Back up your xml-data directory — See the Bamboo Generic Upgrade Guide for instructions.
• Export your Bamboo data for backup — See the Exporting Data for Backup for instructions. Please note, that this may take a long time to complete depending on the number of builds and tests in your system.

2. Ensure that your plugins work

If you are using plugins, ensure that your plugins are compiled against 3.0 before upgrading.

Before you upgrade, please read the following important points that relate to Bamboo 2.7.

Upgrading from Bamboo prior to 2.7

In addition to the notes below, please read the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available here.

Notes for upgrading from Bamboo 2.6.x

• You will need to upgrade to Bamboo 2.7.4 before upgrading to Bamboo 3.0. If you are not running Bamboo 2.6.3, we recommend that you upgrade to it before upgrading to Bamboo 2.7.4. Bamboo 2.6.3 can be downloaded from the Bamboo Archived Downloads page. Bamboo 2.7.x introduces a number of significant and irreversible changes, so a phased upgrade is recommended. Please see the Bamboo 2.7.x Upgrade Guide for more details.
• You will need to set aside time, as described in the Bamboo 2.7.x Upgrade Guide, for Bamboo to migrate existing Plans to the new Plan structure in Bamboo 2.7.4.

Notes for upgrading from Bamboo 2.5 or earlier

• If you are upgrading from Bamboo 2.5 or earlier, you will need to set aside time, as described in the Bamboo 2.6 Upgrade Guide for Bamboo to migrate its test result data (stored in XML files on the filesystem) into the database.

Notes for upgrading from a version of Bamboo prior to 2.0

• If you are upgrading from a version of Bamboo prior to 2.0, you must upgrade to Bamboo 2.0.6 first before upgrading to Bamboo 2.6. Please read the Bamboo 2.0 Upgrade Guide for important upgrade instructions for upgrading from earlier versions of Bamboo.

Developing for Bamboo 3.0

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 3.0 guide, which outlines changes in Bamboo 3.0 that may affect Bamboo plugins compiled for Bamboo version 2.7.x or earlier. In particular, please note that the /build REST endpoint has been replaced with /result. Expand parameters have also been changed from builds.build to results.result.

Checking for Known Issues and Troubleshooting the Bamboo Upgrade

If something is not working correctly after you have completed the steps above to upgrade your Bamboo installation, please check for known Bamboo issues and try troubleshooting your upgrade as described below:

• Check for known issues. Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases we publish information about the known issues in the Bamboo Knowledge Base. Please check the Bamboo 3.0 Known Issues in the Bamboo Knowledge Base and follow the instructions to apply any necessary patches if necessary.
• Did you encounter a problem during the Bamboo upgrade? Please refer to the guide to troubleshooting upgrades in the Bamboo Knowledge Base.
• If you encounter a problem during the upgrade and cannot solve it, please create a support ticket and one of our support engineers will help you.

Bamboo 2.7 Release Notes

9 November 2010

The Atlassian Bamboo team is proud to release Bamboo 2.7.

In Bamboo 2.7, we’ve enhanced Plans so that you can map a complete build process into consecutive steps (such as compilation, testing and deployment), all within a single Plan!

Bamboo’s Concurrent Builds feature allows you to execute a single Plan concurrently on multiple agents — extremely useful if the trigger for building a Plan fires more frequently than the time it takes to build the Plan.

Do you use a distributed version control system (DVCS) or are thinking of migrating to one? Bamboo now supports Mercurial, so that you can take full advantage of this popular DVCS.

Upgrading to Bamboo 2.7 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

• Build Stages
  • Map Your Build Process
  • Parallel Builds
Bamboo 3.1 Documentation

- Enhanced Plan Structure
- Simplified Plan Creation
- Concurrent Builds
- Mercurial Support
- Improved Wallboards
- New Plan and Job Configuration Summaries
- Recent History on Plan and Job Summaries
- Other User Interface Enhancements
  - New Breadcrumb Trail
  - Build Histories
  - Improved Build Result Summary Tabs
- Plus over 130 fixes and improvements

Thank you for your feedback:

🌟 20 new features and improvements implemented
🌟 48 votes fulfilled
🌟 Over 200 issues resolved

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!

<table>
<thead>
<tr>
<th>Upgrading to Bamboo 2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 2.7 Upgrade Guide.</td>
</tr>
</tbody>
</table>

Build Stages

Map Your Build Process

Bamboo 2.7 allows you to define and map a complete build process in a single Plan. Build steps like compile, test and deploy are mapped to Stages in your Plan, where Stages are processed sequentially. Builds can fail fast if something breaks early in the build process, saving you valuable processing time!

Parallel Builds

Single build units within a Stage, called Jobs, can be executed in parallel. This enables you to run different suites of tests simultaneously or the same test against different environments, dramatically reducing the feedback cycle within your build process.

Enhanced Plan Structure

To accommodate the features above, Bamboo's Plans now consist of Stages and Jobs, where one or more Jobs can be grouped into a Stage, as depicted in the Enhanced Plan Structure diagram below.

When Bamboo builds a Plan, it starts building all of the Jobs in its first Stage, followed by all Jobs in the second stage and so on.

- Jobs belonging to a single Stage are built in parallel, depending on the availability of Bamboo agents, but Stages are processed one at a time.
- Within any Stage, all Jobs must be built and have succeeded before Bamboo processes the next Stage. If any Job in a Stage fails, Bamboo will not process any subsequent Stages in that Plan (nor any Jobs within these Stages).

Screenshot: Enhanced Plan Structure
Rest assured that when you upgrade to Bamboo 2.7, your existing Plans will be migrated smoothly into Bamboo 2.7's enhanced Plan structure. For more information please refer to the Bamboo 2.7 Upgrade Guide.

Simplified Plan Creation

Bamboo’s Plan creation features have been simplified. Decide up front how you want to create your new Plan:

- Create a new Plan from scratch
- Clone an existing Plan
- Create a Plan by importing a pom.xml file from a Maven 2 project

The Create a New Plan page is now much easier to use and only shows options that are essential for the Plan to start building its first Job. More configuration options such as those for are available when you edit the configuration of your Plan and/or the Plan’s Jobs.

Screenshot: Creating a New Plan from Scratch
Concurrent Builds

Bamboo’s **Concurrent Builds** feature allows you to execute a single Plan concurrently on multiple agents. This is extremely useful if the trigger for building a Plan fires more frequently than the time it takes to complete building that Plan.

*Screenshot: Plans Building Concurrently*
You can easily configure the number of builds of a Plan that your Bamboo server can execute concurrently through Bamboo's administration console. To avoid overloading your Bamboo agents, it is recommended this number be kept to a minimum as the number of Jobs in your Plans increases.

### Mercurial Support

Bamboo now supports Mercurial. If you use this distributed version control system (DVCS) or are thinking of migrating to it, you can use Bamboo to build any source code maintained in Mercurial repositories.

### Improved Wallboards

The wallboard (formerly known as the build monitor) is designed to present your Bamboo server's latest build results on a whole screen and now has the following improvements:

- More Plans can be shown — If you display the wallboard within a browser window, more or fewer build results will be shown upon re-sizing the window.
- More information from a build result — If your wallboard is displayed on a touchscreen (such as an iPad) or its content can be accessed with a mouse, then touching or clicking a build result on the wallboard will show more information about that build.
- Black background — Save more energy if your wallboard-dedicated monitor is a plasma or relatively recent LED-based screen.

If you've connected Bamboo to JIRA with OAuth and are using the JIRA Wallboard, you can display Bamboo gadgets on a JIRA Wallboard along with other JIRA and GreenHopper Gadgets, GreenHopper burndown charts, Crucible code reviews and more!

Screenshot: The Bamboo Wallboard
New Plan and Job Configuration Summaries

A summary of your Plan's or Job's configuration is available on a single page so that you no longer have to click through a series of tabs to view key settings for any given Plan or Job.

To configure a particular section of a Plan or Job, simply click the 'Plan Configuration' or 'Job Configuration' drop-down menus and select the appropriate option.

Screenshot: Configuration Summary for a Plan
Recent History on Plan and Job Summaries

A list of recently built Plans or Jobs is available on their respective **Plan Summary** and **Job Summary** pages.

Each Plan's build has its own build number, where each build number is preceded by a '#' symbol. Each Job that was built as part of a Plan's build, shares the same build number as its Plan's build number.

**Screenshot: Recent History on the Plan Summary Tab**

In the **Recent History** section, clicking on a Plan's build number link, its 'updated' link and test (right-most) link, takes you to the **Summary**, **Changes** and **Tests** tabs respectively for that particular Plan's build.
From the Plan Summary tab, you can quickly access the Plan's Job Summary page by clicking the name of the relevant Job in the Stages section (under Recent History).

Other User Interface Enhancements

New Breadcrumb Trail

A new intuitive breadcrumb trail makes it clear where you are. Whether you are looking at a Plan or drilling down into a Job to find out why the Build has failed, you will always find your way back.

The tabs below a breadcrumb change depending on your current Plan context. These tabs provide access to options and data that are specific to your particular context.

Clicking back in the breadcrumb trail takes you to a higher level context with the Plan. For example, if you are viewing the Plan's Job Summary View, clicking back one step on the breadcrumb takes you to its Plan Summary View.

Screenshot: Breadcrumb for a Job's Build Result View
Build Histories

Build History tabs on the Plan and Job Summary Views show an expanded version of previous builds in the **Recent History** lists (above).

Improved Build Result Summary Tabs

The **Summary** tab of a Plan's and Job's Build Result View has the following useful features, shown in the screenshot below:

- **Test Summary** (Plan Build Result View only) — This section shows the number of:
  - New Failures — The number of Jobs built in the Plan's current build that failed *but had passed* in the Plan's previous build.
  - Existing Failures — The number of Jobs built in the Plan's current build that failed *and had also failed* in the Plan's previous build.
  - Fixed — The number of Jobs built in the Plan's current build that were fixed since the Plan's previous build.
- **Tests** — This section shows an itemised list of failed tests associated with specific Jobs in a Plan. On a Plan's Build Result View, failed tests associated with all Jobs are shown.
- **Comment** — By clicking on the 'Comment' button, you can easily add a comment associated with a Plan's or Job's build result.

**Screenshot: Test Summary on a Plan's Build Result View**

Plus over 130 fixes and improvements

Refer to our [public JIRA site](https://issues.atlassian.com) to see a full list of issues fixed in this release of Bamboo.

<table>
<thead>
<tr>
<th>JIRA Issues (50 issues)</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td><strong>Key</strong></td>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td>BAM-5845</td>
<td>Cannot start Bamboo 2.6-rc1 on Windows x64</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-6046</td>
<td>BEAC build errors: java.lang.ClassNotFoundException: com.atlassian.bamboo.results.tests.TestResultError</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-6057</td>
<td>Reset broker URL upgrade task throws NPE</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-6228</td>
<td>Foreign key constraint when running a chain on an old database</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-7044</td>
<td>Stages are executed in invalid order</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5852</td>
<td>Builds say they are queued, but are not in the queue.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5917</td>
<td>Isolating temp directory fails on where tmp directory has spaces in it</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Automatic Clover Integration is Performed for Grails, Maven and Ant, When Manual Integration is...
BAM-5920  Configured
BAM-6012  pre/post build plugin 2.4.1 no longer works with bamboo 2.6
BAM-6015  Unable to upload files in plugins using the REST module
BAM-6018  Implicit 'Build Requirements' from 'Builders' are not removed when changing 'Builders', causes builds to not execute
BAM-6058  Initial Build for HG repo triggers off two concurrent builds
BAM-6309  Unable to run Maven build if m2 folder is missing from USER_HOME
BAM-6370  Running BRS Page Logs display is incorrect
BAM-6405  Gadgets to BEAC and JBAC not working on JAC
BAM-6552  chain/edit/editChainConfiguration.ftl uses ${chainKey} instead of ${chain.key}
BAM-6742  I don't see my changes in the "My latest changes" box in the dashboard in JBAC.
BAM-6749  Comment Tooltip does not show text of the comment anymore On JBAC
BAM-6784  Importing Agents Does not maintain db id's
BAM-7057  Upgrade task 1831 will fail for every instance using a Case-Sensitive DB
BAM-7119  Installation using Java 6 fails
BAM-7191  BAM-7057  Upgrade task 1831 will fail for every instance using a Case-Sensitive DB
BAM-7191  Bamboo on Windows does not look for java in JAVA_HOME, only on PATH
BAM-1045  Option to kick off build after saving a plan create/update
BAM-2081  Users should be able to create empty projects in Bamboo.
BAM-5317  Delete multiple agents simultaneously
BAM-6251  BAM-6251  Allow admins to shutdown instances that are not connected to bamboo
BAM-7010  Allow import of Plans from Maven poms stored in Mercurial repositories
BAM-956  Plans build when you create them regardless of their configured trigger
BAM-1240  Improvement to ease of use for create plan wizard
BAM-1264  Bamboo should not automatically build project after the (first) initial checkout on a manual build plan
BAM-2212  Add a mechanism to define the Jabber resource in Bamboo.
BAM-2665  "Move Plan To New Project"
BAM-3908  Alternative build plan configuration that is only one page
BAM-5918  BAM-5918  Add workaround for when concurrent builds are disabled to use single working directory
<table>
<thead>
<tr>
<th>Ticket</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-5943</td>
<td>Elastic bamboo agent (bamboo-agent-home) uses small 10G root partition of elastic instance.</td>
</tr>
<tr>
<td>BAM-2528</td>
<td>Changing the P4 executable, results in plans not getting built.</td>
</tr>
<tr>
<td>BAM-3066</td>
<td>Bamboo server hangs if Container initialization fails</td>
</tr>
<tr>
<td>BAM-3282</td>
<td>P4Exe Is cached randomly</td>
</tr>
<tr>
<td>BAM-3476</td>
<td>Result not saved when checkout fails</td>
</tr>
<tr>
<td>BAM-4197</td>
<td>Incorrect references in wrapper.conf in Bamboo tar.gz package for Linux</td>
</tr>
<tr>
<td>BAM-4430</td>
<td>manual and scheduled shutdown in elastic instance fails to delete attached ebs volumes</td>
</tr>
<tr>
<td>BAM-5248</td>
<td>expired session cookie leads to 500 error page when configuring a plan</td>
</tr>
<tr>
<td>BAM-5382</td>
<td>Remote agents failed to shutdown after bamboo server restart</td>
</tr>
<tr>
<td>BAM-5910</td>
<td>Elastic instances don't get started because LocalAgentManagerImpl.getExecutableAgents returns offline agents</td>
</tr>
<tr>
<td>BAM-6122</td>
<td>phputil --log-xml option deprecated in phputil 3.4.3+</td>
</tr>
<tr>
<td>BAM-6133</td>
<td>custom.svn.lastchange.revision.number is omitted in build result metadata when repository advanced option 'quiet period' is enabled</td>
</tr>
<tr>
<td>BAM-6173</td>
<td>Exception on new instance startup - BambooElevatedSecurityGuarg</td>
</tr>
<tr>
<td>BAM-6176</td>
<td>After Upgrading 2.4.3 to 2.6.1 mail no longer works</td>
</tr>
<tr>
<td>BAM-6253</td>
<td>Mercurial: Plan log shows a new commit available, but build does not trigger.</td>
</tr>
<tr>
<td>BAM-6385</td>
<td>getNextBuildNumber on SubPlan still gives &quot;1&quot;</td>
</tr>
</tbody>
</table>

**Bamboo 2.7 Upgrade Guide**

**On this page:**
- Upgrading from Bamboo 2.6 to 2.7
  - Please set aside some time when upgrading to Bamboo 2.7 or later
  - Old Bamboo Plans migrated smoothly into enhanced Plans
  - Using a Mercurial repository with SSH on remote agents
  - All Bamboo versions using MS SQL 2005 and 2008 demand Read Committed with Row Versioning isolation level.
  - Configuring Plans and Jobs
  - Other Known Issues
  - Developing for Bamboo 2.7
- Upgrading from Bamboo prior to 2.6

**Supported Platforms**

Please read the Supported Platforms page for the full list of supported platforms for Bamboo.

**Upgrading from Bamboo 2.6 to 2.7**
**IMPORTANT! Back up your existing installation of Bamboo before attempting to upgrade to Bamboo 2.7!**

Significant changes will be made to your pre-existing Plans when they are migrated to the enhanced Plan structure for Bamboo 2.7.

After commencing the Bamboo 2.7 upgrade process, there is no easy way to:

- revert your Plans back to the old Bamboo 2.6.x (or earlier) structure or
- recover your Plans if you encounter a problem during the Bamboo 2.7 upgrade process.

When backing up Bamboo, we strongly recommend **backing 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 2.7 before upgrading.

---

Before you upgrade, please read the following important points that relate to Bamboo 2.7.

**Please set aside some time when upgrading to Bamboo 2.7 or later**

Please set aside some time for Bamboo to migrate Plans to the enhanced Plan structure as this process may require a significant period of time to be completed.

As a guideline, when we upgraded a Bamboo 2.6.x server running on an 8 thread/core machine with approximately 100 Plans (45,000 results), it took us approximately 2 hours to complete the Plan migration process, plus an additional 2 or more hours to complete the Bamboo re-indexing process.

The time it will take you to upgrade to Bamboo 2.7 ultimately depends on your hardware and the number of pre-existing Plans requiring migration.

**Old Bamboo Plans migrated smoothly into enhanced Plans**

Each Plan configured in a Bamboo 2.6 (or earlier) installation will be migrated across smoothly into its own enhanced Bamboo 2.7 Plan. After migration, your Plan will consist of a single Default Job in a single Default Stage.

**Using a Mercurial repository with SSH on remote agents**

You will need to upgrade your remote agents if you need your Plans/Jobs to access Mercurial repositories (via SSH authentication using key files with passphrases). For full instructions please see Upgrading Remote Agents for Mercurial.

You should also make sure that the Mercurial capability is properly configured both on the server and for agents.

**All Bamboo versions using MS SQL 2005 and 2008 demand Read Committed with Row Versioning isolation level.**

Before starting the upgrade process ensure that your current Bamboo MS SQL database is set to use Read Committed with Row Versioning as its isolation level. If you are planning to restore a Bamboo Backup Zip, ensure that the new database will have this isolation level as well. For instructions on how to set this isolation level, please see this page.

**Configuring Plans and Jobs**

As a result of the enhanced Plan structure in Bamboo 2.7, you will notice that some configuration settings associated with your old Bamboo Plans are now only available when viewing the Default Jobs of these Plans (following the upgrade). Hence, you will need to ‘drill down’ to an upgraded Plan’s Default Job to access some of the configuration settings you would normally have accessed when editing/configuring a Plan in Bamboo 2.6.x or earlier.

You can easily access a Plan’s Default Job or any other of its Jobs, by viewing the Plan’s Plan Summary tab and clicking the Job’s name in the Stages section of that page.
Other Known Issues

Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases, we publish information about these other known issues in the Bamboo Knowledge Base. Before you begin the upgrade, please check for any of these other known issues in the Bamboo Knowledge Base first and if provided, follow the instructions to apply any necessary patches.

If you encounter a problem during the upgrade and cannot solve it, please create a support ticket and one of our support engineers will help you.

Developing for Bamboo 2.7

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 2.7 guide, which outlines changes in Bamboo 2.7 that may affect Bamboo plugins compiled for Bamboo version 2.6.x or earlier.

Upgrading from Bamboo prior to 2.6

If you’re upgrading from versions prior to 2.6, please upgrade to 2.6.3 release first before upgrading to 2.7. As stated above, 2.7 involves major changes to the database structure and a phased upgrade is recommended.

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 are upgrading from Bamboo 2.5 or earlier, you will need to set aside more time for Bamboo to migrate its test result data stored in XML files on the filesystem into the database. This time is additional to that mentioned above.

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 2.6.

Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.7.2 Release Notes
25 November 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.7.2.

We’ve fixed several bugs in this release. Please see the ‘Updates and Fixes in this Release’ section below for details.

Bamboo 2.7.2 is of course free to all customers with active Bamboo software maintenance.

Don’t have Bamboo 2.7 yet?

Take a look at all the new features in the Bamboo 2.7 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.7.2 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.7.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

### JIRA Issues (9 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-7129</td>
<td>User should be told that after setting custom capabilities on elastic images, instance has to be restarted</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Janusz Gorycki [Atlassian]</td>
<td>—</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Oct 20, 2010</td>
<td>Nov 22, 2010</td>
<td></td>
</tr>
</tbody>
</table>

Bamboo 2.7.2 Upgrade Guide

Upgrading from Bamboo 2.7.1 to 2.7.2

Please follow the Bamboo Generic Upgrade Guide.

⚠️ No additional upgrade tasks are required to upgrade from Bamboo 2.7.1 to 2.7.2.

Upgrading from Bamboo 2.7 or earlier

In addition to the above, please read the Bamboo 2.7 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 2.7.1 Release Notes
10 November 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.7.1.

Please see the 'Updates and Fixes in this Release' section below for details of the bugs fixed in this release.

Bamboo 2.7.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.7 yet?

Take a look at all the new features in the Bamboo 2.7 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading from versions before 2.7, please read the Bamboo 2.7 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.7.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (4 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>BAM-7299</td>
</tr>
<tr>
<td>BAM-4995</td>
</tr>
</tbody>
</table>

Bamboo 2.7.1 Upgrade Guide

Upgrading from Bamboo 2.7 to 2.7.1

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.7 to 2.7.1.

Upgrading from Bamboo 2.7 or earlier

In addition to the above, please read the Bamboo 2.7 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 2.7.3 Release Notes

15 December 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.7.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.7.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.7 yet?

Take a look at all the new features in the Bamboo 2.7 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo
If you are upgrading, please read the Bamboo 2.7.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.7.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (7 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-7453</td>
</tr>
<tr>
<td>BAM-7451</td>
</tr>
<tr>
<td>BAM-7450</td>
</tr>
</tbody>
</table>

Bamboo 2.7.3 Upgrade Guide

Upgrading from Bamboo 2.7.2 to 2.7.3

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.7.1 to 2.7.3.

Upgrading from Bamboo 2.7 or earlier

In addition to the above, please read the Bamboo 2.7 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 2.7.4 Release Notes

18 February 2011

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.7.4.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.7.4 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.7 yet?

Take a look at all the new features in the Bamboo 2.7 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.7.4 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.7.4 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (6 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>
Bamboo 2.7.4 Upgrade Guide

Upgrading from Bamboo 2.7.3 to 2.7.4

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.7.3 to 2.7.4.

Upgrading from Bamboo 2.7 or earlier

In addition to the above, please read the Bamboo 2.7 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

Bamboo 2.6 Release Notes

1 June 2010

The Atlassian Bamboo team is proud to release Bamboo 2.6.

This release brings a host of performance improvements to your continuous integration strategy. Bamboo 2.6 now provides support for up to 100 remote agents and along with several user interface enhancements, you can manage and build many more plans simultaneously from a single Bamboo server.

The 'Current Activity' page incorporates several improvements that allow you to manage builds in real time more effectively. You can also comment on build results to record and let others know what might be happening with a build.

Imports and exports are now faster, more reliable and require less memory to perform. You can now also selectively choose to expire build logs to help save disk space.

If you use Elastic Bamboo, the Bamboo server can now automatically manage your elastic instances. This removes the need for you to manually start and shut down elastic instances, and can help keep your elastic instance usage costs to a minimum.

Bamboo also supports continuous integration for Grails projects and can also automatically incorporate Clover code coverage reports into their build results.

Upgrading to Bamboo 2.6 is free for all customers with active Bamboo software maintenance.

Highlights of this release:

- Support for up to 100 Remote Agents
- Revamped Dashboard Pages and Other Usability Enhancements
- Performance and Security Improvements
- Automatically Managed Elastic Instances
- Grails Integration with Optional Clover Code Coverage
- Plus over 140 fixes and improvements

Thank you for your feedback:

🌟 50 new features and improvements implemented
🌟 Over 50 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!

Upgrading to Bamboo 2.6

You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 2.6 Upgrade Guide.

Support for up to 100 Remote Agents

Bamboo’s scalability has been dramatically increased, now providing support for up to 100 remote agents — up from the previous supported maximum of 25 remote agents. You can now build many more plans simultaneously from a single Bamboo server, with the power of up to 100 remote agents.

To make managing large numbers of remote agents easier, the following user interface enhancements have been introduced into Bamboo:

- In the administration console, builders and JDKs are now grouped by their labels.

  Screenshot: Managing Builders via the Administration Console

- Remote agents are now grouped into separate Online and Offline lists.

  Screenshot: Managing Remote Agents
More...

- When specifying the capability requirements of a plan, you can easily access further information about the agents associated with the plan via improved tooltips. To do this, simply click the name of the agent in the tooltip.

_Screenshot: Plan Requirement Popup_

For more information about Bamboo’s pricing, please refer to the Bamboo pricing page.

2

Revamped Dashboard Pages and Other Usability Enhancements

The Bamboo Dashboard’s 'Current Activity' page has been redesigned to provide more helpful information and make it much more intuitive to use. Bamboo administrators can more easily administer online agents and the build queue on this page.

- The new ‘Building’ section shows which plans are currently being built by an online agent. Each plan’s build in this section also provides an estimate of the remaining time required for its agent to complete the build process.

_Screenshot: New ‘Building’ Section_
The ‘Recently Built’ section is an ‘activity stream’ which constantly updates to show builds which have just completed. You can comment on build results and also set up a RSS feed, to be informed about builds results as soon as they are generated.

Bamboo administrators can easily:
- Reorder plans in the ‘Queue’ through a simple drag-and-drop action
- Enable or disable online agents directly from ‘Online Agents’ pop-up balloon (accessible from the ‘Building’ section)

Bamboo administrators can easily:
- Reorder plans in the ‘Queue’ through a simple drag-and-drop action
- Enable or disable online agents directly from ‘Online Agents’ pop-up balloon (accessible from the ‘Building’ section)
Throughout the Bamboo user interface, plans in a queue are now indicated with a new icon and only plans whose builds are currently being built on a Bamboo agent are indicated with the icon.

If a plan's build was not built, the summary page for its build result will indicate this explicitly, rather than indicating that the build had failed.

Performance and Security Improvements

Several performance improvements have been implemented throughout Bamboo, in particular:

- Bamboo Plan Summary pages now obtain data more efficiently and complete loading in much less time.
- Bamboo Imports and Exports are now more reliable, faster and require less memory.
- In addition to selectively expiring user-defined build artifacts to preserve storage space, Bamboo now allows you to selectively expire build logs too. Of course, you can still choose to expire all build result data (including build artifacts and logs) too.

In Bamboo 2.5.5, we introduced a Captcha feature to help prevent brute force attacks on your Bamboo server. This feature would be activated after a specified number of consecutive failed login attempts. In Bamboo 2.6, this Captcha feature has been extended to cover public signup.

Automatically Managed Elastic Instances

If you use Elastic Bamboo, you no longer have to manually start and shut down elastic instances. Instead, you can choose one of Bamboo's new automatic elastic instance management settings to manage the way elastic instances are started and shut down in Bamboo, and to help reduce your elastic instance usage costs.

This feature also allows Bamboo to start elastic instances capable of executing plans in the build queue, if no other online agents can do so.

Bamboo provides the following three automatic elastic instance management presets:

- **Default** — Balances build queue clearance rates with elastic instance usage costs.
- **Aggressive** — Favours higher build queue clearance rates but with higher elastic instance usage costs.
- **Passive** — Favours lower instance usage costs but with lower build queue clearance rates.

These presets alter the values of five criteria (indicated in the screenshots below) that define how elastic instances are started and shut down. You can also customise these criteria to further fine tune how Bamboo manages elastic instances.
Bamboo now provides continuous integration capabilities for Grails projects. To do this, create a new plan or edit an existing one and on the plan's 'Builder' tab, select a Grails builder from the list of builders automatically detected by Bamboo, or you can add and use a new Grails builder capability directly from this tab.

You can also configure Bamboo to automatically conduct Clover code coverage on a Grails Bamboo plan. When Bamboo runs this Grails plan, Bamboo will automatically install the Clover plugin and generate a code coverage report of your Grails build result.

Using the Grails Clover feature requires a valid Clover license.

Bamboo automatically detects Grails builders based on the value of the computer's `GRAILS_HOME` environment variable.

If you use Elastic Bamboo, our EC2 image supports Grails 1.2.1 and 1.3.1 builder capabilities (as well as Maven 2.1).
More...

Plus over 140 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (153 issues)</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-4407 REST - Hibernate exception on build details load</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5620 Clover 3 coverage reports cannot be parsed by Bamboo</td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
BAM-5689  Impossible to setup new instance of Bamboo

BAM-5845  Cannot start Bamboo 2.6-rc1 on Windows x64

BAM-65  Allows CVS repo to timeout and report on locking issues

BAM-5192  Ability to delete build working directory after a successful build

BAM-5218  When downsizing a license plans users are directed to delete on the _old_ server

BAM-5234  Cannot connect to AWS/EC2 when bamboo app server is behind a proxy. EC2_JVM_ARGS have no effect

BAM-5292  Improve Performance of the Build Configuration Screen

BAM-5682  Ability to delete build working directory after a failed build

BAM-1737  OutOfMemory error when exporting/importing large Bamboo instances

BAM-5172  CVS deletes working copy when using tag/branch and ampersand modules after initial build

BAM-5189  $(system.bamboo.agent.home) should be defined for all agents, not just remote ones

BAM-5276  Maven embedder throws exception if $USER/.m2 is not existing or in a different location

BAM-5281  JIRA Bamboo Plugin creates deadlock in JIRA and Bamboo applications when starting them up in the same Tomcat server

BAM-5354  cannot create plan when plan permissions configuration contains user or group name with whitespace

BAM-5456  CAPABILITY and NOTIFICATIONS table access with Empty values causes (Oracle) Deadlock in larger instances

BAM-5754  Gadgets: Bamboo Plans gadget is giving errors when served from BEAC

BAM-5789  Deadlock during BuildNumber generation

BAM-1948  Option to not export build logs

BAM-3223  Bamboo support 100 agents

BAM-3344  Automatically control starting and stopping of Elastic Agents based on load

BAM-1239  Ordering of the remove plans screen

BAM-2172  Use database, rather than Lucene, for indexing of highly structured data

BAM-2641  Improve the Bamboo persistency

BAM-3662  Import without restart

BAM-4114  Bamboo should provide a separate tmp directory for each build

BAM-4887  Build failure detection for Maven 3 - BUILD SUCCESS

BAM-5217  Configurable base url on export
BAM-5237  add more repository types to plan creation based on maven

BAM-5300  Loading the Build Summary screen filters build results in memory

BAM-5668  Make sessionID a HttpOnly cookie

BAM-2542  Building from CVS tag with force clean results in no changes checked out after initial build.

BAM-3463  Dual digit JIRA issue numbers are rendered wrong

BAM-3707  Dashboard with many projects performs very poorly in Internet Explorer

BAM-3793  User Profile > Edit: Combo box unconventionally used for action not selection.

BAM-4274  CVS Repository Calls dont time out

BAM-4663  Double requests generated on dropdown menu on Plan page in Firefox

BAM-4871  REST: Incorrect representation of JSON data

BAM-5223  Why does Bamboo show the last build output in the live logs.

BAM-5245  Initial plan setup causes FM exception

BAM-5247  Bamboo is broken in chrome/firefox

BAM-5273  Clicking an item on the Builds menu run fire off two requests

BAM-5295  Deadlock in RemoteElasticInstanceImpl

BAM-5308  Revert BAM-5006 - it has broken the functionality of ${bamboo.custom.svn.revision.number}

BAM-5381  Bamboo remote agent does not restart if a FATAL exception is thrown during startup under wrapper.sh

BAM-5383  By deleting an error on the System Error page you will be returned to the Home Page

BAM-5396  Clover license set in .m2/settings.xml overrides license set explicitly in func tests

BAM-5400  SVN checkouts are not based on the global repository revision number

BAM-5528  Plans are executed twice on the same vcs revision key.

BAM-5572  JBAC is sending too many Notifications

BAM-5574  NPE in PaginationAwareInterceptor

BAM-5601  encoding declared in old XML export files does not match actual content

BAM-5612  Substitution of myBaseUrl in administration.xml right after import nukes data required for upgrade tasks

BAM-5630  TestCaseResultError cannot export data which contains "]]>

BAM-5642  aggregate functions not allowed in order by clause on some dbmases
BAM-5644  Upgrade task fails on MS SQL Server due to deadlock
BAM-5645  Deadlocks reported during change detection on MS SQL Server
BAM-5647  Export of CommitFiles fails for null commitFileRevision
BAM-5702  UI bug on config/Builder page - stacktrace shown
BAM-5761  Null author name causes export to fail on Oracle
BAM-5762  Export - Too many open cursors on Oracle
BAM-5793  Investigate LazyInitializationException
BAM-5799  Avoid duplicate records for authors
BAM-5800  Logger not initialized + exceptions during build on local agent
BAM-5812  Current Activity -> Queue does not show all queued builds
BAM-4639  Add a "description" field for plans
BAM-5452  Plugin Points for Build Chains
BAM-5651  Brute force protection
BAM-5656  Captcha on signup
BAM-3236  Add option to disable automatic building of new build plans
BAM-4526  Bamboo to use svnkit ISVNAuthenticationManager instead of DefaultSVNAuthenticationManager
BAM-4639  Add a "description" field for plans
BAM-4757  Make the axis on the build duration per build graph start at 0
BAM-4764  Please export BuildDefinitionConverter so that plugins can have it injected.
BAM-5087  Add CVS support for importing from Maven
BAM-5098  Add plan name to the list of build specific variables
BAM-5272  Support java proxy connection to EC2
BAM-5294  Warning box refers to fixed issue
BAM-5309  Improve dashboard performance by reducing calls to getUser() and isFavourite()
BAM-5310  Reduce calls to localAgentManager.getAgent() on the Build Results Table
BAM-5311  FreemarkerContext:hasBuilds() should not request all plans from the DashboardCachingManager
BAM-5312  Move/Delete Builds page is slow
BAM-5337  make number of 'recent builds' displayed in dashboard > current activity flexible
<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-5358</td>
<td>Persist some calculated data for ArtifactLink</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5385</td>
<td>Allow user to see more of an agent's build history</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5422</td>
<td>Remove deprecated properties on BuildCompletedEvent</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5442</td>
<td>Expire build logs</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5515</td>
<td>AccessLoggingFilter should allow everything to be logged</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5569</td>
<td>'view' a plan config tab and click 'edit' - you land on a tab that you've last edited (and not the one you just viewed)</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5576</td>
<td>comprehensive Javadoc for SystemProperty.java</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5700</td>
<td>Option for XMPP TLS</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2969</td>
<td>Bamboo doesn't delete administrationconfiguration.xml file after finishing import</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4565</td>
<td>Bamboo does not start correctly on Windows7</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4570</td>
<td>Bamboo Acceptance Tests are failing on non-Panda environments</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4574</td>
<td>Package <a href="http://datejs.googlecode.com/files/date.js">http://datejs.googlecode.com/files/date.js</a> for gadgets</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4962</td>
<td>Bamboo can not look at &gt;1024 build results when looking for test failures.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5004</td>
<td>Long running task icon is off-centre</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5124</td>
<td>StopBuildManager with abandon result does not abandon result</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5145</td>
<td>No default tab on dashboard</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5148</td>
<td>Dashboard is not updated to reflect newly created chain</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5151</td>
<td>Chain Actions menu on view Chain page is gone</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5152</td>
<td>NPE in notifications when trying to run chain</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5153</td>
<td>Running a chain from the actions menu runs the chain twice</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5154</td>
<td>Repository is null and throws exception when running chain</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5166</td>
<td>BuildState not set properly when build is not run, UI also dies</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5168</td>
<td>Build result view for non executed build shows ftl errors</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5173</td>
<td>Exception on System Info page in Administration section</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5180</td>
<td>Should not show chains or builds tab if there are no items to display.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5181</td>
<td>NPE on dashboard when there are no builds or chains present</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-5182</td>
<td>Agent upload of artifacts fails to set Content-Length in HTTP/1.1 (on apache lighthttpd)</td>
<td>Resolved</td>
</tr>
<tr>
<td>Issue Key</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>BAM-5184</td>
<td>Can't create Plan from POM if the only Builder detected was the Elastic Agent</td>
<td></td>
</tr>
<tr>
<td>BAM-5207</td>
<td>Do not show elastic capabilities in the builders/jdos/individual capabilities list if elastic bamboo is disabled.</td>
<td></td>
</tr>
<tr>
<td>BAM-5225</td>
<td>Delta State on BuildResultsSummary is incorrect for initial build</td>
<td></td>
</tr>
<tr>
<td>BAM-5228</td>
<td>Dashboard is not updated properly after a plan is deleted</td>
<td></td>
</tr>
<tr>
<td>BAM-5244</td>
<td>Cannot undo Maven POM dependency management</td>
<td></td>
</tr>
<tr>
<td>BAM-5262</td>
<td>Build Action Menus still look dodgy.</td>
<td></td>
</tr>
<tr>
<td>BAM-5263</td>
<td>Return URL should be restricted to current server</td>
<td></td>
</tr>
<tr>
<td>BAM-5287</td>
<td>Name change from Build to Plan security aware broke nant plugin</td>
<td></td>
</tr>
<tr>
<td>BAM-5293</td>
<td>urls with <code>&amp;amp;</code> cause NPE</td>
<td></td>
</tr>
<tr>
<td>BAM-5296</td>
<td>Differences in the implementation of a BuildResultsSummary cause labels to be duplicated on the <code>Related builds by date</code> tab of the Jira Bamboo Plugin</td>
<td></td>
</tr>
<tr>
<td>BAM-5324</td>
<td>Exception in oauth consumer page header</td>
<td></td>
</tr>
<tr>
<td>BAM-5326</td>
<td>getLabelNames on BuildResultsSummary failing due to LazyInitialisationException</td>
<td></td>
</tr>
<tr>
<td>BAM-5328</td>
<td>Large number of exceptions being thrown on JBAC</td>
<td></td>
</tr>
<tr>
<td>BAM-5344</td>
<td>Make the redirect less ugly for a non-admin user when evaluation license expires</td>
<td></td>
</tr>
<tr>
<td>BAM-5387</td>
<td>Breadcrumbs for &quot;Not Built&quot; build result are not working</td>
<td></td>
</tr>
<tr>
<td>BAM-5394</td>
<td>Recent Builds display implies only 15 builds have ever run on the agent</td>
<td></td>
</tr>
<tr>
<td>BAM-5407</td>
<td>ConcurrentModificationException stacktrace in email body</td>
<td></td>
</tr>
<tr>
<td>BAM-5413</td>
<td>Calling getContentType on a JarUrlConnection closes the stream</td>
<td></td>
</tr>
<tr>
<td>BAM-5416</td>
<td>Fix the notificationDispatcher to play nice with plugins 2.0 plugins</td>
<td></td>
</tr>
<tr>
<td>BAM-5425</td>
<td>Adding correct AWS account credentials gives a stack trace if the account does not have an EC2 subscription</td>
<td></td>
</tr>
<tr>
<td>BAM-5431</td>
<td>REST API missing / in url for build log artifact</td>
<td></td>
</tr>
<tr>
<td>BAM-5486</td>
<td>System info page is slow for large instances</td>
<td></td>
</tr>
<tr>
<td>BAM-5504</td>
<td>Problems with REST API's build log artifact</td>
<td></td>
</tr>
<tr>
<td>BAM-5510</td>
<td>Broken link to the dashboard on the /error/error.action</td>
<td></td>
</tr>
<tr>
<td>BAM-5585</td>
<td>java.lang.ClassCastException: $Proxy222 cannot be cast to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>com.atlassian.bamboo.build.CustomBuildCompleteAction</td>
<td></td>
</tr>
<tr>
<td>BAM-5624</td>
<td>Depending on underlying database, test case import can fail when test case name is too long</td>
<td></td>
</tr>
<tr>
<td>BAM-5631</td>
<td>No session exception while using plugin from tutorial.</td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 2.6 Upgrade Guide

On this page:

- Upgrading from Bamboo 2.5 to 2.6
  - Please set aside some time when upgrading to Bamboo 2.6 or later
  - Automatic Clover Integration Issue
  - Bamboo Home Directory - Disk Usage changes
  - Changes in seraph-config.xml that affect new Bamboo security features
  - Other Known Issues
  - Developing for Bamboo 2.6
- Upgrading from Bamboo prior to 2.5

**Supported Platforms**

Please read the Supported Platforms page for the full list of supported platforms for Bamboo.

**Upgrading from Bamboo 2.5 to 2.6**

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 2.6 before upgrading.
Before you upgrade, please read the following important points that relate to Bamboo 2.6.

**Please set aside some time when upgrading to Bamboo 2.6 or later**

As part of the performance improvements in version 2.6, test result data is stored differently. In versions of Bamboo prior to (and excluding) 2.6, all test result data has been stored in XML files on the filesystem. From Bamboo 2.6, some* of this test result data is stored in the database, permitting quicker retrieval of this information (and consequently faster Bamboo responsiveness) than what can be achieved by accessing XML files.

* Only test result data from failed and fixed builds is stored in the database, since this data will most likely be examined by Bamboo users. (Fixed builds are those which built successfully but had failed the previous time they were built.) Be aware that the test result data for successful builds is still stored in XML files on the filesystem.

During the Bamboo 2.6 upgrade process, relevant test result data generated by previous versions of Bamboo will automatically be migrated to the database when Bamboo 2.6 first starts up. No user-intervention is required during this process, which only runs once.

> All subsequent Bamboo starts will not involve this data migration process.

**Bamboo administrators should be aware that this data migration process might take some time**, depending on the amount of data that needs to be moved to the database. In many cases, this process should be completed within a matter of minutes. However, if your stored test result data is extensive, this data migration process could take over an hour.

The table below is a guideline to help provide an estimate on how long it will take this data migration process to complete during the Bamboo upgrade procedure. The first column is a multiplication of the number of builds in history with the average number of test results per build. You can estimate the number of builds in history by multiplying the number of plans configured in Bamboo by the number of times each of these plans has run. For example, if you have 20 plans configured and each plan has run 300 times, there will be 6,000 builds (i.e. 20 x 300) in the build history. Note that expired builds are removed from the build history.

<table>
<thead>
<tr>
<th>Number of Builds in History x Number of Tests per Plan</th>
<th>Estimated Migration Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,500,000 (5,000 x 500)</td>
<td>&lt; 3 min</td>
</tr>
<tr>
<td>5,000,000 (10,000 x 500)</td>
<td>&lt; 6 min</td>
</tr>
<tr>
<td>10,000,000 (20,000 x 500)</td>
<td>&lt; 10 min</td>
</tr>
<tr>
<td>15,000,000 (30,000 x 500)</td>
<td>&lt; 15 min</td>
</tr>
<tr>
<td>20,000,000 (40,000 x 500)</td>
<td>&lt; 25 min</td>
</tr>
<tr>
<td>25,000,000 (50,000 x 500)</td>
<td>&lt; 45 min</td>
</tr>
<tr>
<td>30,000,000 (60,000 x 500)</td>
<td>&lt; 75 min</td>
</tr>
<tr>
<td>35,000,000 (70,000 x 500)</td>
<td>up to 3 hours</td>
</tr>
</tbody>
</table>

The estimated migration time (above) is only just an estimate. The actual time it will take for this step of your Bamboo 2.6 upgrade to complete will also strongly depend on the performance of the hardware running Bamboo and the database that Bamboo uses.

**Automatic Clover Integration Issue**

A bug in Bamboo 2.6 forces automatic Clover integration and adds Clover targets or goals for Ant, Maven and Grail builds, despite having opted for manual Clover integration.

If you are affected by this issue, please apply the patch provided in JIRA issue BAM-5920.

**Bamboo Home Directory — Disk Usage changes**

This issue only affects Bamboo 2.6 and is fixed in Bamboo 2.6.1 and above.

Due to backend changes in Bamboo 2.6 (implemented for a feature that will be fully supported in a future version of Bamboo), the structure for storing temporary build files in the Working Directory has changed.

In versions of Bamboo prior to (and excluding) 2.6 had the following structure:

```
.../xml-data/build-dir/PLAN-KEY/
```

From Bamboo 2.6, the location for storing this data is now:
Hence, each agent now has its own directory for storing temporary build files, which means that the disk usage requirements for the Bamboo Home directory have increased in Bamboo 2.6. If you are concerned about disk usage, please upgrade to Bamboo 2.6.1 or above.

**Changes in seraph-config.xml that affect new Bamboo security features**

As part of the brute force attack protection feature (introduced in Bamboo 2.5.5) and Captcha on public signup, the following lines have been added to the seraph-config.xml file.

```xml
<rolemapper class="com.atlassian.bamboo.user.authentication.BambooRoleMapper"/>
<authenticator class="com.atlassian.bamboo.user.authentication.BambooAuthenticator"/>
<controller class="com.atlassian.bamboo.user.authentication.BambooSecurityController"/>
<elevatedsecurityguard
class="com.atlassian.bamboo.user.authentication.BambooElevatedSecurityGuard"/>
```

If you use a customised version of the seraph-config.xml file with Bamboo, you will need to ensure that these lines of code are added to your customised seraph-config.xml, to ensure the availability of these new Bamboo security features.

**Other Known Issues**

Sometimes we find out about a problem with the latest version of Bamboo after we have released the software. In such cases, we publish information about these other known issues in the Bamboo Knowledge Base. Before you begin the upgrade, please check for any of these other known issues in the Bamboo Knowledge Base first and if provided, follow the instructions to apply any necessary patches.

If you encounter a problem during the upgrade and cannot solve it, please create a support ticket and one of our support engineers will help you.

**Developing for Bamboo 2.6**

If you are a Bamboo plugin developer, please refer to our Changes for Bamboo 2.6 guide, which outlines changes in Bamboo 2.6 that may affect Bamboo plugins compiled for Bamboo version 2.5.x or earlier.

**Upgrading from Bamboo prior to 2.5**

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 2.5.

⚠️ Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

**Bamboo 2.6.3 Release Notes**

13 October 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.6.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.6.3 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.6 yet?**

Take a look at all the new features in the Bamboo 2.6 Release Notes and see what you are missing out on!

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.6.3 Upgrade Guide.

**Updates and Fixes in this Release**
The issues addressed in Bamboo 2.6.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

### JIRA Issues (4 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-6309</td>
<td>Unable to run Maven build if m2 folder is missing from USER_HOME</td>
<td>Krzysztof Brazulewicz [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>Red</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 20, 2010</td>
<td>Sep 22, 2010</td>
<td></td>
</tr>
</tbody>
</table>

### Bamboo 2.6.3 Upgrade Guide

**Upgrading from Bamboo 2.6.2 to 2.6.3**

Please follow the Bamboo Generic Upgrade Guide. No additional upgrade tasks are required to upgrade from Bamboo 2.6.2 to 2.6.3.

**Upgrading from Bamboo 2.5.x or earlier**

In addition to the above, please read the Bamboo 2.6 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

### Bamboo 2.6.2 Release Notes

**6 August 2010**

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.6.2.

We’ve fixed several bugs in this release. Please see the ‘Updates and Fixes in this Release’ section below for details.

Bamboo 2.6.2 is of course free to all customers with active Bamboo software maintenance.

**Don’t have Bamboo 2.6 yet?**

Take a look at all the new features in the Bamboo 2.6 Release Notes and see what you are missing out on!

### Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.6.2 Upgrade Guide.

### Updates and Fixes in this Release

The issues addressed in Bamboo 2.6.2 are shown below. To view the list in JIRA, please refer to our main JIRA site.

**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>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-6794</td>
<td>Importing Agents Does not maintain db id’s</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Red</td>
<td>Resolved</td>
<td>Fixe</td>
<td></td>
</tr>
<tr>
<td>BAM-5917</td>
<td>Isolating temp directory fails on where tmp directory has spaces in it</td>
<td>Marcin Gardias [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Red</td>
<td>Resolved</td>
<td>Fixe</td>
<td></td>
</tr>
<tr>
<td>BAM-6176</td>
<td>After Upgrading 2.4.3 to 2.6.1 mail no longer works</td>
<td>Przemek Bruski [Atlassian]</td>
<td>Joshua Grigonis</td>
<td>Red</td>
<td>Resolved</td>
<td>Fixe</td>
<td></td>
</tr>
<tr>
<td>BAM-6133</td>
<td>custom.svn.lastchange.revision.number is omitted in build result metadata when repository advanced</td>
<td>Marcin Gardias</td>
<td>Ulrich Kuhnhardt</td>
<td>Red</td>
<td>Resolved</td>
<td>Fixe</td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 3.1 Documentation

### Bambo 2.6.2 Upgrade Guide

**Upgrading from Bamboo 2.6.1 to 2.6.2**

Please follow the Bamboo Generic Upgrade Guide.

*No additional upgrade tasks are required to upgrade from Bamboo 2.6.1 to 2.6.2.*

**Upgrading from Bamboo 2.5.x or earlier**

In addition to the above, please read the Bamboo 2.6 Upgrade Guide and the Upgrade Guide for every version you are skipping during the upgrade. The complete list of Upgrade Guides is available in the Bamboo Upgrade Guides section.

### Bamboo 2.6.1 Release Notes

**8 June 2010**

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.6.1.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.6.1 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.6 yet?**

Take a look at all the new features in the Bamboo 2.6 Release Notes and see what you are missing out on!

### Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.6.1 Upgrade Guide.

### Updates and Fixes in this Release

The issues addressed in Bamboo 2.6.1 are shown below. To view the list in JIRA, please refer to our main JIRA site.

**JIRA Issues (6 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-5920</td>
<td>Automatic Clover Integration is performed for Grails, Maven and Ant, When Manual</td>
<td>Krystian Brazulewicz [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>[Resolved]</td>
<td>[Fixed]</td>
<td>[Jun 02, 2010]</td>
<td>[Feb 25, 2011]</td>
<td></td>
</tr>
</tbody>
</table>
### Bamboo 2.6.1 Upgrade Guide

**Upgrading from Bamboo 2.6 to 2.6.1**

Please follow the [Bamboo Generic Upgrade Guide](#).

#### No additional upgrade tasks are required to upgrade from Bamboo 2.6 to 2.6.1.

**Upgrading from Bamboo 2.5.x or earlier**

In addition to the above, please read the [Bamboo 2.6 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 2.5 Release Notes

**4 January 2010**

The Atlassian Bamboo team is proud to release **Bamboo 2.5**.

Bamboo 2.5 brings a host of new Maven integration features to your favourite build tool. You can now elect to have your plan dependencies managed by your Maven project. Bamboo will automatically set up the dependencies based on the information in your pom.xml file. If you have information for a build plan already in your Maven project, you can import your plan into Bamboo as well. Simply specify the location of your pom.xml plus any required authentication details and Bamboo will do the rest.

Two new bulk actions have been added to Bamboo in this release. The first new bulk action allows you to enable the new Maven 2 dependencies feature for multiple plans. The second new bulk action can be used to run manual builds for multiple plans without triggering dependencies (For example, if you want to run initial builds to create dependencies for plans with the Maven 2 dependencies feature enabled).

We've also streamlined both the Bamboo setup wizard and plan creation wizard. Express setup options, inline functions as well as redesigned screens make it even easier for you to complete these setup tasks.

Finally, if you are running Confluence, you will be happy to know that Bamboo gadgets are fully compatible with Confluence 3.1.

#### Atlassian Bamboo 100 Remote Agent Beta Program

We are pleased to announce a beta program to test Bamboo with more than 25 remote agents. If you would like to participate, please sign up via this form:

[Sign up for the Atlassian Bamboo 100 Remote Agent Beta Program](#)

Upgrading to Bamboo 2.5 is free for all customers with active Bamboo software maintenance.

### Highlights of this release:

- Maven Dependency Management
- Plan Import from a pom.xml
- Additional Bulk Actions
- Streamlined Plan Creation
- Express Setup Wizard
Thank you for your feedback:

🌟 34 new features and improvements implemented
🌟 33 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!

<table>
<thead>
<tr>
<th>Upgrading to Bamboo 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can download Bamboo from the Atlassian website. If upgrading from a previous version, please read the Bamboo 2.5 Upgrade Guide.</td>
</tr>
</tbody>
</table>

**Highlights of Bamboo 2.5**

1. **Maven Dependency Management**

Bamboo 2.5 can now use Maven (Maven 2 only) to manage your dependencies between plans. You can choose to allow Bamboo to do this when you create a new plan or edit an existing plan that uses Maven 2 as the builder. Bamboo will automatically set up the dependencies based on the information in your pom.xml file.

- Read more about Viewing a Job's Maven Dependencies.

2. **Plan Import from a pom.xml**

We've also introduced a new feature that allows you to create a plan based on information from your Maven (Maven 2) project. There's no need to re-enter information already specified in the pom.xml. Simply enter the location of your pom.xml and any required authentication details, and Bamboo will parse the pom.xml to create your build plan.
Read more about Import a Maven 2 Project.

Additional Bulk Actions

We've also added two new bulk actions to Bamboo, enable Maven 2 dependencies for multiple plans and run manual builds for multiple plans. Enabling Maven 2 dependencies for multiple plans allows Maven to manage dependencies between plans (described above). Running manual builds for multiple plans via the bulk actions menu runs the selected builds with option of triggering dependencies.

Read more about modifying Multiple (Bulk) Plans.

Streamlined Plan Creation

The plan creation wizard now allows you to create a plan without progressing through all of the steps. If you are cloning a plan or don't want to provide all the information on the latest tabs at the time of creation, you can save it after the second or third step respectively. If you are still working on your plan setup, you can prevent the initial build from running when you save too.

We've also improved the Builder and Notifications tabs in the plan creation wizard. You can now add new JDKs, Builders, Mail and IM servers inline without leaving the wizard.

Read more about the Creating a Plan.
Express Setup Wizard

We've streamlined the Setup Wizard for Bamboo in this release. If you are happy to use the default settings and embedded database bundled with Bamboo, you can get Bamboo up and running in only two steps. If you want to customise all of the settings, the longer version of the setup wizard has also been improved to allow you enter the required information in less steps.

- Read more about running the Setup Wizard.

Plus over 70 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (76 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>BAM-230</td>
</tr>
<tr>
<td>BAM-1329</td>
</tr>
<tr>
<td>BAM-1354</td>
</tr>
<tr>
<td>BAM-5106</td>
</tr>
<tr>
<td>BAM-5107</td>
</tr>
<tr>
<td>BAM-5108</td>
</tr>
<tr>
<td>BAM-5110</td>
</tr>
<tr>
<td>BAM-5112</td>
</tr>
<tr>
<td>BAM-5114</td>
</tr>
<tr>
<td>BAM-5115</td>
</tr>
<tr>
<td>BAM-4668</td>
</tr>
</tbody>
</table>
BAM-4670 support bulk manual build.

BAM-4833 Link to IntelliJ or Eclipse from build screen

BAM-4636 There is no way to edit a Global Variable (only Add/Delete)

BAM-1543 Notification Policy: first failure and first success

BAM-2568 FishEye integration should support FishEye instances configured to view a subset of a VCS repository

BAM-3878 Contents of comment tab completely blank when logged out

BAM-5039 Include agent information in build history list

BAM-5105 Use autocomplete plan picker for Gadgets in JIRA

BAM-5109 Ability to finish the plan creation wizard with minimal data

BAM-5111 Added a Plan enabled footer to the plan configuration screen

BAM-5113 Express setup for initial install

BAM-2550 Global Variable Substitution for Web Url Fields.

BAM-4395 feature to specify POM file name

BAM-4645 Faster Setup: Visual indication for long running tasks in Setup Wizard

BAM-4651 Remove use of backport-util-concurrent in favour of java.util.concurrent

BAM-4671 support a flag to ignore dependencies when bulk manual building

BAM-4750 Use Maven's internal API to determine whether an artifact is a snapshot version.

BAM-4751 Automatic dependencies shall override user dependencies

BAM-4841 Dependency block should block on parents as well

BAM-4848 Maven dependency should be parsed by default(?)

BAM-4913 Add note in plan permission advising users that global admins have all permissions

BAM-4632 Bamboo is counting ignored JUnit tests.

BAM-4644 moving plans across projects does not rename log files (artifacts).

BAM-4890 Some gadgets not saving certain fields after refreshing the page

BAM-4911 Cannot configure a build that doesn't produce test results

BAM-1074 Bad recovery from failure for initial checkout to complete

BAM-1872 Bamboo UI is incorrect if plan has been disabled while it is doing its initial checkout

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved

Resolved
BAM-2472  Classloading fails when server is locked for exporting
BAM-2611  Bamboo doesn’t generate correct FishEye links for Perforce
BAM-2756  Remote Agent detects JRE as JDK
BAM-2845  Failure message not displayed for junit 4 assertEquals test results
BAM-3319  Unable to configure Bamboo Web Repository URL to represent a subset of the underlying repository for FishEye
BAM-3430  Web repository URLs are generated incorrectly
BAM-4260  User with "build" permission cannot enable / disable builds
BAM-4379  Resolution of duplicate agent names performs at O^n
BAM-4704  Installer not setting bamboo home
BAM-4805  Build result summary (build duration) chart scale is misleading
BAM-4812  webapp/atlassian-bamboo-agent-elastic-assembly-2.4.0.tar.gz is missing the artifacts needed to customise an ami
BAM-4821  Bamboo gadget xml info does not handle trusted apps
BAM-4830  The disable buttons on the dashboard don’t work, agents re-enable themselves
BAM-4872  Tabs are missing for IE6 in Bamboo 2.4
BAM-4888  Maven 2 project with automatic Clover integration can’t find Clover
BAM-5320  Clover integration screenshots are out of date
BAM-4402  NPE on upgrade task 1502
BAM-4545  Can’t cancel from login page
BAM-4556  IE Takes a long time to render the checkbox tree
BAM-4557  First chart gadget added works, subsequent ones don’t
BAM-4638  Clover 1-Click integration for Ant is broken on Windows
BAM-4640  Updating repository polling time doesn’t work
BAM-4683  Pressing enter on the plan wizard takes you to the previous page
BAM-4719  Bamboo deadlock on RemoteElasticInstance
BAM-4755  UI issues with new style dropdowns
BAM-4766  Bamboo gadget update interval is not persisted when re-editing
BAM-4767  Build hover popup styles broken in 2.5-m2 on BEAC
Bamboo 2.5 Upgrade Guide

Supported Platforms
We have made significant changes to our supported platforms (application servers, databases, browsers, etc) in this release. Please read the Supported Platforms page for the full list of supported platforms for Bamboo.

Upgrading from Bamboo 2.4 to 2.5

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 2.5 before upgrading.

Please also note the following important points:

1. **Developing for Bamboo 2.5**

   **New Method for Repository Interface**

   Please note, a new method has been added to the repository interface in line with web repository changes for 2.5. If you have extended the AbstractRepository class, this change will not affect you. However, if you have not extended the AbstractRepository class you will need to implement the new method before your code will compile.

2. **Remote API Support automatically enabled**

   Remote API support for your Bamboo instance will be automatically enabled when you upgrade to Bamboo 2.5. If this is a security concern, you can **disable remote API support** via your administration console, however Bamboo gadgets may not work correctly.

3. **Database Changes**

   Please note that during the upgrade, Bamboo will automatically remove the table BUILD_ASSOCIATION and the table PLAN_DEPENDENCIES will be added. No user intervention is required. However, please ensure that Bamboo has the appropriate access to your database before the upgrade tasks are run (i.e. when you start Bamboo).
4. Pre/Post Build Command Plugin problems

The pre/post build command plugin (v2.4 and earlier) currently does not work with Bamboo 2.5 and will prevent you from creating new plans. If you are using this plugin, we recommend that you either disable it or wait for a new fixed version of the plugin to be released before upgrading to Bamboo 2.5.

5. "Unsupported Databases" is no longer a selectable option in Setup Wizard

The Setup Wizard no longer offers "Unsupported Database" as a selectable option when choosing to connect to an external database. If you are using an unsupported database, you will need to set the following system property before starting your upgraded Bamboo server to enable "Unsupported Database" as a selectable option in the Setup Wizard:

-Dbamboo.enable.unsupported.db=true

Upgrading from Bamboo prior to 2.4

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 2.5.

Please ensure that you read the Bamboo 2.0 Upgrade Guide which contains important upgrade instructions for upgrading from earlier versions of Bamboo.

Bamboo 2.5.5 Release Notes

4 May 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.5.5. This point release is a highly recommended upgrade as it contains important fixes to security vulnerabilities in Bamboo (listed below). For more information about these security vulnerabilities, please refer to the Bamboo Security Advisory 2010-05-04.

Please also refer to the Bamboo 2.5.5 Upgrade Guide for important changes in Bamboo, which are designed to minimise the risk of security attacks.

Bamboo 2.5.5 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.5 yet?

Take a look at all the new features in the Bamboo 2.5 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.5.5 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.5.5 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (6 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>BAM-5400</td>
</tr>
<tr>
<td>BAM-5308</td>
</tr>
<tr>
<td>BAM-5714</td>
</tr>
</tbody>
</table>
Bamboo 2.5.5 Upgrade Guide

Upgrade Notes

A few changes to Bamboo's behaviour have resulted as a consequence of some important fixes to security vulnerabilities in Bamboo 2.5.5. For more information about these security vulnerabilities and their fixes, please refer to the Bamboo Security Advisory 2010-05-04.

Setting File Paths in Bamboo

When modifying Bamboo's 'File Path' option on the Export or Import administration pages or the 'Backup Path' option on the Scheduled Backup page, you can only change the name of files associated with these options (not the the actual file path component itself). To change these file path components, you must explicitly run Bamboo with the following system property:

```
bamboo.paths.set.allowed=true
```

Please refer to Configuring System Properties for details on how to run Bamboo with system properties.

Brute Force Attack Prevention

By default, if you attempt to log in to Bamboo three times unsuccessfully, then for subsequent login attempts, Bamboo will require you recognise a distorted picture of a word and type that word into a text field. For more information, please refer to Enabling or Disabling Captcha for Failed Logins.

HttpOnly Session ID Cookies

In Bamboo Standalone distributions session ID cookies now use the HttpOnly flag by default. This makes it more difficult for malicious (JavaScript) code on a client's browser to gain access to these session ID cookies, thereby minimising the risk of common XSS attacks.

If you are running the Bamboo EAR-WAR distribution, then to minimise the risk of common XSS attacks, we strongly recommend that you configure the application server (Tomcat) running Bamboo to transmit session ID cookies using the HttpOnly flag. Please refer to Configuring Tomcat to Use HttpOnly Session ID Cookies for more information.

Upgrading from Bamboo 2.5.3 to 2.5.5

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.5.3 to 2.5.5.

Upgrading from Bamboo 2.4.x or earlier

In addition to the above, please read the Bamboo 2.5 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 2.5.2 Release Notes

24 February 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.5.2.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.5.2 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.5 yet?

Take a look at all the new features in the Bamboo 2.5 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.5.2 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (10 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-65</td>
</tr>
<tr>
<td>Allows CVS repo to</td>
</tr>
<tr>
<td>timeout and report on</td>
</tr>
<tr>
<td>locking issues</td>
</tr>
<tr>
<td>Assignee: Mark</td>
</tr>
<tr>
<td>Chaimungkalanont</td>
</tr>
<tr>
<td>[Atlassian]</td>
</tr>
<tr>
<td>Reporter: Mark</td>
</tr>
<tr>
<td>Chaimungkalanont</td>
</tr>
<tr>
<td>[Atlassian]</td>
</tr>
<tr>
<td>Priority:</td>
</tr>
<tr>
<td>Status: Resolved</td>
</tr>
<tr>
<td>Resolution: Fixed</td>
</tr>
<tr>
<td>Created: Jul 11, 2006</td>
</tr>
<tr>
<td>Updated: Feb 11, 2010</td>
</tr>
</tbody>
</table>
**Bamboo 2.5.2 Upgrade Guide**

**Upgrading from Bamboo 2.5.1 to 2.5.2**

Please follow the Bamboo Generic Upgrade Guide.

*No additional upgrade tasks are required to upgrade from Bamboo 2.5.1 to 2.5.2.*

**Upgrading from Bamboo 2.4.x or earlier**

In addition to the above, please read the Bamboo 2.5 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 2.5.1 Release Notes**

**28 January 2010**

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.5.1.

We've fixed several bugs in this release. Please see the ‘Updates and Fixes in this Release’ section below for details.

Bamboo 2.5.1 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.5 yet?**

Take a look at all the new features in the Bamboo 2.5 Release Notes and see what you are missing out on!

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.5.1 Upgrade Guide.

**Updates and Fixes in this Release**

<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-5247</td>
</tr>
<tr>
<td>BAM-3463</td>
</tr>
<tr>
<td>BAM-5184</td>
</tr>
<tr>
<td>BAM-3455</td>
</tr>
</tbody>
</table>
Bamboo 2.5.1 Upgrade Guide

Upgrading from Bamboo 2.5 to 2.5.1

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.5 to 2.5.1.

Upgrading from Bamboo 2.4.x or earlier

In addition to the above, please read the Bamboo 2.5 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 2.5.3 Release Notes

18 March 2010

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.5.3.

We've fixed several bugs in this release. Please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.5.3 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.5 yet?

Take a look at all the new features in the Bamboo 2.5 Release Notes and see what you are missing out on!

Bamboo 2.5.3 Upgrade Guide

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.5.3 Upgrade Guide.

Updates and Fixes in this Release

The issues addressed in Bamboo 2.5.3 are shown below. To view the list in JIRA, please refer to our main JIRA site.

<table>
<thead>
<tr>
<th>JIRA Issues (10 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>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-5354</td>
<td></td>
<td>BAM-5354</td>
<td>cannot create plan when plan permissions configuration contains user or group name with whitespace</td>
<td>Krystian Brazulewicz [Atlassian]</td>
<td>Ulrich Kuhnhardt [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 18, 2010</td>
<td>Mar 08, 2010</td>
</tr>
<tr>
<td>BAM-5218</td>
<td></td>
<td>BAM-5218</td>
<td>When downsizing a license plans users are directed to delete on the <em>old</em> server</td>
<td>Krystian Brazulewicz [Atlassian]</td>
<td>Ulrich Kuhnhardt [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jan 14, 2010</td>
<td>Feb 1, 2010</td>
</tr>
<tr>
<td>BAM-5349</td>
<td></td>
<td>BAM-5349</td>
<td>Redirection error when browsing to Bamboo pages when licence is expired</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Nick Chistyakov</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 18, 2010</td>
<td>Jul 14, 2010</td>
</tr>
<tr>
<td>BAM-5340</td>
<td></td>
<td>BAM-5340</td>
<td>bamboo post build labeller to use metadata such as ${bamboo.custom.svn.revision.number} as labels.</td>
<td>Ulrich Kuhnhardt [Atlassian]</td>
<td>Ulrich Kuhnhardt [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 16, 2010</td>
<td>Feb 1, 2010</td>
</tr>
</tbody>
</table>

Bamboo 2.5.3 Upgrade Guide

Upgrading from Bamboo 2.5.2 to 2.5.3

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.5.2 to 2.5.3.

Upgrading from Bamboo 2.4.x or earlier
In addition to the above, please read the Bamboo 2.5 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 2.4 Release Notes**

**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:**

- Bamboo Gadgets in JIRA
- Clover Enhancements
- REST Improvements
- Runtime Log4j Configuration
- Plus over 20 fixes and improvements

**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.*

Please keep logging your votes and issues. They help us decide what needs doing!

---

**Highlights of Bamboo 2.4**

**1**

**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. 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.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<queue expand="true">
  <triggerReason>Manual build</triggerReason>
  <changes expand="true" size="1" max-result="10" start-index="0"/>
  <changes expand="true" size="1" max-result="10" start-index="0"/>
</queue>
```

Runtime Log4j Configuration

We’re a 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

<table>
<thead>
<tr>
<th>Package</th>
<th>Current Level</th>
<th>New Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.atlassian.bamboo.agent</td>
<td>DEBUG</td>
<td>ALL</td>
</tr>
<tr>
<td>com.atlassian.bamboo.buildqueue</td>
<td>DEBUG</td>
<td>ALL</td>
</tr>
<tr>
<td>com.atlassian.bamboo.user.BambooUserManagerImpl</td>
<td>WARN</td>
<td>ALL</td>
</tr>
<tr>
<td>com.opensymphony.xwork.util.LocalizedTextUtil</td>
<td>ERROR</td>
<td>ALL</td>
</tr>
</tbody>
</table>

5

Plus over 20 fixes and improvements

<table>
<thead>
<tr>
<th>JIRA Issues (28 issues)</th>
<th>Type</th>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-3608</td>
<td>Bamboo REST API should expose functionality to retrieve all comments for selected build</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4293</td>
<td>REST API - Get current (running) build details</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-549</td>
<td>Add a portlet to the Jira-Bamboo plugin (possibly Confluence also)</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2867</td>
<td>Be able to reference more than one Bamboo from JIRA</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4335</td>
<td>UniqueAuthorNameValidator throws NPE when repository alias is added and table author contains {{null}} entries</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4349</td>
<td>Sort the view capabilities page lexicographically</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4492</td>
<td>Update BuildResultService in REST documentation</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4513</td>
<td>Prevent Jira Bamboo Plugin from adding gadgets to browser if they dont exist</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4573</td>
<td>phuunit builder should (optionally) only check for a return code or make the number of lines for 'OK' message flexible</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4407</td>
<td>REST - Hibernate exception on build details load</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4384</td>
<td>Unable to view Plan in REST</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-4517</td>
<td>svnkit authentication fails due to authentication file access problems to local cache on all platforms</td>
<td></td>
<td></td>
<td></td>
<td>Resolved</td>
</tr>
</tbody>
</table>
BAM-4013  Favourite toggle broken in all plans screen
Resolved

BAM-4241  rest/api/latest/chart should return useful information
Resolved

BAM-4360  Ensure that all commands when mounting EBS volumes handle failure recursively
Resolved

BAM-4376  Queue re-ordering doesn't work in Safari 4
Resolved

BAM-4342  Plugins 2 xwork plugins throw ClassNotFoundException
Resolved

BAM-4350  Maven 2.1 will not respect path if not already the default M2_HOME
Resolved

BAM-4411  Build reorder section does not show up on the BuildQueue screen
Resolved

BAM-4523  OAuth page looks like crap
Resolved

BAM-4542  Connection from remote agent fails if the agent time is in the past
Resolved

BAM-4543  No cancel link from login page
Resolved

BAM-4550  Legacy clover tab should not show if there is no data
Resolved

BAM-4551  Plan is limited to 15 (Critical)
Resolved

BAM-4553  JIRA Plugin "Latest plan status" link doesn't work (seems to just reload the page) in FireFox 3
Resolved

BAM-4558  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.
Resolved

BAM-4562  Viewing REST pages on IE8 actually crashes IE8. this only seems to happen when you are not logged in
Resolved

BAM-4568  Bamboo Gadgets/Rest Do not accomodate for when the bamboo instance has no plans.
Resolved

Bamboo 2.4 Upgrade Guide

Bamboo 2.4.x does not run on JBoss 4.2.3 or later
We are aware of a JBoss issue that currently prevents Bamboo 2.4.x from running on JBoss 4.2.3 or later. If you are using JBoss 4.2.3 or later, we recommend that you do not upgrade your Bamboo installation until a fix has been implemented. Please see BAM-4705 for more information.

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

Please note, due to some once-off versioning changes for Bamboo 2.4 (the major release only), Bamboo 2.4 should be considered to be Bamboo 2.4.0 for development purposes. Hence, if you developing plugins for Bamboo 2.4 (the major release only), you will need to develop
against version 2.4.0 of the Bamboo artifacts.

---

**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 2.4.3 Release Notes**

**9 December 2009**

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.4.3.

We've fixed a major IE6 bug in this release, please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.4.3 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.4 yet?**

Take a look at all the new features in the Bamboo 2.4 Release Notes and see what you are missing out on!

---

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.4.3 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>
<th>Updated</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-4872</td>
<td>Tabs are missing for IE6 in Bamboo 2.4</td>
<td>Mark Chaimungkaianont [Atlassian]</td>
<td>Edwin Wong [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Nov 19, 2009</td>
<td>Dec 01, 2009</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bamboo 2.4.3 Upgrade Guide**

**Upgrading from Bamboo 2.4.2 to 2.4.3**

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.4.2 to 2.4.3.

**Upgrading from Bamboo 2.3.x or earlier**

In addition to the above, please read the Bamboo 2.4 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 2.4.2 Release Notes**

**25 November 2009**

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.4.2.

We've fixed a number of bugs in this release, please see the 'Updates and Fixes in this Release' section below for details.

Bamboo 2.4.2 is of course free to all customers with active Bamboo software maintenance.

**Don't have Bamboo 2.4 yet?**

Take a look at all the new features in the Bamboo 2.4 Release Notes and see what you are missing out on!
Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.4.1 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (4 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-4826</td>
</tr>
<tr>
<td>BAM-4128</td>
</tr>
</tbody>
</table>

Bamboo 2.4.2 Upgrade Guide

**Upgrading from Bamboo 2.4.1 to 2.4.2**

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.4.1 to 2.4.2.

**Upgrading from Bamboo 2.3.x or earlier**

In addition to the above, please read the Bamboo 2.4 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 2.4.1 Release Notes

10 November 2009

The Atlassian Bamboo team is proud to announce the release of Bamboo 2.4.1.

We've fixed a number of significant bugs in this release, including a HTTP Content-Type bug preventing browsing of REST methods in IE7 (see BAM-4533) and a bug preventing custom elastic images from being created (see BAM-4812).

Bamboo 2.4.1 is of course free to all customers with active Bamboo software maintenance.

Don't have Bamboo 2.4 yet?

Take a look at all the new features in the Bamboo 2.4 Release Notes and see what you are missing out on!

Upgrading from a Previous Version of Bamboo

If you are upgrading, please read the Bamboo 2.4.1 Upgrade Guide.

Updates and Fixes in this Release

<table>
<thead>
<tr>
<th>JIRA Issues (7 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>
Bamboo 2.4.1 Upgrade Guide

Upgrading from Bamboo 2.4 to 2.4.1

Please follow the Bamboo Generic Upgrade Guide. No additional upgrade tasks are required to upgrade from Bamboo 2.4 to 2.4.1.

Upgrading from Bamboo 2.3.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 Release Notes

Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don’t have Bamboo 3.0? 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:

🌟 over 36 new features and improvements implemented
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

1

Dependency Blocking Strategies

Bamboo 2.3 gives you greater control over your builds with the introduction of dependency blocking. Dependency blocking is an 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.

![](dependency_blocking_diagram.png)
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. Prioritise 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 Jobs in the Build Queue and Configuring Notifications for a Plan and its Jobs.

Atlassian Bamboo

![Agents and Build Queue interface]

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.

Create Elastic Image Configuration

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.

### 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, 11:00:00 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>

---

**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.

### Builder Configuration

- **Builder:**
  - PHPUnit
- **Arguments:**
  - Arguments passed to the PHPUnit execute each time the source code changes.
- **System Environment Variables:**
  - (Optional) Any extra environment variables you want to pass to your build, e.g., `MAVEN_OPTS=-Xmx256m -Xms128m`.
- **Working Sub Directory:**
  - (Optional) Bamboo assumes that the build root directory is the working directory. Use this option to specify an alternative working directory (must be a subdirectory of the root directory).

### Where shall PHPUnit store test result file

- Log test execution in XML format to file

### Where shall PHPUnit store code coverage data in Clover XML format

- Write code coverage data in Clover XML format

### Clover Code Coverage File:

- Relative path to file where PHPUnit will store code coverage data in Clover XML format (option `--coverage-clover`).

### Where shall PHPUnit store code coverage data in HTML format

- Generate code coverage report in HTML format

---

**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

### JIRA Issues (89 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-3836</td>
<td>Allow users to easily use ActiveMQ JMX integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-3635</td>
<td>Support multiple AMIs in Bamboo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-3634</td>
<td>Alert users of builds which can not be built by any agents</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-3614</td>
<td>Upgrade plugin system to 2.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-3612</td>
<td>Talkback from elastic agent to bamboo server to include EBS volume mount results</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-3287</td>
<td>Create Php Builder Plugin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2683</td>
<td>Configurable log4j.properties for the remote agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-1677</td>
<td>Servlet plugin point</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-1182</td>
<td>Bulk editing of plans</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4327</td>
<td>Plugin-able top level navigation and footers</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4326</td>
<td>Automatically control starting and stopping of elastic instances by a schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4324</td>
<td>Show success rate on a particular build agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4253</td>
<td>Automatic SVN error recover should be case insensitive</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4195</td>
<td>Allow unit tests to run w/o the need for an ANT_HOME to be set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4133</td>
<td>Improve Build Results load times for bamboo-user group</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-4093</td>
<td>Add Elastic Instance hostname/IP address to REST API for listRunningInstances.action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4087</td>
<td>Improve error handling and reporting for EC2 ebs snapshot / volume procedures (scripts)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4084</td>
<td>Bamboo should figure out if EBS is supposed to be attached and if not, and notifies user.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4081</td>
<td>Customise the fedora base ami so that updates have already been applied</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3890</td>
<td>Notification when a build has been in the queue for x minutes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3766</td>
<td>Upgrade default AMI OS to newer version of Fedora Linux - see reduced scope of this issue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3710</td>
<td>REST API returns only one VCS revision for the build, even if multiple changesets are contained in the build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3697</td>
<td>Upgrade Bamboo to Bonnie 3.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1844</td>
<td>Implement intelligent dependency checking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1412</td>
<td>Dependencies should check parents for modifications prior to build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1393</td>
<td>Checking parent projects for changes before build</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-932</td>
<td>allow a build to be placed at the head of the build queue... (or edit the queue order)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-781</td>
<td>Support ordering of build dependencies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-743</td>
<td>For dependent builds, check parent is not running or queued before checkout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-734</td>
<td>&quot;Block&quot; dependent builds when a parent build is running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-560</td>
<td>Dependencies should handle more complex cases</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4330</td>
<td>Move to top of queue button</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4325</td>
<td>Ability to specify availability zones for Elastic Agents</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3958</td>
<td>Allow build expiry time to be configured</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3788</td>
<td>Everywhere: Sometimes we use the trash can icon, at other times we use the word &quot;Delete&quot;.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3709</td>
<td>Builder dropdown should be sorted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3156</td>
<td>Bamboo should handle &quot;No page associated with this URI&quot; in a more graceful way.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2503</td>
<td>Add ability to reorder the build queue(s) on the fly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3784</td>
<td>Create New Plan &gt; Permissions: Permission Types box bullet points are too far indented to the left.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-3781</td>
<td>Create Plan &gt; Artifacts: &quot;save&quot; and &quot;cancel&quot; action links have inconsistent case.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4257</td>
<td>New REST API disabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4289</td>
<td>Quartz jobs randomly stops</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-4264</td>
<td>Notifications tab when creating plan ui is very screwed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BAM-4263  Mac OSX installer doesn't ask for or set Bamboo Home property  
BAM-4259  Elastic agent history -> delete causes HTTP 404 error: duplicated bamboo token in returnURL  
BAM-4095  API: I need a field with info that build is currently in "building" state.  
BAM-3674  Failing Since Wrong Build #  
BAM-3650  Shutting down my Elastic Image doesn't delete my ebs volume  
BAM-3643  Many many open activeMQ threads consume 800% cpu on beac  
BAM-3495  Build says it is currently in queue, but does not appear IN the queue  
BAM-4297  startElasticInstances REST method should use configuration name instead of id  
BAM-4292  Maven builds importing bamboo web get atlassian-core 3.6  
BAM-4288  Elastic Bamoo Agents do not detect failed EBS mounts  
BAM-4262  UpdateCvsModuleBulkAction doesn't have an empty constructor  
BAM-4256  Duplicate notifications not being detected in oracle  
BAM-4244  Upgrade 2.2.4 -> 2.3 snapshot fails on Upgrade task 1310  
BAM-4235  Internal Server Error while saving PHPUnit build with disabled "Log test execution to XML file"  
BAM-4180  Link to viewImages on the builders page not correct  
BAM-4161  NullPointerException for BuildQueueMonitor  
BAM-4142  Revert Google Collections change  
BAM-4121  Elastic schedules may clash  
BAM-4119  Enable / disable agents don't work from manage instances screen  
BAM-4071  During setup exception java.lang.NoSuchMethodException: com.atlassian.bamboo.ww2.actions.admin.elastic.ConfigureElasticImageConfiguration.<init>() is thrown  
BAM-4069  Bamboo is still reporting false positives with SUCCESS_MESSAGE_LINES = 3000  
BAM-4055  /api/rest/getRecentlyCompletedBuildResultsForBuild.action gives empty output  
BAM-4016  IntrospectionException thrown due to interface method mismatch  
BAM-3971  Improve command interface to allow better variable substitution  
BAM-3839  "Remove All Artifacts" link should require confirmation  
BAM-3819  Exceptions in logs when accessing REST method for build without build result  
BAM-3770  guessChangeSetId returning null will cause emails to throw exceptions
BAM-3757  After X Failed Notifications "Updated By" is broken.
BAM-3722  AuthorName is not html escaped when serving Ajax responses
BAM-3716  JIRA Bamboo plugin throwing NullPointerException for Confluence
BAM-3703  Wrong defined M2_HOME for elastic bamboo image
BAM-3689  Elastic agents do not automatically shut down if the Bamboo server is restarted
BAM-3673  Expire builds throws SQLIntegrityConstraintViolationException
BAM-3664  Build results REST XML invalid
BAM-3523  Elastic agent not unregistered when instance is being killed manually
BAM-3121  Changing the way the "Updated by" text field is being constructed
BAM-2349  Stop Bamboo from running persistence upgrades if license is expired
BAM-2184  Disallow directory listing in Jetty.
BAM-1469  Disabled agents are reenabled on restart
BAM-4168  Web repository module can be added without a web repository url
BAM-3660  Builds that took less than 1 sec to execute should be reported as 'less than a second' instead of 'Unknown'
BAM-3242  Entering incorrect project key gives big stack trace
BAM-2965  Missing custom build data when examined from a CustomBuildCompleteAction
BAM-3787  View Plan > Tests: "Most Recent" column hard to understand. Most Recent What?
BAM-3780  User Profile > Edit: "Jabber Address" label isn't clear.
BAM-3575  Weird Panelling on General Configuration Updated

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
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 2.3.1 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 (3 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 2.3.1 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](#).
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.

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

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.
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
BAM-3607 Builds should be able to have a configurable timeout threshold

BAM-3605 CLONE - The LATEST URL redirect(s) only apply to top level artifacts.

BAM-3591 Initial logging level of Remote Agent is DEBUG

BAM-3561 Bamboo uses db column name "resource", this is reserved in Oracle

BAM-3554 inconsistent slashing on View Instance page

BAM-3553 Elastic Bamboo Configuration administration tab should be always visible

BAM-3521 Agent Matrix should limited to active plans & agents

BAM-3509 Restarting Remote Agent in wrapper causes license issues with 1 Remote Agent licenses

BAM-3508 "Can't open file" error causes build to not be run

BAM-3504 Plan Requirements Configuration page showing incorrect coloration and heading for images

BAM-3480 Remote Agent creates spurious directory tree under the default bamboo home bin/ directory

BAM-3479 Refactor VariableSubstitutionBean's so that their internal bamboo variables can be accessed by other components

BAM-3462 Edit shared capabilities broken

BAM-3458 Artifact copier fails to copy any artifacts if a target directory is missing

BAM-3457 Make instance type of EC2instance available.

BAM-3445 SVN UpdateEventHandler throws NPE

BAM-3432 Shutdown all elastic agent fails

BAM-3412 Change to SVN URL through global variables not detected

BAM-3408 Evaluation Expiry message for all products

BAM-3386 Cannot test IM notifications in Edit mode for non @talk.google.com accounts

BAM-3341 All capability sets are imported as LocalCapabilitySets regardless if they are remote or local

BAM-3339 Elastic Bamboo implementation, M4

BAM-3335 Elastic Bamboo implementation M3

BAM-3329 Bamboo ships with out-of-date version of ehcache settings

BAM-3292 Global Variables list in Administration panel is sorted randomly

BAM-3281 Previous button on page '5. Artifacts' goes to '6. Notifications' instead of '4. Requirements'

BAM-3280 Allow users to use the bootstrap jar directly
BAM-3250  API for retrieving build artifacts

BAM-3246  Remote build logging can slow down a build

BAM-3243  Export fails with "Adding text to an XML document must not be null"

BAM-3240  Local cvs repositories are not supported

BAM-3220  Can not test IM client while in edit mode

BAM-3194  Build Hang Prediction shows no logs for ‘299’ minutes, even when messages are coming through.

BAM-3189  Tests summary screen always shows last 25 builds

BAM-3183  Test results directory can't be updated

BAM-3179  Ability to delete comments

BAM-3177  Artifact editing/viewing screens are inconsistent in the order of the columns

BAM-3176  Latest status of a build may clear comment form

BAM-3172  Allow underscore '_' character in Global Variable name

BAM-3168  Ability to Abandon a build

BAM-3166  Maven download logs should be filtered

BAM-3155  Build Configuration | Builder | Test Result Directory not persisted for Bash (Command type)

BAM-3141  Typo in LocalBuildResultProcessor 'Ignoriing'

BAM-3138  Scheduled builds and builds running on remote agents don't seem to switch to a new source root in the build directory

BAM-3125  Allow 'wget' unix utility to access plan's artifacts by providing username and password

BAM-3122  Export of custom data is not null safe

BAM-3120  Ability to download artifact via the REST API

BAM-3108  User Picker for build permissions lets you 2 when you can only add 1

BAM-3105  Include failure details into Bamboo mail notification

BAM-3104  Send Bamboo messages as HTML formatted mails

BAM-3091  Incorrect error for editProfile when not logged in

BAM-3090  Access artifacts from the standard "pretty" URL

BAM-3089  Build results summary page relies on specific punctuation in trigger reason.

BAM-3083  Bamboo email notifications should include the unit tests that failed

Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
Resolved
BAM-3043  Change logging level of AccessLogFilter from INFO to DEBUG

BAM-2994  The LATEST URL redirect(s) only apply to top level artifacts.

BAM-2991  SVN URL change is not picked up by all the agent.

BAM-2989  Artifact collections with large numbers of small files take forever to copy

BAM-2983  The remote agent dies if a (remote) build is stopped while artifacts are being transferred.

BAM-2951  Add plugin point so that Agent/Build matching can be customised

BAM-2946  Bamboo should display source repository revision used for particular build.

BAM-2933  Ability to substitute to existing system variables for the System Environment field

BAM-2852  Improve log transfer between Bamboo agent and build server

BAM-2835  Bamboo should look for the BUILD FAILED message along with the BUILD SUCCESSFUL message to determine build outcome.

BAM-2803  Reduced log spam from remote agents

BAM-2721  Artifacts should not be copied if builds fail.

BAM-2713  Remove Errors From Dashboard

BAM-2664  Transferring artifacts of large size require too much CPU resources

BAM-2612  Make latest artifact link an HTTP redirect instead of a meta-refresh browser redirect

BAM-2606  Global Variables should be available in the Custom Data Map for a build

BAM-2605  Bamboo doesn't trim spaces, when entering email addresses in the build notifications screen

BAM-2560  Add links to order form on pages that notify customers their maintenance has expired

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

BAM-1413  Support for MSBuild

BAM-1403  Add more detail to the broken build e-mail - compiler error or broken test

BAM-1396  Have a time limit on builds

BAM-1394  Add "Build Actions" menu to Build Results page

BAM-1375  Labeller plugin assumes that build log contains instances of SimpleLogEntry rather than LogEntry

BAM-1360  Bamboo should explicitly build projects when user triggers a build ON manual build strategy

BAM-1355  Ability to remove Mail Server of IM Server Configuration

BAM-1299  Improve the Notification Framework in Bamboo, to register listeners

BAM-1177  Run builders on Amazon cloud

BAM-1107  Broadcast IM comments
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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Bam-3555</td>
</tr>
<tr>
<td>Bam-4055</td>
</tr>
<tr>
<td>Bam-4057</td>
</tr>
<tr>
<td>Bam-4087</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide.](#)

*Don't have Bamboo 3.0? 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!

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>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-4006</td>
<td>JNA</td>
<td>BAM-4006</td>
<td>JNA version update</td>
<td>Brydie McCoy</td>
<td>Mark Chaimungkalanong</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Jun 01, 2009</td>
<td>Jun 02, 2009</td>
</tr>
<tr>
<td>BAM-3957</td>
<td>BAM-3957</td>
<td>Tomcat &gt;5.5.25 does not allow &quot;=&quot; signs in cookies</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>May 21, 2009</td>
<td>May 26, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3949</td>
<td>BAM-3949</td>
<td>next build result data content is not updated for scheduled builds</td>
<td>Brydie McCoy</td>
<td>Ulrich Kuhnhardt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>May 20, 2009</td>
<td>May 31, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3919</td>
<td>BAM-3919</td>
<td>remote agent wrapper &lt;start&gt; command overrides remote agent/wrapper configuration</td>
<td>Unassigned</td>
<td>Ulrich Kuhnhardt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>May 18, 2009</td>
<td>Jun 09, 2010</td>
<td></td>
</tr>
<tr>
<td>BAM-3882</td>
<td>BAM-3882</td>
<td>Bamboo removes quotes (\”) from maven goal settings</td>
<td>Brydie McCoy</td>
<td>Ulrich Kuhnhardt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>May 05, 2009</td>
<td>May 28, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3861</td>
<td>BAM-3861</td>
<td>REST API not correctly escaping '&lt;' characters in the author attribute of a commit</td>
<td>Brydie McCoy</td>
<td>Ulrich Kuhnhardt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>May 04, 2009</td>
<td>May 21, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3850</td>
<td>BAM-3850</td>
<td>Freemarker type mismatch on build queue</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Apr 27, 2009</td>
<td>May 26, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3833</td>
<td>BAM-3833</td>
<td>Perforce configuration doesn't use system variables during validation for client and port</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Apr 22, 2009</td>
<td>Jun 02, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3816</td>
<td>BAM-3816</td>
<td>Regression: ConglomerateCookieValue on Dashboard</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Apr 20, 2009</td>
<td>May 26, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3688</td>
<td>BAM-3688</td>
<td>Upgrading to 2.2.1 with expired license causes exceptions</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Mar 24, 2009</td>
<td>May 27, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3332</td>
<td>BAM-3332</td>
<td>Plan list collapse/expand is not persisted on tomcat version &gt; 5.5.25</td>
<td>Brydie McCoy</td>
<td>Ulrich Kuhnhardt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Dec 10, 2008</td>
<td>Jun 08, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-3255</td>
<td>BAM-3255</td>
<td>Bamboo triggered SVN update fails with NullPointerException (upgrade svnkit, please)</td>
<td>Krystian Brazuliewicz</td>
<td>Marcel May</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Nov 30, 2008</td>
<td>Jun 02, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-2758</td>
<td>BAM-2758</td>
<td>Remote Agent detects previously deleted capabilities after restart</td>
<td>Brydie McCoy</td>
<td>James Bunt</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Jun 12, 2008</td>
<td>May 26, 2009</td>
<td></td>
</tr>
<tr>
<td>BAM-1938</td>
<td>BAM-1938</td>
<td>Bamboo fails to perform wildcard matches, when searching viewUsers page if the text is wrapped around a &quot;&quot; wildcard character</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>Fixed</td>
<td></td>
<td></td>
<td>Nov 15, 2007</td>
<td>May 27, 2009</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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 [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 (10 Issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><strong>BAM-3723</strong></td>
</tr>
<tr>
<td><strong>BAM-3643</strong></td>
</tr>
<tr>
<td><strong>BAM-3121</strong></td>
</tr>
<tr>
<td><strong>BAM-2965</strong></td>
</tr>
</tbody>
</table>
CustomBuildCompleteAction

<table>
<thead>
<tr>
<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>
</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide.**

Don't have Bamboo 3.0? 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!

**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 (12 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>Resolved</td>
<td>Fixed</td>
<td>Feb 25, 2009</td>
<td>Feb 25, 2009</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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 Atlassian's 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!
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.

- Read more about linking issues to builds.

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>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Summary</td>
<td></td>
</tr>
<tr>
<td>BAM-3293</td>
<td>Bamboo Upgrade Guide may call for reconfiguring external user repositories in wrong order</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-3055</td>
<td>External User Management is marked as readonly but it is still attempting to write to Crowd.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2992</td>
<td>Add the agent information of the agent executing the build to the Build Context.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2974</td>
<td>A Version Tab Panel in JIRA Bamboo plugin for &quot;builds&quot;</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2973</td>
<td>A Project Tab Panel in JIRA Bamboo plugin for &quot;builds&quot;</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2972</td>
<td>Plan Summary Tab for issues built in plan</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-2971</td>
<td>Mark which builds have a JIRA issue is fixed in</td>
<td>Resolved</td>
</tr>
</tbody>
</table>
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 doesnt 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.  Resolved

BAM-2171  URL rewrite issue with JIRA's Bamboo plugin  Resolved

BAM-1955  The shipped Bamboo source code assembly is missing the bamboo-ldap-web-app module  Resolved

BAM-1937  REST API Login.action does not work if anonymous user access to plans is disabled.  Resolved

BAM-1791  Indexing Rework  Resolved

BAM-1619  Incorrect link generated by JIRA (Bamboo Integration) Portlet  Resolved

BAM-1615  FileNotFoundException due to Bamboo fragmenting indexes in ../<Bamboo-Home>/index  Resolved

BAM-1323  Indexes are slow to reindex  Resolved

BAM-1228  Links to JIRA is potentially confusing  Resolved

BAM-984  Repeated NullPointerExceptions on Shutdown; shutdown fails to complete  Resolved

BAM-954  Changes View -> Clicking on the revision number of the file should lead to an annotated file view as of this revision  Resolved

BAM-953  Changes tab should feature a Changeset link  Resolved

BAM-142  ShutdownHook doesn't work correctly in Tomcat  Resolved

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 `BRS_LINKEDJIRAISSUES`. No user intervention is required.

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
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!

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 (20 issues)</th>
</tr>
</thead>
</table>

<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-3241</td>
<td>BAM-3241</td>
<td>Bamboo 2.1.4 forces SVN working copy to upgrade to 1.5</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Nov 25, 2008</td>
<td>Dec 01, 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 (e.g., if 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 3.0** has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide.

Don't have Bamboo 3.0? 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!

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>Type</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 3.0** has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide.

*Don't have Bamboo 3.0? 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!

---

**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

### JIRA Issues (16 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-3098</td>
<td>Dashboard Ajax update not updating full row</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Oct 01, 2008</td>
<td>Oct 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-3087</td>
<td>Sentence incomplete in Plan Details -&gt; Build Plan Key description</td>
<td>Mark Chaimungkalanont</td>
<td>Boris Capitanu</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Sep 25, 2008</td>
<td>Sep 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-3086</td>
<td>Build agent fails sporadically with UncategorizedJmsException</td>
<td>Mark Chaimungkalanont</td>
<td>Dara Lillis</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Sep 25, 2008</td>
<td>Oct 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-3056</td>
<td>Admins updating a user resets the notification preferences</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Sep 11, 2008</td>
<td>Oct 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-3005</td>
<td>Renaming a custom capability, sets the readonly flag in the plan requirement to true.</td>
<td>Mark Chaimungkalanont</td>
<td>Ajay Sridhar</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Aug 28, 2008</td>
<td>Sep 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-2995</td>
<td>Adding comment containing issue key results in badly formed URL</td>
<td>Mark Chaimungkalanont</td>
<td>Matthew Jensen</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Aug 18, 2008</td>
<td>Sep 20C</td>
</tr>
<tr>
<td></td>
<td>BAM-2986</td>
<td>Freemarker exception</td>
<td>Mark Chaimungkalanont</td>
<td>Ajay Sridhar</td>
<td></td>
<td></td>
<td>Resolved</td>
<td>Aug 13, 2008</td>
<td>Dec</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!

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

JIRA Issues (26 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>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-3042</td>
<td>Valid Broker URLs are ruined by attempt to add &quot;maxinactivityDuration&quot;</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Sep 2008</td>
<td></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 [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 2008</td>
<td></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 Chaimungkalanont [Atlassian]</td>
<td>Edwin Wong [Atlassian]</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 2008</td>
<td></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 Chaimungkalanont [Atlassian]</td>
<td>Mick van der Most van Spijk</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Aug 2008</td>
<td></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 Guminiski [Atlassian]</td>
<td>Jacek Jaroczynski</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 1st 2008</td>
<td></td>
</tr>
<tr>
<td>BAM-2806</td>
<td>&quot;Triggered&quot; is spelt incorrectly</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Stuart Miller</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 2nd 2008</td>
<td></td>
</tr>
<tr>
<td>BAM-2691</td>
<td>Warning when moving plans not updated with 2.0 changes (link + text)</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Kirk Wylie</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 1st 2008</td>
<td></td>
</tr>
<tr>
<td>BAM-2050</td>
<td>The anonymous user signup page doesn't validate user's email</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Ajay Sridhar</td>
<td>Down</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jan 2008</td>
<td></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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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!

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

JIRA Issues (2 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></td>
<td>BAM-2967</td>
<td>Bamboo 2.1 fails upgrade on Oracle database</td>
<td>Ajay Sridhar</td>
<td>Ajay Sridhar</td>
<td></td>
<td>!Resolved</td>
<td>Fixed</td>
<td>Aug 06, 2008</td>
<td>Feb 11, 2010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2885</td>
<td>When a force clean build is triggered, contents of symlinks are also removed</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td></td>
<td>!Resolved</td>
<td>Fixed</td>
<td>Jul 17, 2008</td>
<td>Aug 11, 2008</td>
<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.
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:

🌟 over 70 new features and improvements implemented
🌟 over 240 votes fulfilled

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=Windows XP' 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 for more information.

- A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

Together, capabilities and requirements control which agents can execute builds for particular Jobs. Each Job can only be built by agents whose capabilities meet the Job's requirements. See Configuring a Job's Requirements for more information.

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).
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.

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.

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.

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.

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 `bamboo.jms.broker.uri` 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.
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 Configuring a Job's 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 Configuring a Job's 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 Configuring a Job's 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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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.

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!

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

**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>
<th>Due</th>
</tr>
</thead>
</table>

Perforce sync
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2743</td>
<td>Remote agents may return even when already marked as dead</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 10, 2008</td>
<td>Sep 01, 2009</td>
</tr>
<tr>
<td>BAM-2412</td>
<td>Bamboo export fails if an LDAP user doesn't have a full name defined.</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Mar 27, 2008</td>
<td>Sep 01, 2009</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.

⚠️ 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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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.

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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
</tbody>
</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 upgrade. The complete list of Upgrade Guides is available here.

**Bamboo 2.0.4 Release Notes**
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!

### 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

**JIRA Issues (16 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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-2688</td>
<td>Jabber connection with talk.google.com doesn't work when using port 443</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2687</td>
<td>CVS change detection should be more tolerant of symlinks of CVSROOT</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2678</td>
<td>Perforce occasionally fails to connect to the server</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2676</td>
<td>Nullable pointer when editing IM server and Mail server</td>
<td>Brydie McCoy</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2675</td>
<td>Import fails during setup of Bamboo 2.0.x</td>
<td>Adrian Hempel</td>
<td>Adrian Hempel</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2673</td>
<td>net.sf.hibernate.LazyInitializationException: cannot access loading collection</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2669</td>
<td>Jira Server Password Not Imported Correctly</td>
<td>Brydie McCoy</td>
<td>Brydie McCoy</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2667</td>
<td>Can't edit mail server after upgrading from 2.0.2 - 2.0.3</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2666</td>
<td>Global Variables not substituted in Perforce Repository Configuration</td>
<td>Mark Chaimungkalanont</td>
<td>Damon Kropf-Untucht</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2655</td>
<td>Clicking on the favourite Icon, makes the Bamboo server unresponsive on MSSQL</td>
<td>Mark Chaimungkalanont</td>
<td>Ajay Sridhar</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2646</td>
<td>Customers using the net.sf.hibernate.dialect.MySQLDialect dialect can't upgrade to Bamboo 2.0</td>
<td>Ajay Sridhar</td>
<td>Ajay Sridhar</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2625</td>
<td>javax.crypto.IllegalArgumentException: cannot access loading collection</td>
<td>Mark Chaimungkalanont</td>
<td>Lars Vonk</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2571</td>
<td>Nullable pointer logged when no coverage is parsed from Clover</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td>BAM-2567</td>
<td>1.2.4 CVS's revision key was locale sensitive and may be incorrectly upgraded</td>
<td>Mark Chaimungkalanont</td>
<td>Mark Chaimungkalanont</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Upgrade to 2.0.x fails if user is using a</td>
<td>Brydie McCoy</td>
<td>Ajay Sridhar</td>
<td>My</td>
<td>Resolved</td>
<td>Fixed</td>
</tr>
</tbody>
</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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!

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>
<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-2505</td>
<td></td>
<td></td>
<td>entering a number followed by space in</td>
<td>Brydie McCoy</td>
<td>Joe Xie atlassian</td>
<td>![Resolved]</td>
<td>Fixed</td>
<td>Apr 20, 2008</td>
<td>May 13, 2008</td>
<td></td>
</tr>
</tbody>
</table>
Bamboo 2.0.3 Upgrade Guide

Upgrading from Bamboo 2.0.0.x to 2.0.3

Please follow the Bamboo Generic Upgrade Guide.

No additional upgrade tasks are required to upgrade from Bamboo 2.0.0.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 1.2.0 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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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!

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

<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>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-2544</td>
<td>Multiple Externals will cause an exception on startup after restart</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td></td>
<td></td>
<td>Fixed</td>
<td>May 01, 2008</td>
<td>May 05, 2008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2543</td>
<td>Memory leak caused by subversion client</td>
<td>Brydie McCoy</td>
<td>Edwin Wong</td>
<td></td>
<td></td>
<td>Fixed</td>
<td>May 01, 2008</td>
<td>May 05, 2008</td>
<td></td>
</tr>
</tbody>
</table>
**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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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!

**Upgrading from a Previous Version of Bamboo**

If you are upgrading, please read the Bamboo 2.0.1 Upgrade Guide.

**Updates and Fixes in this Release**

<table>
<thead>
<tr>
<th>JIRA Issues (18 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>BAM-2515</td>
</tr>
</tbody>
</table>
Bamboo 2.0 Upgrade Guide

Upgrading from Bamboo 2.0 to 2.0.1

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

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 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.

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 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.

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 (Please note that there is no 2.0 Beta 7). 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:

   ```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.

![Warning]

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 Configuring a Job's 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 Configuring a Job's 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 Configuring a Job’s 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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? Take a look at the features of Bamboo's latest major version and try it out!

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>
<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-2407</td>
<td>Remote Agent running on Sun JDK 1.6 exits after build is queued for it</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Rex Tener</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mar 27, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ticket</td>
<td>Description</td>
<td>Authors</td>
<td>Resolution Status</td>
<td>Created</td>
<td>Updated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2401</td>
<td>Leave builds in a building state while the agent reports it is idle</td>
<td>Adrian Hempel (Atlassian)</td>
<td>Resolved</td>
<td>Mar 25, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2389</td>
<td>Add more sample plugins to Bamboo Development Kit</td>
<td>Adrian Hempel (Atlassian)</td>
<td>Resolved</td>
<td>Mar 19, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2381</td>
<td>Failing since build for a test reports the wrong result</td>
<td>Brydie McCoy (Atlassian)</td>
<td>Resolved</td>
<td>Mar 17, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2315</td>
<td>LDAP user cache is not refreshed when principals are added to AD/LDAP</td>
<td>Brydie McCoy (Atlassian)</td>
<td>Resolved</td>
<td>Feb 28, 2008</td>
<td>Sep 01, 2009</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2267</td>
<td>Editing capabilities and requirements</td>
<td>Mark Chaimungkalanont (Atlassian)</td>
<td>Resolved</td>
<td>Feb 19, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Adrian Hempel (Atlassian)</td>
<td>Resolved</td>
<td>Feb 13, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2140</td>
<td>A capability screen that allows bulk editing of capabilities and requirements</td>
<td>Mark Chaimungkalanont (Atlassian)</td>
<td>Resolved</td>
<td>Jan 22, 2008</td>
<td>Feb 12, 2010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-2122</td>
<td>Some exceptions during a build don’t cause build to abort</td>
<td>Adrian Hempel (Atlassian)</td>
<td>Resolved</td>
<td>Jan 22, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1862</td>
<td>Move the XFire dependency for Crowd from 1.2.1 to 1.2.6</td>
<td>Brydie McCoy (Atlassian)</td>
<td>Resolved</td>
<td>Oct 24, 2007</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1813</td>
<td>The project level RSS feeds are ordered wrong.</td>
<td>Brydie McCoy (Atlassian)</td>
<td>Resolved</td>
<td>Oct 11, 2007</td>
<td>Apr 01, 2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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.](image)

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**

---

525
Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don’t have Bamboo 3.0? Take a look at the features of Bamboo’s latest major version and try it out!

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 (Please note that there is no 2.0 Beta 7). 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.

<table>
<thead>
<tr>
<th>JIRA Issues (21 issues)</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-2391 SVN externals now prohibitively slow</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2384 Test setting up database with JNDI</td>
<td>Unassigned</td>
<td>None</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2382 Perforce Web Repository Uris are not implemented in 2.0</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Brydie McCoy [Atlassian]</td>
<td>⬇</td>
</tr>
<tr>
<td>BAM-2375 Artifacts copy problems with local/remote agents</td>
<td>Unassigned</td>
<td>None</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2372 Repository web URL still create links even though none is specified</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2362 Can’t edit existing perforce repository plan to a new P4 workspace</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Scott Marshall</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2358 Can’t Find JUnit Results</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Sam Berlin</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2309 NullPointerException in AgentHeartBeat on startup</td>
<td>Edwin Wong [Atlassian]</td>
<td>David O’Flynn [Atlassian]</td>
<td>⬆</td>
</tr>
<tr>
<td>BAM-2283 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>
</tr>
<tr>
<td>BAM-2270 Link incorrectly escaped in Builder JDK error Message</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Edwin Wong [Atlassian]</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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 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.

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 2.0 Beta (but will be supported in the official Bamboo 2.0 release). Please see the Bamboo security advisory for further details. RESOLVED!
- 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 (11 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-2366</td>
</tr>
<tr>
<td>BAM-2363</td>
</tr>
<tr>
<td>BAM-2250</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 3.0** has been released. Read the **Bamboo 3.0 Release Notes and Upgrade Guide**. *Don't have Bamboo 3.0? Take a look at the features of Bamboo's latest major version and try it out!*

12 March 2008

Atlassian is proud to announce the release of 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.

**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](#).

<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>U</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>Resolved</td>
<td>Fixed</td>
<td>Mar 06, 2008</td>
<td>Apr 01, 2008</td>
<td>2</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>Resolved</td>
<td>Fixed</td>
<td>Mar 04, 2008</td>
<td>Apr 01, 2008</td>
<td>2</td>
</tr>
<tr>
<td>🟢</td>
<td>BAM-2325</td>
<td>ScheduleBackupConfiguration exception if backup is</td>
<td>Mark Chaumungkalananont</td>
<td>Aristedes Maniatis</td>
<td>🟢</td>
<td>Fixed</td>
<td></td>
<td>Mar 04, 2008</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
### 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

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!
- 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.

### JIRA Issues (17 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>Dur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BAM-2299</td>
<td>Provide the ability to add a comment to a build result via Bamboo's REST API</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Adrian Hempel [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 27, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2294</td>
<td>Setup wizard fails at first step if IP address cannot be determined</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Adrian Hempel [Atlassian]</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 26, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2248</td>
<td>Unable to delete a build</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Benjamin LERMAN</td>
<td></td>
<td>Resolved</td>
<td>Fixed</td>
<td>Feb 15, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BAM-2200</td>
<td>Bamboo 2.0 beta 1 WAR file not working in</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>MattyJ</td>
<td></td>
<td>Resolved</td>
<td>Not a bug</td>
<td>Feb 08, 2008</td>
<td>Apr 01, 2008</td>
<td></td>
</tr>
</tbody>
</table>
### 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 3.0** has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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.
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](http://jira.atlassian.com).

**JIRA Issues (18 issues)**

<table>
<thead>
<tr>
<th>Key</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>
</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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-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 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 (13 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-2236</td>
</tr>
<tr>
<td>BAM-2231</td>
</tr>
<tr>
<td>BAM-2222</td>
</tr>
<tr>
<td>BAM-2201</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? Take a look at the features of Bamboo's latest major version and try it out!

8 February, 2008

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.

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.

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-2250 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.

<table>
<thead>
<tr>
<th>JIRA Issues (189 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>BAM-3010</td>
</tr>
<tr>
<td>BAM-2609</td>
</tr>
<tr>
<td>BAM-2462</td>
</tr>
<tr>
<td>BAM-2460</td>
</tr>
<tr>
<td>BAM-2448</td>
</tr>
<tr>
<td>BAM-2447</td>
</tr>
<tr>
<td>BAM-2435</td>
</tr>
<tr>
<td>BAM-2431</td>
</tr>
<tr>
<td>BAM-2430</td>
</tr>
<tr>
<td>BAM-2429</td>
</tr>
<tr>
<td>BAM-2425</td>
</tr>
<tr>
<td>BAM-2422</td>
</tr>
<tr>
<td>BAM-2407</td>
</tr>
<tr>
<td>BAM-2405</td>
</tr>
<tr>
<td>BAM-2404</td>
</tr>
<tr>
<td>BAM-2401</td>
</tr>
<tr>
<td>BAM-2391</td>
</tr>
<tr>
<td>BAM-2390</td>
</tr>
<tr>
<td>BAM-2388</td>
</tr>
<tr>
<td>BAM-2383</td>
</tr>
<tr>
<td>BAM-2382</td>
</tr>
<tr>
<td>BAM-2381</td>
</tr>
<tr>
<td>BAM-2372</td>
</tr>
<tr>
<td>BAM-2368</td>
</tr>
<tr>
<td>BAM-2366</td>
</tr>
<tr>
<td>BAM-2360</td>
</tr>
<tr>
<td>BAM-2359</td>
</tr>
<tr>
<td>BAM-2358</td>
</tr>
<tr>
<td>BAM-2355</td>
</tr>
<tr>
<td>BAM-2353</td>
</tr>
<tr>
<td>BAM-2348</td>
</tr>
<tr>
<td>BAM-2346</td>
</tr>
<tr>
<td>BAM-2330</td>
</tr>
<tr>
<td>BAM-2329</td>
</tr>
<tr>
<td>BAM-2327</td>
</tr>
<tr>
<td>BAM-2325</td>
</tr>
<tr>
<td>BAM-2322</td>
</tr>
<tr>
<td>BAM-2321</td>
</tr>
<tr>
<td>BAM-2317</td>
</tr>
<tr>
<td>BAM-2314</td>
</tr>
<tr>
<td>BAM-2312</td>
</tr>
<tr>
<td>BAM-2309</td>
</tr>
<tr>
<td>BAM-2308</td>
</tr>
<tr>
<td>BAM-2299</td>
</tr>
<tr>
<td>BAM-2294</td>
</tr>
<tr>
<td>BAM-2293</td>
</tr>
<tr>
<td>BAM-2291</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>BAM-2290</td>
</tr>
<tr>
<td>BAM-2288</td>
</tr>
<tr>
<td>BAM-2284</td>
</tr>
<tr>
<td>BAM-2283</td>
</tr>
<tr>
<td>BAM-2282</td>
</tr>
<tr>
<td>BAM-2279</td>
</tr>
<tr>
<td>BAM-2278</td>
</tr>
<tr>
<td>BAM-2276</td>
</tr>
<tr>
<td>BAM-2270</td>
</tr>
<tr>
<td>BAM-2267</td>
</tr>
<tr>
<td>BAM-2264</td>
</tr>
<tr>
<td>BAM-2262</td>
</tr>
<tr>
<td>BAM-2260</td>
</tr>
<tr>
<td>BAM-2250</td>
</tr>
<tr>
<td>BAM-2248</td>
</tr>
<tr>
<td>BAM-2246</td>
</tr>
<tr>
<td>BAM-2245</td>
</tr>
<tr>
<td>BAM-2241</td>
</tr>
<tr>
<td>BAM-2240</td>
</tr>
<tr>
<td>BAM-2238</td>
</tr>
<tr>
<td>BAM-2236</td>
</tr>
<tr>
<td>BAM-2235</td>
</tr>
<tr>
<td>BAM-2231</td>
</tr>
<tr>
<td>BAM-2230</td>
</tr>
<tr>
<td>BAM-2229</td>
</tr>
<tr>
<td>BAM-2226</td>
</tr>
<tr>
<td>Issue Number</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>BAM-2225</td>
</tr>
<tr>
<td>BAM-2224</td>
</tr>
<tr>
<td>BAM-2223</td>
</tr>
<tr>
<td>BAM-2222</td>
</tr>
<tr>
<td>BAM-2208</td>
</tr>
<tr>
<td>BAM-2207</td>
</tr>
<tr>
<td>BAM-2206</td>
</tr>
<tr>
<td>BAM-2205</td>
</tr>
<tr>
<td>BAM-2202</td>
</tr>
<tr>
<td>BAM-2201</td>
</tr>
<tr>
<td>BAM-2199</td>
</tr>
<tr>
<td>BAM-2197</td>
</tr>
<tr>
<td>BAM-2196</td>
</tr>
<tr>
<td>BAM-2195</td>
</tr>
<tr>
<td>BAM-2194</td>
</tr>
<tr>
<td>BAM-2193</td>
</tr>
<tr>
<td>BAM-2176</td>
</tr>
<tr>
<td>BAM-2166</td>
</tr>
<tr>
<td>BAM-2152</td>
</tr>
<tr>
<td>BAM-2150</td>
</tr>
<tr>
<td>BAM-2147</td>
</tr>
<tr>
<td>BAM-2146</td>
</tr>
<tr>
<td>BAM-2140</td>
</tr>
<tr>
<td>BAM-2132</td>
</tr>
<tr>
<td>BAM-2129</td>
</tr>
<tr>
<td>BAM-2122</td>
</tr>
<tr>
<td>Issue</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>BAM-2118</td>
</tr>
<tr>
<td>BAM-2117</td>
</tr>
<tr>
<td>BAM-2116</td>
</tr>
<tr>
<td>BAM-2113</td>
</tr>
<tr>
<td>BAM-2104</td>
</tr>
<tr>
<td>BAM-2103</td>
</tr>
<tr>
<td>BAM-2102</td>
</tr>
<tr>
<td>BAM-2100</td>
</tr>
<tr>
<td>BAM-2096</td>
</tr>
<tr>
<td>BAM-2095</td>
</tr>
<tr>
<td>BAM-2077</td>
</tr>
<tr>
<td>BAM-2073</td>
</tr>
<tr>
<td>BAM-2063</td>
</tr>
<tr>
<td>BAM-2057</td>
</tr>
<tr>
<td>BAM-2032</td>
</tr>
<tr>
<td>BAM-2029</td>
</tr>
<tr>
<td>BAM-2017</td>
</tr>
<tr>
<td>BAM-2013</td>
</tr>
<tr>
<td>BAM-2007</td>
</tr>
<tr>
<td>BAM-1996</td>
</tr>
<tr>
<td>BAM-1976</td>
</tr>
<tr>
<td>BAM-1968</td>
</tr>
<tr>
<td>BAM-1967</td>
</tr>
<tr>
<td>BAM-1966</td>
</tr>
<tr>
<td>BAM-1964</td>
</tr>
<tr>
<td>BAM-1944</td>
</tr>
<tr>
<td>Issue Key</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>BAM-1932</td>
</tr>
<tr>
<td>BAM-1897</td>
</tr>
<tr>
<td>BAM-1882</td>
</tr>
<tr>
<td>BAM-1860</td>
</tr>
<tr>
<td>BAM-1847</td>
</tr>
<tr>
<td>BAM-1821</td>
</tr>
<tr>
<td>BAM-1813</td>
</tr>
<tr>
<td>BAM-1807</td>
</tr>
<tr>
<td>BAM-1797</td>
</tr>
<tr>
<td>BAM-1752</td>
</tr>
<tr>
<td>BAM-1740</td>
</tr>
<tr>
<td>BAM-1719</td>
</tr>
<tr>
<td>BAM-1709</td>
</tr>
<tr>
<td>BAM-1696</td>
</tr>
<tr>
<td>BAM-1688</td>
</tr>
<tr>
<td>BAM-1682</td>
</tr>
<tr>
<td>BAM-1658</td>
</tr>
<tr>
<td>BAM-1652</td>
</tr>
<tr>
<td>BAM-1628</td>
</tr>
<tr>
<td>BAM-1618</td>
</tr>
<tr>
<td>BAM-1604</td>
</tr>
<tr>
<td>BAM-1586</td>
</tr>
<tr>
<td>BAM-1585</td>
</tr>
<tr>
<td>BAM-1572</td>
</tr>
<tr>
<td>BAM-1518</td>
</tr>
<tr>
<td>BAM-1504</td>
</tr>
<tr>
<td>BAM-1466</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>BAM-1465</td>
</tr>
<tr>
<td>BAM-1416</td>
</tr>
<tr>
<td>BAM-1401</td>
</tr>
<tr>
<td>BAM-1365</td>
</tr>
<tr>
<td>BAM-1356</td>
</tr>
<tr>
<td>BAM-1314</td>
</tr>
<tr>
<td>BAM-1303</td>
</tr>
<tr>
<td>BAM-1297</td>
</tr>
<tr>
<td>BAM-1295</td>
</tr>
<tr>
<td>BAM-1294</td>
</tr>
<tr>
<td>BAM-1288</td>
</tr>
<tr>
<td>BAM-1257</td>
</tr>
<tr>
<td>BAM-1251</td>
</tr>
<tr>
<td>BAM-1186</td>
</tr>
<tr>
<td>BAM-1178</td>
</tr>
<tr>
<td>BAM-1110</td>
</tr>
<tr>
<td>BAM-1082</td>
</tr>
<tr>
<td>BAM-1069</td>
</tr>
<tr>
<td>BAM-1041</td>
</tr>
<tr>
<td>BAM-1034</td>
</tr>
<tr>
<td>BAM-1012</td>
</tr>
<tr>
<td>BAM-991</td>
</tr>
<tr>
<td>BAM-990</td>
</tr>
<tr>
<td>BAM-981</td>
</tr>
<tr>
<td>BAM-926</td>
</tr>
</tbody>
</table>
Bamboo 3.1 Documentation

| BAM-924 | Perforce client error forces full checkout/build | Resolved |
| BAM-894 | Multiple remote cross platform builds | Resolved |
| BAM-760 | Errors on dashboard should show which build they were raised in | Resolved |
| BAM-705 | Log the user who started the build | Resolved |
| BAM-662 | Having independant build queues underutilises resources | Resolved |
| BAM-498 | Building hangs if one build does not get response from source repository. | Resolved |
| BAM-479 | Starting builds on multiple servers | Resolved |
| BAM-462 | Some errors in XML results parsing not reported to the UI | Resolved |
| BAM-293 | Ability to do remote/distributed builds | Resolved |
| BAM-229 | Link dependent builds to the build which launched them | Resolved |
| BAM-195 | Create new BuildReason object which can pass along more information for dependency builds. | Resolved |
| BAM-21  | Investigate making change detector to run asynchronously or in a queue | Resolved |

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:

---

Bamboo 2.0 Beta 1 Upgrade Guide

Upgrading to Bamboo 2.0 Beta 1

| BAM-924 | Perforce client error forces full checkout/build | Resolved |
| BAM-894 | Multiple remote cross platform builds | Resolved |
| BAM-760 | Errors on dashboard should show which build they were raised in | Resolved |
| BAM-705 | Log the user who started the build | Resolved |
| BAM-662 | Having independant build queues underutilises resources | Resolved |
| BAM-498 | Building hangs if one build does not get response from source repository. | Resolved |
| BAM-479 | Starting builds on multiple servers | Resolved |
| BAM-462 | Some errors in XML results parsing not reported to the UI | Resolved |
| BAM-293 | Ability to do remote/distributed builds | Resolved |
| BAM-229 | Link dependent builds to the build which launched them | Resolved |
| BAM-195 | Create new BuildReason object which can pass along more information for dependency builds. | Resolved |
| BAM-21  | Investigate making change detector to run asynchronously or in a queue | Resolved |
<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 Configuring a Job's 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 Configuring a Job's 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 be 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 Configuring a Job’s Requirements.

**Bamboo 1.2 Release Notes**

Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don’t have Bamboo 3.0? Take a look at the features of Bamboo’s latest major version and try it out!

Atlassian Software Systems is proud to announce the release of Bamboo 1.2. This release contains:

- Permissions (global and plan-based)
- External database support
- Perforce triggering support
- Scheduled backups
- New Bundled NAnt plugin
- Lots of minor features and bug fixes

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.

Want to see Bamboo 1.2 in action? Check out our live opensource instance.

**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 the Bamboo plugins, 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 Fix for IndexOutOfBoundsException in the SVNDeltaReader</td>
<td>✅</td>
<td>✅ Resolved</td>
</tr>
<tr>
<td>BAM-1426 Ability to customize the installation name for a Bamboo instance</td>
<td>✅</td>
<td>✅ Resolved</td>
</tr>
<tr>
<td>BAM-1422 Option to Export/Backup without Artifacts</td>
<td>✅</td>
<td>✅ Resolved</td>
</tr>
<tr>
<td>BAM-1417</td>
<td>Ability to remove artifacts for a build result</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1415</td>
<td>Perforce Build Trigger Scripts</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1406</td>
<td>Perforce change logs not picked up due to update returning too many results.</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1399</td>
<td>Log Output Download Option</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1395</td>
<td>New Jabber command for getting more change logs information</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1384</td>
<td>build expiry should be available on a per project basis</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>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 NullPointerException while deleting build</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1366</td>
<td>NullPointerException after clicking &quot;1. Plan Details&quot; 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>Resolved</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 ImproperlyParsed</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>
</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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><strong>Type</strong></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>BAM-1693</td>
</tr>
<tr>
<td>BAM-1687</td>
</tr>
</tbody>
</table>
**Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? Take a look at the features of Bamboo's latest major version and try it out!**

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>JIRA Issues (24 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

---

**Bamboo 3.1 Documentation**

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
  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**

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.

**Updates and issues fixed**

<table>
<thead>
<tr>
<th>JIRA Issues (24 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---</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>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-1680</td>
</tr>
<tr>
<td>BAM-1668</td>
</tr>
<tr>
<td>BAM-1654</td>
</tr>
<tr>
<td>BAM-1646</td>
</tr>
<tr>
<td>BAM-1625</td>
</tr>
<tr>
<td>BAM-1442</td>
</tr>
<tr>
<td>BAM-1308</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](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](link).

Bamboo 1.2.2 Release Notes

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>
<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-1547</td>
<td>BAM-1547</td>
<td>Users containing the character '@' in their User Names, can't create plans</td>
<td>Edwin Wong [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>☑️ Resolved</td>
<td>Fixed</td>
<td>Aug 02, 2007</td>
<td>Aug 200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>Description</td>
<td>Assignee</td>
<td>Resolved By</td>
<td>Resolution</td>
<td>ID</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>----------</td>
<td>-------------</td>
<td>------------</td>
<td>----</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1445</td>
<td>&quot;Time Taken&quot; duplication on successful export</td>
<td>Brydie McCoy [Atlassian]</td>
<td>James Odeen</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 09, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1436</td>
<td>It's possible to add duplicate labels for a build result you then can't remove them</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 05, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1390</td>
<td>JavaScript error while navigation through the tabs in Edit plan screen</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Suresh Gopalakrishnan</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 25, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1285</td>
<td>trying to edit the build properties to include the bamboo.buildVersion parameter</td>
<td>Brydie McCoy [Atlassian]</td>
<td>John Reynolds</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 24, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1252</td>
<td>Users Can't Export Bamboo with LDAP turned on</td>
<td>Brydie McCoy [Atlassian]</td>
<td>mayank</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 17, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1188</td>
<td>Can't send IM when Bamboo is integrated with Crowd</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 06, 2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1086</td>
<td>z-index of comment hover text is lower than other page elements</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Chris Beams</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Mar 30, 2007</td>
<td></td>
<td></td>
<td></td>
<td></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:

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
<th>Assignee</th>
<th>Resolved By</th>
<th>Resolution</th>
<th>ID</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1445</td>
<td>&quot;Time Taken&quot; duplication on successful export</td>
<td>Brydie McCoy [Atlassian]</td>
<td>James Odeen</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 09, 2007</td>
</tr>
<tr>
<td>BAM-1436</td>
<td>It's possible to add duplicate labels for a build result you then can't remove them</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jul 05, 2007</td>
</tr>
<tr>
<td>BAM-1390</td>
<td>JavaScript error while navigation through the tabs in Edit plan screen</td>
<td>Brydie McCoy [Atlassian]</td>
<td>Suresh Gopalakrishnan</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Jun 25, 2007</td>
</tr>
<tr>
<td>BAM-1285</td>
<td>trying to edit the build properties to include the bamboo.buildVersion parameter</td>
<td>Brydie McCoy [Atlassian]</td>
<td>John Reynolds</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 24, 2007</td>
</tr>
<tr>
<td>BAM-1252</td>
<td>Users Can't Export Bamboo with LDAP turned on</td>
<td>Brydie McCoy [Atlassian]</td>
<td>mayank</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 17, 2007</td>
</tr>
<tr>
<td>BAM-1188</td>
<td>Can't send IM when Bamboo is integrated with Crowd</td>
<td>Mark Chaimungkalanont [Atlassian]</td>
<td>Ajay Sridhar [Atlassian]</td>
<td>Resolved</td>
<td>Fixed</td>
<td>May 06, 2007</td>
</tr>
<tr>
<td>BAM-1086</td>
<td>z-index of comment hover text is lower than other page elements</td>
<td>Adrian Hempel [Atlassian]</td>
<td>Chris Beams</td>
<td>Resolved</td>
<td>Fixed</td>
<td>Mar 30, 2007</td>
</tr>
</tbody>
</table>
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>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1477</td>
<td>Webwork 2.2.2 security exploit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1475</td>
<td>Plan dependencies not exported or imported (not sure which)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1471</td>
<td>Bamboo 1.2 JAVA (Unsupported major.minor version 49.0) error.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1470</td>
<td>The Exclude email header &quot;Precedence: bulk&quot; option is not exported and imported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1460</td>
<td>Bamboo POM's need to be updated for the next point release</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1455</td>
<td>Slow performance on test results page.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1453</td>
<td>Importing from a different server from builds pre: 406 causes inconsistent build directories</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1452</td>
<td>Upgrades from 1.0.x directly to 1.2 fails</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1437</td>
<td>Line breaks are lost in user comments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1359</td>
<td>Integrate enhanced regex labeller plugin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-978</td>
<td>Bamboo Plugin Dev Kit pom.xml has a snapshot dependency.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-977</td>
<td>Plugin Dev Kit has Mac OSX specific files in it.</td>
<td></td>
<td></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

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: test, admin</td>
<td>Edit</td>
</tr>
<tr>
<td>Notify After X Failed Builds</td>
<td>Roles: Watcher</td>
<td>Edit</td>
</tr>
<tr>
<td>All Completed Builds</td>
<td>Roles: Committer</td>
<td>Edit</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**

<table>
<thead>
<tr>
<th>Notification Trigger:</th>
<th>Roles:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Completed Builds</td>
<td>Watcher - Users who have marked this build as favourite</td>
</tr>
</tbody>
</table>

**Notification Preferences**

Different users prefer to get notified in different ways. Bamboo now lets you control that, via the new user notification preferences.

**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.

**Metadata**

This build has the following metadata. These are property key value pairs describing the build. You can specify your own metadata in the build process via plugins.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildKey</td>
<td>EXT-BLAH</td>
</tr>
<tr>
<td>buildNumber</td>
<td>31</td>
</tr>
<tr>
<td>bambooVersion</td>
<td>1.1</td>
</tr>
</tbody>
</table>

**File Trigger Inclusions/Exclusions**

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>JIRA Issues (50 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>BAM-1191</td>
</tr>
<tr>
<td>BAM-1189</td>
</tr>
<tr>
<td>BAM-1183</td>
</tr>
<tr>
<td>BAM-1173</td>
</tr>
<tr>
<td>BAM-1163</td>
</tr>
<tr>
<td>BAM-1155</td>
</tr>
<tr>
<td>BAM-1134</td>
</tr>
<tr>
<td>BAM-1124</td>
</tr>
<tr>
<td>BAM-1115</td>
</tr>
<tr>
<td>BAM-1090</td>
</tr>
<tr>
<td>BAM-1088</td>
</tr>
<tr>
<td>BAM-1081</td>
</tr>
<tr>
<td>BAM-1080</td>
</tr>
<tr>
<td>BAM-1073</td>
</tr>
<tr>
<td>BAM-1059</td>
</tr>
<tr>
<td>BAM-1058</td>
</tr>
<tr>
<td>BAM-1053</td>
</tr>
<tr>
<td>BAM-1032</td>
</tr>
<tr>
<td>BAM-1029</td>
</tr>
<tr>
<td>BAM-1020</td>
</tr>
<tr>
<td>BAM-1016</td>
</tr>
<tr>
<td>BAM-1011</td>
</tr>
<tr>
<td>BAM-1010</td>
</tr>
<tr>
<td>BAM-996</td>
</tr>
<tr>
<td>BAM-952</td>
</tr>
<tr>
<td>BAM-950</td>
</tr>
<tr>
<td>BAM-942</td>
</tr>
<tr>
<td>BAM-940</td>
</tr>
<tr>
<td>BAM-938</td>
</tr>
<tr>
<td>BAM-934</td>
</tr>
<tr>
<td>BAM-915</td>
</tr>
<tr>
<td>BAM-856</td>
</tr>
<tr>
<td>BAM-847</td>
</tr>
<tr>
<td>BAM-819</td>
</tr>
<tr>
<td>BAM-792</td>
</tr>
<tr>
<td>BAM-787</td>
</tr>
<tr>
<td>BAM-767</td>
</tr>
<tr>
<td>BAM-729</td>
</tr>
<tr>
<td>BAM-681</td>
</tr>
<tr>
<td>BAM-680</td>
</tr>
</tbody>
</table>
Bamboo 3.1.2 Release Notes

Atlassian is proud to announce the release of Bamboo 3.1.2! Bamboo 3.1.2 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</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>21 issues</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1296</td>
<td>🟢</td>
<td>Resolved</td>
</tr>
<tr>
<td>PATH variable gets clobbered in JDK 1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1289</td>
<td>🟢</td>
<td>Resolved</td>
</tr>
<tr>
<td>Export not working in Bamboo 1.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1279</td>
<td>🟢</td>
<td>Resolved</td>
</tr>
<tr>
<td>User and group browser very slow in LDAP environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1265</td>
<td>🟢</td>
<td>Resolved</td>
</tr>
<tr>
<td>When adding builders - adding a label with space in the beginning fails to build</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1254</td>
<td>Upgrade bundled svnkit version to get bugfixes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1238</td>
<td>Improve the CVS trigger doco</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1235</td>
<td>Backslashes in usernames not displayed correctly on tooltip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1232</td>
<td>Update docs after packaging the scripts in the WAR version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1231</td>
<td>Include the scripts for triggering CVS and SVN builds in the WAR version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1229</td>
<td>Dependency builds incorrectly reported as initial or manual build in email notifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1138</td>
<td>File Version Number problems with Perforce</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1094</td>
<td>start-bamboo wrapper uses false uname syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1065</td>
<td>Bamboo reports twice as many unit test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1046</td>
<td>Valid cron expression not really valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-973</td>
<td>Users with non standard characters in their names show up corrupted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-971</td>
<td>JIRA issue parser should look for numbers in project key as well.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-745</td>
<td>Encoding (of project name)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-518</td>
<td>Unable to copy build artifacts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAM-1233</td>
<td>A failed build is shown as a successful build</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1227</td>
<td>Bamboo throws freemarker exception when LDAP accounts don't have full names for users</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1226</td>
<td>Building a project with Bamboo with JDK 1.4 sometimes fails when it does build fine outside of Bamboo</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1224</td>
<td>Improve the REST api in Bamboo to allow getting details from Bamboo at the project level instead of the plan/build level.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1216</td>
<td>logging in as an ldap user causes a db exception</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1215</td>
<td>Import fails if the import file is too large</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1206</td>
<td>Unable to remove build plan dependency</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1200</td>
<td>Unable to change how Bamboo sends notifications to user</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1195</td>
<td>Wording changes for notifications</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1192</td>
<td>&quot;build.dependency.select.none&quot; shown when there are no dependencies</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1165</td>
<td>Text on 'Completed Builds' page doesn't match plan build strategy.</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1154</td>
<td>Changes Made While Build is In Queue Don't Get Change Logs</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1151</td>
<td>Create plan input lost on timeout</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1120</td>
<td>Arguments field in bash script/script builders are mandatory</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1117</td>
<td>Ant builder checks last 5 lines for BUILD SUCCESSFUL</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-1066</td>
<td>Bamboo sometimes doesn't know who did CVS commit</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-825</td>
<td>OutOfMemoryException thrown when importing large zip-files</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>BAM-505</td>
<td>Bamboo may pickup wrong java installation in sub-processes</td>
<td>Resolved</td>
<td></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**

Bamboo 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? Take a look at the features of Bamboo's latest major version and try it out!

Atlassian is proud to announce the final release of Bamboo 1.0! Bamboo 1.0 is the first official release of Atlassian's new Continuous Integration and Build Server.

Bamboo is more than just a build server — it is an entire Build Telemetry system designed to provide you with unprecedented insight into your development processes.
To 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>Plan</th>
<th>Status</th>
<th>Priority</th>
<th>Key</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Bamboo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All Plans</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Other updates and bug fixes.

**JIRA Issues** (36 issues)

<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>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-878</td>
<td>NumberFormatException for Test</td>
<td>![High]</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>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-875</td>
<td>User page no longer show tabs with author information on them</td>
<td>![High]</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>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-867</td>
<td>test mail should contain clickable base url</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-860</td>
<td>New more condensed dashboard</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-857</td>
<td>Document our external Javascript widgets</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-839</td>
<td>Tests Page Has URL Escapes</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-837</td>
<td>Allow Properties to be Passed to Ant</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-835</td>
<td>Build completed time on summary page is actually build start time</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-821</td>
<td>Unable to export build configuration - no info on how to repair</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-807</td>
<td>Bamboo passes bad parameter diff ViewVC</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-803</td>
<td>Use minified version of js libs in 1.0 final</td>
<td>![High]</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-796</td>
<td>Number Format Exception</td>
<td>![High]</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide.  

Don't have Bamboo 3.0? 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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BAM-1139</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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don't have Bamboo 3.0? 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.

**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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>BAM-1113</td>
</tr>
<tr>
<td>BAM-1096</td>
</tr>
<tr>
<td>BAM-1085</td>
</tr>
<tr>
<td>BAM-1078</td>
</tr>
<tr>
<td>BAM-1056</td>
</tr>
<tr>
<td>BAM-1028</td>
</tr>
<tr>
<td>BAM-979</td>
</tr>
</tbody>
</table>
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 (eg. 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

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>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-1063</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1017</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1008</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1005</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-1000</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-993</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-986</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-976</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-964</td>
<td></td>
<td>Resolved</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>---</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>
<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>
</tbody>
</table>
Bamboo 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

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</th>
<th>(28 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Summary</td>
</tr>
<tr>
<td>BAM-975</td>
<td>Edit Configuration broken</td>
</tr>
<tr>
<td>BAM-965</td>
<td>Plan Summary does not render in IE 7 when logged in as a user</td>
</tr>
<tr>
<td>BAM-959</td>
<td>Broken builds have incorrect links when restarting builds</td>
</tr>
<tr>
<td>BAM-958</td>
<td>Links to source is broken</td>
</tr>
<tr>
<td>BAM-943</td>
<td>Sessions need to closed in the finally block</td>
</tr>
<tr>
<td>BAM-939</td>
<td>Need to convert build level plugins to use web fragments</td>
</tr>
<tr>
<td>BAM-937</td>
<td>Importing data doesn't guarantee unique ids</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

![Warning](warning.png)

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

---

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAM-936</td>
<td>Smack Client does not recognize project/plan keys with numbers</td>
</tr>
<tr>
<td>BAM-935</td>
<td>SVN Repository doesn't use default authentication</td>
</tr>
<tr>
<td>BAM-933</td>
<td>Export and Import doesn't work when moving to a new Bamboo Home path</td>
</tr>
<tr>
<td>BAM-925</td>
<td>Viewing User via authors and via profile need to be separate requests as different info is needed in both</td>
</tr>
<tr>
<td>BAM-918</td>
<td>Builders edit screen not populating existing values.</td>
</tr>
<tr>
<td>BAM-917</td>
<td>If an initial build has no queues to go into, it may cause repeated clean builds</td>
</tr>
<tr>
<td>BAM-916</td>
<td>Subversion Event Handler is not all that Null safe</td>
</tr>
<tr>
<td>BAM-908</td>
<td>Standalone Bamboo cannot start in certain Linux environments</td>
</tr>
<tr>
<td>BAM-899</td>
<td>Unable to Edit Build Configuration</td>
</tr>
<tr>
<td>BAM-897</td>
<td>Null pointer exception creating a build plan that uses svn+ssh</td>
</tr>
<tr>
<td>BAM-889</td>
<td>Precedence: bulk mail header causing notifications to be blocked</td>
</tr>
<tr>
<td>BAM-869</td>
<td>Error creating new build plan</td>
</tr>
<tr>
<td>BAM-851</td>
<td>Bamboo cannot run on 64-bit linux machines</td>
</tr>
<tr>
<td>BAM-848</td>
<td>Responding via IM to build notifications is unreliable, the comment is not ascribed to me</td>
</tr>
<tr>
<td>BAM-841</td>
<td>IE7 Fails Often, Cancels Page</td>
</tr>
<tr>
<td>BAM-790</td>
<td>IE 7 sometimes can't display build page</td>
</tr>
<tr>
<td>BAM-746</td>
<td>Headless Unix Server</td>
</tr>
<tr>
<td>BAM-692</td>
<td>Manual build strategy gobbles up CVS errors</td>
</tr>
<tr>
<td>BAM-634</td>
<td>Sample plugin: Out of Memory tagging</td>
</tr>
<tr>
<td>BAM-297</td>
<td>“Disabled” status should be noted prominently on build summary</td>
</tr>
<tr>
<td>BAM-90</td>
<td>Lower priority of spawned build processes</td>
</tr>
</tbody>
</table>
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>BAM-697</td>
<td>CVS connection fails if password has @ in it</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></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><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-676</td>
<td>Unable to re-index due to locked file</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-658</td>
<td>Project names not ordered in dropdown to create project</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-656</td>
<td>No way to ‘complete’ setting up a project as ‘save’ hidden by javascript</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-652</td>
<td>Checkboxes don't work properly when removing dependant builds</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-647</td>
<td>Split webapp WAR module into a JAR and a WAR module</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-632</td>
<td>Breadcrumbs should have build numbers</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-631</td>
<td>Reports on top ten longest time to fix for tests</td>
<td><img src="https://img.shields.io/badge/Priority-0-orange-green" alt="" /></td>
</tr>
<tr>
<td>BAM-630</td>
<td>Test summary page still using the old style</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-625</td>
<td>Redirect after a plan is created</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-624</td>
<td>IE caches ajax response for comments and labels</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-622</td>
<td>Last screen of create build broken on Safari</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-616</td>
<td>Capture code changes for dependent and scheduled builds</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-612</td>
<td>Label grammar</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-611</td>
<td>Allow two character plan keys</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-609</td>
<td>Error displaying build queue admin page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-608</td>
<td>Too many files open error</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-606</td>
<td>Invalid path to clover throws error</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-603</td>
<td>Accessing /api/index.action throws a freemarker error</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-602</td>
<td>Minor issues with the build status wizard section on the editBuildConfiguration.action page</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-595</td>
<td>Adding a comment from the Summary page doesn't work in IE</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-579</td>
<td>Make top right options clearer</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-572</td>
<td>Need option to disable signups</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-570</td>
<td>Testing for mail and IM servers should be more visible</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-569</td>
<td>Auto report grouping for Tests doesn't seem to work</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-566</td>
<td>The back button on create plan wizard clears previously selected values</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-563</td>
<td>Validation for report not being selected</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-543</td>
<td>Minor tweaks of the Admin pages</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-542</td>
<td>Plugin Points for Web Fragments</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-490</td>
<td>Ability to run a Bamboo in 'private mode'</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-162</td>
<td>Passwords in plaintext</td>
<td>Resolved</td>
</tr>
<tr>
<td>BAM-91</td>
<td>Static files are not cached, increases size of downloads</td>
<td>Resolved</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 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**

Page: Bamboo 3.1 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 3.0 has been released. Read the Bamboo 3.0 Release Notes and Upgrade Guide. Don’t have Bamboo 3.0? 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:

<table>
<thead>
<tr>
<th>Bamboo Paths</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current running directory</td>
<td>/opt/j2ee/domains/bamboo.atlassian.com/jira/apache-tomcat-5.5.20</td>
</tr>
<tr>
<td>Configuration Path</td>
<td>/home/j2ee/bamboo/xml-data/configuration</td>
</tr>
<tr>
<td>Build Path</td>
<td>/home/j2ee/bamboo/xml-data/builds</td>
</tr>
<tr>
<td>Build Working Directory</td>
<td>/home/j2ee/bamboo/xml-data/build-dir</td>
</tr>
<tr>
<td>Bamboo Home</td>
<td>/home/j2ee/bamboo</td>
</tr>
</tbody>
</table>

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 your Bamboo Home, Build Data Path and Configuration Path directories, identified in Step 1. For more information about these directories please see Important Directories and Files.

If you are using an external database, then use native database backup tools to acquire a database dump.

Note that it is not necessary to back up your Working Directory.
Step 4. Re-install Bamboo

Please note:
If you are installing using the .zip or .tgz packages:

- 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).

If you are installing using the Windows installer or the Mac installer:

- Ensure that you will create a new empty database that can be used during the installation Wizard and that the Bamboo Home directory from the old installation will not be used. The new installation will need to point to a new path so that a new Bamboo Home can be created. Once the installation was finished, you can then point the new installation to the old Bamboo Home, changing the path at file `<Bamboo-install>/webapps/WEB-INF/classes/bamboo-init.properties`

- 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.

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

Before you start Bamboo
Bamboo needs to have write access to your database to complete the upgrade tasks that will run when you start up Bamboo. Please consult your database documentation to ensure that you have configured your database appropriately.

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 allow 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.

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
- Our Patch Policy
- Security Advisories

Finding and Reporting a Security Vulnerability

Atlassian’s approach to releasing patches is detailed in How to Report a Security Issue.

Publication of Bamboo Security Advisories

Atlassian’s approach to publishing security advisories is detailed in Security Advisory Publishing Policy.

Severity Levels

Atlassian’s scale for measuring security issues is detailed in Severity Levels for Security Issues.

Our Patch Policy

Atlassian’s approach to releasing patches is detailed in our Security Patch Policy.

Security Advisories

- Bamboo Security Advisory 2011-03-29
- Bamboo Security Advisory 2010-05-04
- Bamboo Security Advisory 2009-03-09
- Bamboo Security Advisory 2008-02-08 (Bamboo 2.0 Beta)

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.

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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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
     ```
   - **(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
     ```
   - **(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
     -Djavax.net.ssl.trustStore=/path/to/client.ts -jar bamboo-agent-2.0.jar
     http://192.168.3.235:8085/agentServer/
     ```

   **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 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 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://www.cgisecurity.com), [CERT](http://www.cert.org) 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://www.cgisecurity.com), [CERT](http://www.cert.org) 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.
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 [cigisecurity](http://cigisecurity.com), [CERT](http://www.cert.org) 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 Security Advisory 2010-05-04**

In this advisory:

- XSS Vulnerabilities
  - Severity
  - Risk Assessment
  - Vulnerability
  - Risk Mitigation
  - Fix
- General Tightening of the Bamboo Security Model
  - Severity
  - Risk Assessment
  - Vulnerability
  - Risk Mitigation
  - Fix
  - Changed Behaviour in Bamboo

**XSS Vulnerabilities**
Severity

Atlassian rates these vulnerabilities as **high**, according to the scale published in *Severity Levels for Security Issues*. The scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed several cross-site scripting (XSS) vulnerabilities in Bamboo, which may affect Bamboo instances. These vulnerabilities have security implications and are especially important for anyone running publicly accessible instances of Bamboo.

- The attacker might take advantage of the vulnerability to steal other users’ session cookies or other credentials, by sending the credentials back to the attacker’s own web server.
- The attacker's text and script might be displayed to other people viewing a Bamboo page. This is potentially damaging to your company’s reputation.

You can read more about XSS attacks at [cgitsecurity](http://cgitsecurity), [CERT](http://cert) and other places on the web.

Vulnerability

All version of Bamboo up to and including Bamboo 2.5.3 are susceptible to these vulnerabilities.

An attacker can inject their own malicious JavaScript code into areas of Bamboo listed in the table below. This code could be executed by simply entering the URL into the browser address bar or when a user performs a specific function in Bamboo, such as clicking a link or a button.

<table>
<thead>
<tr>
<th>Affected areas in Bamboo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server Administration User Interface</strong> — Including the User and Group Security, System and Communication sections.</td>
</tr>
<tr>
<td><strong>Main Bamboo User Interface</strong> — Including the Create Plan and Build Configuration areas and Log and various Result views.</td>
</tr>
</tbody>
</table>

Risk Mitigation

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Please see the ‘fix’ section below.

Fix

Bamboo 2.5.5 fixes these vulnerabilities. See the release notes and upgrade guide for more information about this release and changes to Bamboo’s behaviour. You can download the latest version of Bamboo from the download centre.

There are no patches available to fix these vulnerabilities for previous versions of Bamboo.

General Tightening of the Bamboo Security Model

Severity

Atlassian rates one of these vulnerabilities as **high** and the other as **moderate**, according to the scale published in *Severity Levels for Security Issues*. The scale allows us to rank a vulnerability as critical, high, moderate or low.

Risk Assessment

We have identified and fixed two potential security vulnerabilities in Bamboo. These vulnerabilities have security implications that are especially important for anyone running publicly accessible instances of Bamboo.

An attacker, who has gained administrator access to a Bamboo instance, could set Bamboo's export, import and scheduled backup paths to a location within the Bamboo web application directory. Once this has been done, the attacker will be able to download any Bamboo data which has been exported or backed up by Bamboo. If you have followed standard guidelines for hardening your application servers, then your Bamboo instance should be less susceptible to this vulnerability. Therefore, we have provided an optional mechanism that prevents directory paths from being changed.

Bamboo does not set a maximum number of repeated login attempts. This makes Bamboo vulnerable to brute force attacks. Therefore, we have prevented brute force attacks by imposing a maximum number of repeated login attempts.

For Bamboo Standalone distributions, we have set Bamboo's session ID cookies to use the *HttpOnly* flag. This makes it more difficult for malicious (JavaScript) code on a client’s browser to gain access to these session ID cookies, thereby minimising the risk of common XSS attacks.

Vulnerability

All version of Bamboo up to and including Bamboo 2.5.3 are susceptible to these vulnerabilities.

Please refer to the following JIRA issues for more information:

- BAM-5775 for restricting the ability to set Bamboo's file paths.
- BAM-5708 for brute force attack prevention in Bamboo.
• **BAM-5668** for HttpOnly session ID cookies in Bamboo Standalone.

**Risk Mitigation**

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Please see the 'fix' section below.

If you are running the Bamboo EAR-WAR distribution, then to minimise the risk of common XSS attacks, we strongly recommend that you configure the application server (Tomcat) running Bamboo to transmit session ID cookies using the HttpOnly flag. Please refer to Configuring Tomcat to Use HttpOnly Session ID Cookies for more information.

**Fix**

Bamboo 2.5.5 fixes these vulnerabilities. See the release notes and upgrade guide for more information about this release and changes to Bamboo's behaviour. You can download the latest version of Bamboo from the download centre.

There are no patches available to fix these vulnerabilities for previous versions of Bamboo.

**Changed Behaviour in Bamboo**

As a consequence of these security fixes, the following changes to Bamboo's default behaviour have occurred.

- When modifying Bamboo's 'File Path' option on the Export or Import administration pages or the 'Backup Path' option on the Scheduled Backup page, you can only change the name of files associated with these options (not the actual file path component itself). To change these file path components, you must explicitly run Bamboo with the following system property:

  ```
  bamboo.paths.set.allowed=true
  ```

  Please refer to Configuring System Properties for details on how to run Bamboo with system properties.

- If you attempt to log in to Bamboo three times unsuccessfully, Bamboo will then require subsequent login attempts to be accompanied by text from a Captcha image.

For details about changes to Bamboo's behaviour as a result of these fixes to security vulnerabilities, please refer to the Bamboo 2.5.5 Upgrade Guide.

**Bamboo Security Advisory 2011-03-29**

This advisory announces a security vulnerability that we have found in all versions of Bamboo prior to 2.7.4 and fixed in 2.7.4 and later. You need to upgrade your existing Bamboo installations to fix this vulnerability. JIRA Studio is not vulnerable to any of the issues described in this advisory.

Atlassian is committed to improving product security. The vulnerabilities listed in this advisory have been discovered by Atlassian, unless noted otherwise. The reporter may also have requested that we do not credit them.

If you have questions or concerns regarding this advisory, please raise a support request at http://support.atlassian.com/.

In this advisory:

- XSS Vulnerability in Bamboo User Management
  - Severity
  - Risk Assessment
  - Vulnerability
  - Risk Mitigation
  - Fix

**XSS Vulnerability in Bamboo User Management**

**Severity**

Atlassian rates the severity level of these vulnerabilities as high, according to the scale published in Severity Levels for Security Issues. The scale allows us to rank the severity as critical, high, moderate or low.

**Risk Assessment**

We have identified and fixed a cross-site scripting (XSS) vulnerability in Bamboo. This XSS vulnerability allows an attacker to embed their own JavaScript into a Bamboo page. You can read more about XSS attacks and consequences at cgisecurity.com, The Web Application Security Consortium and other places on the web.

**Vulnerability**

The table below describes the Bamboo versions and the specific functionality affected by the XSS vulnerability.

<table>
<thead>
<tr>
<th>Bamboo Feature</th>
<th>Affected Bamboo Versions</th>
<th>Issue Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

582
**Risk Mitigation**

We recommend that you upgrade your Bamboo installation to fix these vulnerabilities. Alternatively, if you are not in a position to upgrade immediately and you judge it necessary, you can restrict access to trusted groups.

**Fix**

Bamboo 2.7.4 and later versions fix this issue. View the issue linked above for information on fix versions. For a full description of this release, see the Bamboo 2.7.4 Release Notes. You can download the latest version of Bamboo from the Bamboo download centre.

There are no patches available to fix these vulnerabilities. You must upgrade your Bamboo installation.

**Supported Platforms**

This page contains announcements of the end of support for various platforms and browsers when used with Bamboo. This is summarised in the table below. Please see the sections following for the full announcements.

**End of Support Matrix for Bamboo**

The table below summarises information regarding the end of support announcements for upcoming Bamboo releases. If a platform (version) has already reached its end of support date, it is not listed in the table.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Bamboo End of Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>No platforms have been scheduled for end of support at this time.</td>
</tr>
</tbody>
</table>

**Why is Atlassian ending support for these platforms?**

Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as the complexity of our applications grows, the cost of supporting multiple platforms increases exponentially. Each new feature has to be tested on several combinations of application servers, databases, web browsers, etc, with setup and ongoing maintenance of automated tests. Moving forward, we want to reduce the time spent there to increase Bamboo development speed significantly.

**On this page (most recent announcements first):**

- Deprecated Java Platforms for Bamboo (16 February 2011)
- Deprecated Web Browsers for Bamboo (16 February 2011)

**Deprecated Java Platforms for Bamboo (16 February 2011)**

This section announces the end of Atlassian support for certain Java Platforms for Bamboo.

We will stop supporting the following Java Platforms:

- From Bamboo 3.1, due in the first half of 2011, support for Java Platform 5 (JDK/JRE 1.5) will end.

We are ending support for Java Platform 5, in line with Sun's Java SE Support Road Map (i.e. "End of Service Life" for Java Platform 5 dated October 30, 2009). We are committed to helping our customers understand this decision and assist them in updating to Java Platform 6, our supported Java Platform.

The details are below. Please refer to the Supported Platforms for more details regarding platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

**End of Life Announcement for Java Platform Support**

<table>
<thead>
<tr>
<th>Java Platform</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Platform 5 (JDK/JRE 1.5)</td>
<td>When Bamboo 3.1 releases, due in the first half of 2011</td>
</tr>
</tbody>
</table>

**Java Platform 5 End of Support Notes:**

- 'Support End Date' means that Bamboo 3.0.x and previous released versions will continue to work with Java Platform 5 (JDK/JRE 1.5), however we will not fix bugs related to Java Platform 5 past the support end date.
- Bamboo 3.1 will only be tested with and support Java Platform 6 (JDK/JRE 1.6).
- If you have concerns with this end of support announcement, please email eol-announcement at atlassian dot com.
Deprecated Web Browsers for Bamboo (16 February 2011)

This section announces the end of Atlassian support for certain web browser versions for Bamboo. End of support means that Atlassian will not fix bugs related to certain web browser versions past the support end date.

We will stop supporting the following web browser versions from Bamboo 3.0, due February 2011:

- Microsoft Internet Explorer 7 (IE7)

The details are below. Please refer to the list of supported platforms for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

End of Life Announcement for Web Browser Support

<table>
<thead>
<tr>
<th>Web Browser</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer (version 7 only)</td>
<td>When Bamboo 3.0 releases, due February 2011</td>
</tr>
</tbody>
</table>

- Internet Explorer Notes:
  - Atlassian intends to end support for IE7 in Bamboo 3.0. Bamboo 2.7 is the last version that will support IE7.
  - IE8 will still be supported.
  - 'Support End Date' means that Bamboo 2.7 and previously released versions will continue to work with IE7. However, we will not fix bugs affecting IE7 past the support end date.
  - Bamboo 3.0 will not be tested with IE7.

End of Support Announcements for Bamboo

This page contains announcements of the end of support for various platforms and browsers when used with Bamboo. This is summarised in the table below. Please see the sections following for the full announcements.

End of Support Matrix for Bamboo

The table below summarises information regarding the end of support announcements for upcoming Bamboo releases. If a platform (version) has already reached its end of support date, it is not listed in the table.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Bamboo End of Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Platform 5 (JDK/JRE 1.5)</td>
<td>Bamboo 3.1 (announcement)</td>
</tr>
</tbody>
</table>

Why is Atlassian ending support for these platforms?
Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as the complexity of our applications grows, the cost of supporting multiple platforms increases exponentially. Each new feature has to be tested on several combinations of application servers, databases, web browsers, etc, with setup and ongoing maintenance of automated tests. Moving forward, we want to reduce the time spent there to increase Bamboo development speed significantly.

On this page (most recent announcements first):

- Deprecated Java Platforms for Bamboo (16 February 2011)
- Deprecated Web Browsers for Bamboo (16 February 2011)

Deprecated Java Platforms for Bamboo (16 February 2011)

This section announces the end of Atlassian support for certain Java Platforms for Bamboo.

We will stop supporting the following Java Platforms:

- From Bamboo 3.1, due in the first half of 2011, support for Java Platform 5 (JDK/JRE 1.5) will end.

We are ending support for Java Platform 5, in line with Sun's Java SE Support Road Map (i.e. "End of Service Life" for Java Platform 5 dated October 30, 2009). We are committed to helping our customers understand this decision and assist them in updating to Java Platform 6, our supported Java Platform.

The details are below. Please refer to the Supported Platforms for more details regarding platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

End of Life Announcement for Java Platform Support
Java Platform 5 End of Support Notes:
- 'Support End Date' means that Bamboo 3.0.x and previous released versions will continue to work with Java Platform 5 (JDK/JRE 1.5), however we will not fix bugs related to Java Platform 5 past the support end date.
- Bamboo 3.1 will only be tested with and support Java Platform 6 (JDK/JRE 1.6).
- If you have concerns with this end of support announcement, please email eol-announcement at atlassian dot com.

Deprecated Web Browsers for Bamboo (16 February 2011)

This section announces the end of Atlassian support for certain web browser versions for Bamboo. End of support means that Atlassian will not fix bugs related to certain web browser versions past the support end date.

We will stop supporting the following web browser versions from Bamboo 3.0, due February 2011:
- Microsoft Internet Explorer 7 (IE7)

The details are below. Please refer to the list of supported platforms for details of platform support for Bamboo. If you have questions or concerns regarding this announcement, please email eol-announcement at atlassian dot com.

End of Life Announcement for Web Browser Support

<table>
<thead>
<tr>
<th>Web Browser</th>
<th>Support End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer (version 7 only)</td>
<td>When Bamboo 3.0 releases, due February 2011</td>
</tr>
</tbody>
</table>

Internet Explorer Notes:
- Atlassian intends to end support for IE7 in Bamboo 3.0. Bamboo 2.7 is the last version that will support IE7.
- IE8 will still be supported.
- 'Support End Date' means that Bamboo 2.7 and previously released versions will continue to work with IE7. However, we will not fix bugs affecting IE7 past the support end date.
- Bamboo 3.0 will not be tested with IE7.

Bamboo FAQ
Bamboo FAQ

Answers to commonly raised questions about configuring and using Bamboo:

- **What Is Continuous Integration?**

- **Installation FAQ**
  - Changing the Root Context Path
  - Configuring Bamboo on start-up
  - Getting Bamboo Standalone to use the jetty.xml file
  - Installation notes for Bamboo on JBoss 4.x
  - Running Bamboo as a Service on Windows
  - Running Bamboo over HTTPS
  - Running Bamboo service on Windows as the local user
  - Setting up JNDI mail on JBoss 4.2.2

- **Support Policies**
  - Bamboo Support Policy
  - Bug Fixing Policy
  - Deploying Multiple Atlassian Applications in a Single Tomcat Container
  - How to Report a Security Issue
  - New Features Policy
  - Patch Policy
  - Security Advisory Publishing Policy
  - Security Patch Policy
  - Severity Levels for Security Issues

- **Usage FAQ**
  - 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 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
  - Do I have to upgrade all remote agents for Bamboo Release 2.1.2?
  - Enable User Management debug logging 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
  - 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
  - Troubleshooting SOL Exceptions - Detailed Hibernate Logging
  - Using Bamboo For Release Management
  - Using Bamboo with Clover
  - Working with Sun JAVA libraries
  - Bamboo indicates that my Ant or Maven builds failed, even though they were successful
  - How can I pass bamboo variables to my build script
  - Finding the Support Entitlement Number (SEN)
  - How do I manually set the version of new Subversion workspaces?
  - Securing your repository connection
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

On this page:

- Bamboo FAQ
- Bamboo Evaluator’s FAQ
- Need more help?

Installation FAQ

- Changing the Root Context Path
- Configuring Bamboo on start-up
- Getting Bamboo Standalone to use the jetty.xml file
- Installation notes for Bamboo on JBoss 4.x
- Running Bamboo as a Service on Windows
- Running Bamboo over HTTPS
- Running Bamboo service on Windows as the local user
- Setting up JNDI mail on JBoss 4.2.2
- Setting up JNDI on Jetty

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:

```
wrapper.app.parameter.4=/
```
and replace it with the following line:

```
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 runtime parameters for bamboo.war
- Configuring Bamboo agent on start-up
- Configuring Bamboo agent service
- Configuring classic Bamboo agent

### Configuring Bamboo Server on start-up

#### 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:

```
# This is how the Bamboo server will be started
#
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 /*"
```

- `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.

For example to to add more java parameters to the bamboo process extend the parameter list...
Configuring Bamboo runtime parameters for bamboo.war

The application container that deploys bamboo has to be configured with the additional java parameter.

Example Tomcat:

```
/bin/setenv.sh
...
JAVA_OPTS="-server -XX:MaxPermSize=256m -Dbamboo.home=/path/to/bamboo-tomcat-home -Xmx512m
-Djava.awt.headless=true -D<your-parameter>=<value> $JAVA_OPTS"
export JAVA_OPTS
...```

Configuring Bamboo agent on start-up

Configuring Bamboo agent service

Bamboo agent uses the same wrapper as the server. If you are running Bamboo as a service, then edit the conf/wrapper.conf file in `<Bamboo-Agent-Home>`

For example to to add more java parameters to the bamboo process extend the parameter list

```
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
wrapper.java.additional.4=-D<your-parameter>
```

Configuring classic Bamboo agent

To apply additional properties to the classic Bamboo agent append the system to the start-up command.

In the example below, we are specifying a Bamboo Home for the agent by adding `-Dbamboo.home` system property during startup:

```
// Without a Bamboo Home specified
// With a Bamboo Home specified
java -jar -Dbamboo.home=/bamboo/home bamboo-agent-2.6.2.jar
http://192.168.35.128:8085/bamboo/agentServer/
```

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):

For Bamboo 3.0 and newer:

```
RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH
8085 ./webapp/"
```

For Bamboo older than 3.0:
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:

For Bamboo 3.0 and newer:

RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.eclipse.jetty.xml.XmlParser.Validating=false com.atlassian.bamboo.server.Server webapp/WEB-INF/classes/jetty.xml"

For Bamboo older than 3.0:

RUN_CMD="java -Xms256m -Xmx512m -Djava.awt.headless=true -classpath $CLASSPATH -Dorg.mortbay.xml.XmlParser.NotValidating=true 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=/"

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:**

**Linux Platforms**

Replace your existing `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml` file, with this jetty.xml file.

**Windows Platforms**

Replace your existing `<Bamboo-install>/webapp/WEB-INF/classes/jetty.xml` file, with this jetty.xml file.

**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.
Bamboo 2.4.x does not run on JBoss 4.2.3 or later

We are aware of a JBoss issue that currently prevents Bamboo 2.4.x from running on JBoss 4.2.3 or later. If you are using JBoss 4.2.3 or later, we recommend that you do not upgrade your Bamboo installation until a fix has been implemented. Please see BAM-4705 for more information.

File extraction notes

To deploy Bamboo EAR-WAR onto your JBoss application server, copy the Bamboo WAR file to 

```bash
../<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. 

```bash
<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:

```bash
if exist "%JBoss_HOME%\bin\native" set JAVA_OPTS=%JAVA_OPTS%
-Djava.library.path=%JBoss_HOME%\bin\native
```

- **Linux-based systems:**
  1. Find the run.sh file
  2. Edit JAVA_OPTS to set the desired properties variable:

```bash
# Setup JBoss sepecific properties
JAVA_OPTS="-Dprogram.name=$PROGNAME $JAVA_OPTS"
```

For further reference

Please visit the [JBoss Wiki page on setting JavaOpts](http://host:port/atlassian-bamboo-1.1.2/).

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 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:

```
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 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.msc’.
   2. The ‘Services’ window will display (see screenshot below). Double-click the ‘Bamboo build server’ row.

   ![Bamboo Service Window](image)

   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 mail on JBoss 4.2.2**

This page explains how to set up the (gmail) mail service for JBoss 4.2.2. with the following features:

- smtp over SSL
**TLS encryption**

After installing bamboo.war on jboss modify `<jboss-install>/server/default/deploy/mail-service.xml` to be

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: mail-service.xml 62349 2007-04-15 16:48:15Z dimitris@jboss.org $ -->
<server>

<!-- Mail Connection Factory -->
<mbean code="org.jboss.mail.MailService" name="jboss:service=Mail">
  <attribute name="JNDIName">java:/Mail</attribute>
  <attribute name="User">${account}@gmail.com</attribute>
  <attribute name="Password">${pw}</attribute>
  <attribute name="Configuration">
    <!-- A test configuration -->
    <configuration>
      <!-- Change to your mail server protocol -->
      <property name="mail.transport.protocol" value="smtp"/>
      <!-- Change to the user who will receive mail -->
      <property name="mail.user" value="${account}@gmail.com"/>
      <!-- Change to the SMTP gateway server -->
      <property name="mail.smtp.host" value="smtp.gmail.com"/>
      <!-- The mail server port -->
      <property name="mail.smtp.port" value="465"/>
      <!-- Change to the address mail will be from -->
      <property name="mail.from" value="${account}@whatever.com"/>
      <property name="mail.smtp.auth" value="true"/>
      <property name="mail.smtp.user" value="${account}@gmail.com"/>
      <property name="mail.smtp.password" value="${pw}"/>
      <property name="mail.smtp.ssl.enable" value="true"/>
      <property name="mail.smtp.socketFactory.class" value="javax.net.ssl.SSLSocketFactory"/>
      <!-- Enable debugging output from the javamail classes -->
      <property name="mail.debug" value="false"/>
    </configuration>
  </attribute>
  <depends>jboss:service=Naming</depends>
</mbean>
</server>
```

**Problems**

If you encounter a class loading problem you will need to remove

- `<bamboo-war>/WEB-INF/lib/activation-x.x.x.jar`
- `<bamboo-war>/WEB-INF/lib/mail-x.x.x.jar`

from bamboo.war to avoid the clash with jboss' native libraries.

**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):
If you are using Bamboo 2.0:

<Call name="addHandler">
  <Arg>
    <New class="org.mortbay.jetty.webapp.WebAppContext">
      <Arg name="webApp">
        <SystemProperty name="bamboo.webapp" default="webapp"/>
      </Arg>
      <Arg name="contextPath">/
      </Arg>
    </New>
  </Arg>
</Call>

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="addService">
  <Arg>
    ...
  </Arg>
</Call>
-->

In Bamboo 2.0:

<!--
<New id="resourceID" class="org.mortbay.jetty.plus.naming.Resource">
  <Arg>resourceName</Arg>
  <Arg>
    <New class="org.mortbay.naming.factories.MailSessionReference">
      ...
    </New>
  </Arg>
</New>
-->

If you are experiencing class loading problems with your mail session. Try uncommenting the following line in the web applications context (2.0 only):

<!----<Set name="parentLoaderPriority">true</Set>---->

Support Policies

Welcome to the support policies index page. Here, you'll find information about how Atlassian Support can help you and how to get in touch with our helpful support engineers. Please choose the relevant page below to find out more.

- Bamboo Support Policy
- Bug Fixing Policy
- Deploying Multiple Atlassian Applications in a Single Tomcat Container
To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue under Bamboo. Our friendly support engineers will get right back to you with an answer.

**Bamboo Support Policy**

This page contains details about the scope of Bamboo Support.

**On this page:**

- Build Failures
- Distributed Builds
- EC2
- Plugins
- Related Information

**Build Failures**

Atlassian will provide Troubleshooting Guide(s) and documentation to help customers resolve Bamboo-related issues.

Ultimately, users are responsible for the administration and maintenance of their build systems and infrastructure.

However, if the root cause of the problem is partially or wholly related to Bamboo, we will create a Bug Report or Feature request to address the issue.

Any bug or feature request reported during the course of investigation is subject to Atlassian's Bug Fixing and New Features Policies, as outlined in the How to Get Legendary Support from Atlassian document.

**Distributed Builds**

The pre-requisites outlined in the Technical Overview section of Troubleshooting Guide must be met for server/agent communication to work.

If Atlassian determines that a customer's agent connectivity or communication problem results from a network or environmental factor, it is the customer's responsibility to address this problem and keep their network maintained.

**EC2**

Atlassian does not support custom elastic images (custom AMIs) and recommends using an EBS volume to customise your image if desired. While we are happy to assist with issues related to the elastic agent, we can not help troubleshoot modifications to the default image which are not directly related to Bamboo functionality.

**Plugins**

Atlassian offers support for certain third party plugins as listed in our supported plugins list. For unsupported plugins, issues should be raised with the provider of the plugin.

The following can be classified as being third-party plugins:

- Integration with repositories other than Subversion, CVS and Perforce.
- Third party builders, test and code coverage tools other than what is shipped with Bamboo.

Each plugin's supported status is listed on its page in the Plugin Exchange.

**Related Information**

- Atlassian Bug Fixing Policy
- New Features Policy
- Patch Policy
- How to Get Legendary Support from Atlassian

**Bug Fixing Policy**

**Summary**
Bamboo 3.1 Documentation

- Atlassian Support will help with workarounds and bug reporting.
- Critical bugs will generally be fixed in the next maintenance release.
- Non-critical bugs will be scheduled according to a variety of considerations.

Raising a Bug Report

Atlassian Support is eager and happy to help verify bugs — we take pride in it! Please open a support request in our support system providing as much information as possible about how to replicate the problem you are experiencing. We will replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if they're possible.

Customers and plugin developers are also welcome to open bug reports on our issue tracking systems directly. Use http://jira.atlassian.com for the stand-alone products and http://studio.atlassian.com for JIRA Studio.

When raising a new bug, you should rate the priority of a bug according to our JIRA usage guidelines. Customers should watch a filed bug in order to receive e-mail notification when a "Fix Version" is scheduled for release.

How Atlassian Approaches Bug Fixing

Maintenance (bug fix) releases come out more frequently than major releases and attempt to target the most critical bugs affecting our customers. The notation for a maintenance release is the final number in the version (i.e., the 1 in 3.0.1).

If a bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions) then it will be fixed in the next maintenance release provided that:

- The fix is technically feasible (i.e., it doesn't require a major architectural change).
- It does not impact the quality or integrity of a product.

For non-critical bugs, the developer assigned to fixing bugs prioritises the non-critical bug according to these factors:

- How many of our supported configurations are affected by the problem.
- Whether there is an effective workaround or patch.
- How difficult the issue is to fix.
- Whether many bugs in one area can be fixed at one time.

The developers responsible for bug fixing also monitor comments on existing bugs and new bugs submitted in JIRA, so you can provide feedback in this way. We give high priority consideration to security issues.

When considering the priority of a non-critical bug, we try to determine a 'value' score for a bug which takes into account the severity of the bug from the customer's perspective, how prevalent the bug is and whether roadmap features may render the bug obsolete. We combine this with a complexity score (i.e., how difficult the bug is). These two dimensions are used when developers self serve from the bug pile.

Further reading

See How to Get Legendary Support from Atlassian for more support-related information.

Deploying Multiple Atlassian Applications in a Single Tomcat Container

Deploying multiple Atlassian applications in a single Tomcat container is not supported. Upgrading any of the applications (even for point releases) is likely to break it. There are also a number of known issues with this configuration:

- You may not be able to start up all of the applications in the container, due to class conflicts (in 3rd party libraries bundled with our application) that result from the Atlassian applications sharing a single JVM in the Tomcat container.
- You will not be able to determine the startup order of the applications. Hence, you may experience problems such as JIRA starting before Crowd, rather than vice versa.
- Memory problems are also common as one application may allocate all of the memory in the Tomcat JVM to itself, starving the other applications.

We also recommend that you do not deploy multiple Atlassian applications in a single Tomcat container for a number of practical reasons:

- You will need to shut down Tomcat to upgrade any application.
- If one application crashes, the other applications running in the Tomcat container will be inaccessible.

How to Report a Security Issue

Finding and Reporting a Security Vulnerability

If you find a security bug in the product, please open an issue on http://jira.atlassian.com in the relevant 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 that Atlassian can keep track of the issue and get a patch out as soon as possible.

If you discover a security vulnerability, please attempt to create a test case that proves this vulnerability locally before opening either a bug or a support issue. When creating an issue, please include information on how the vulnerability can be reproduced; see http://confluence.atlassian.com/display/DOC/Bug+Fixing+Policy for general bug reporting guidelines. We will prioritise fixing the reported vulnerability if your report has information on how the vulnerability can be exploited.

**Further reading**

See How to Get Legendary Support from Atlassian for more support-related information.

**New Features Policy**

**Summary**

- We do not publish roadmaps.
- Product Managers review our most popular voted issues on a regular basis.
- We schedule features based on a variety of factors.
- Our Atlassian Bug Fixing Policy is distinct from our Feature Request process.
- Atlassian provides consistent updates on the top 20 feature/improvement requests (in our issue tracker systems).

**How to Track what Features are Being Implemented**

When a new feature or improvement is scheduled, the 'fix-for' version will be indicated in the JIRA issue. This happens for the upcoming release only. We maintain roadmaps for more distant releases internally, but because these roadmaps are often pre-empted by changing customer demands, we do not publish them.

**How Atlassian Chooses What to Implement**

In every major release we aim to implement highly requested features, but it is not the only determining factor. Other factors include:

- Direct feedback from face to face meetings with customers, and through our support and sales channels.
- Availability of staff to implement features.
- Impact of the proposed changes on the application and its underlying architecture.
- How well defined the requested feature is (some issues gain in popularity rapidly, allowing little time to plan their implementation).
- Our long-term strategic vision for the product.

**How to Contribute to Feature Development**

Influencing Atlassian’s release cycle

We encourage our customers to vote on feature requests in JIRA. The current tally of votes is available online in our issue tracking systems, http://jira.atlassian.com and http://studio.atlassian.com. Find out if your improvement request already exists. If it does, please vote for it. If you do not find it, create a new feature or improvement request online.

Extending Atlassian Products

Atlassian products have powerful and flexible extension APIs. If you would like to see a particular feature implemented, it may be possible to develop the feature as a plugin. Documentation regarding the plugin APIs is available. Advice on extending either product may be available on the user mailing-lists, or at our community forums.

If you require significant customisations, you may wish to get in touch with our partners. They specialise in extending Atlassian products and can do this work for you. If you are interested, please contact us.

**Further reading**

See How to Get Legendary Support from Atlassian for more support-related information.

**Patch Policy**

**Patch Policy**

Atlassian will only provide software patches in extremely unusual circumstances. If a problem has been fixed in a newer release of the product, Atlassian will request that you upgrade your instance to fix the issue. If it is deemed necessary to provide a patch, a patch will be provided for the current release and the last maintenance release of the last major version (e.g. JIRA 4.2.4) only.

Patches are issued under the following conditions:
The bug is critical (production application down or major malfunction causing business revenue loss or high numbers of staff unable to perform their normal functions).

A patch is technically feasible (i.e., it doesn't require a major architectural change) OR

The issue is a security issue, and falls under our Security Patch Policy.

Atlassian does not provide patches for non-critical bugs.

Provided that a patch does not impact the quality or integrity of a product, Atlassian will ensure that patches supplied to customers are added to the next maintenance release. Customers should watch for release.

Patches are generally attached to the relevant http://jira.atlassian.com issue.

Further reading

See How to Get Legendary Support from Atlassian for more support-related information.

Security Advisory Publishing Policy

Publication of Security Advisories

When a security vulnerability in an Atlassian product is discovered and resolved, Atlassian will inform customers through the following mechanisms:

- We will post a security advisory in the latest documentation of the affected product at the same time as releasing a fix for the vulnerability. This applies to all security advisories, including severity levels of critical, high, medium and low.
- We will send a copy of all security advisories to the "Technical Alerts" mailing list for the product concerned.
  
  Note: To manage your email subscriptions and ensure you are on this list, please go to my.atlassian.com and click 'Email Prefs' near the top right of the page.
- If the person who reported the vulnerability wants to publish an advisory through some other agency, such as CERT, we will assist in the production of that advisory and link to it from our own.

Early warning of critical security vulnerabilities:

- If the vulnerability is rated critical (see our criteria for setting severity levels) we will send an early warning to the 'Technical Alerts' mailing list approximately one week before releasing the fix. This early warning is in addition to the security advisory itself, described above.
- However, if the vulnerability is publicly known or being exploited, we will release the security advisory and patches as soon as possible, potentially without early warning.

Further reading

See How to Get Legendary Support from Atlassian for more support-related information.

Security Patch Policy

Product Security Patch Policy

Atlassian makes it a priority to ensure the customers' systems cannot be compromised by exploiting vulnerabilities in Atlassian products.

Scope

This page describes when and how we release security patches and security upgrades for our products. It does not describe the whole of disclosure process that we follow. It also excludes Studio, since Studio will always be patched by Atlassian without additional notifications.

Critical vulnerabilities

When a Critical security vulnerability is discovered by Atlassian or reported by a third party, Atlassian will do all of the following:

- Issue a new, fixed release for the current version of the affected product as soon as possible, usually in a few days.
- Issue a binary patch for the current release.
- Issue a binary patch for the latest maintenance release of the previous version of the product.
- Patches for older versions or releases normally will not be issued.

Patches will be attached to the relevant JIRA issue. You can use these patches as a "stop-gap" measure until you upgrade your installation in order to fully fix the vulnerability.

Non-critical vulnerabilities

When a security issue of a High, Medium or Low severity is discovered, Atlassian will do all of the following:

- Include the fix into the next scheduled release, both for the current and previous maintenance versions.
- Where practical, provide new versions of plugins or other components of the product that can be upgraded independently.
You should upgrade your installation in order to fix the vulnerability.

Other Information

Severity level of vulnerabilities is calculated based on Severity Levels for Security Issues.

Visit our general Atlassian Patch Policy as well.

Examples

Example 1: A critical severity vulnerability is found in a (hypothetical current release) JIRA 5.3.2. The last bugfix release in 5.2.x branch was 5.2.3. In this case, a patch will be created for 5.3.2 and 5.2.3. In addition, new bugfix releases, 5.3.3 and 5.2.4, which are free from this vulnerability, will be created in a few days.

Example 2: A high or medium severity vulnerability is found in the same release as in the previous example. The fix will be included into the currently scheduled releases 5.3.3 and 5.2.4. Release schedule will not be brought forward and no patches will be issued. If the vulnerability is in a plugin module, then a plugin upgrade package may still be supplied.

Further reading

See How to Get Legendary Support from Atlassian for more support-related information.

Severity Levels for Security Issues

Severity Levels

Atlassian security advisories include a severity level. This severity level is based on our self-calculated CVSS score for each specific vulnerability. CVSS is an industry standard vulnerability metric. You can learn more about CVSS at FIRST.org web site.

CVSS scores are mapped into the following severity ratings:

- Critical
- High
- Moderate
- Low

An approximate mapping guideline is as follows:

<table>
<thead>
<tr>
<th>CVSS score range</th>
<th>Severity in advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 2.9</td>
<td>Low</td>
</tr>
<tr>
<td>3 – 5.9</td>
<td>Medium</td>
</tr>
<tr>
<td>6.0 – 7.9</td>
<td>High</td>
</tr>
<tr>
<td>8.0 – 10.0</td>
<td>Critical</td>
</tr>
</tbody>
</table>

Below is a summary of the factors which illustrate types of vulnerabilities usually resulting in a specific severity level. Please keep in mind that this rating does not take into account details of your installation.

Severity Level: Critical

Vulnerabilities that score in the Critical range usually include:

- 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.

For critical vulnerabilities, is advised that you patch or upgrade as soon as possible, unless you have other mitigating measures in place. For example, if your installation is not accessible from the Internet, this may be a mitigating factor.

Severity Level: High

Vulnerabilities that score in the High range usually have the following characteristics:

- The vulnerability is difficult to exploit.
- Exploitation does not result in elevated privileges.
- Exploitation does not result in a significant data loss.

Severity Level: Moderate
Vulnerabilities that score in the Moderate range usually have the following characteristics:

- Denial of service vulnerabilities that are difficult to set up.
- 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 where exploitation provides only very limited access.

**Severity Level: Low**

Vulnerabilities in the Low range typically have very little impact on an organisation’s business. Exploitation of such vulnerabilities usually requires local or physical system access.

**Further reading**

See [How to Get Legendary Support from Atlassian](#) for more support-related information.

**Usage FAQ**

- 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 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
- Do I have to upgrade all remote agents for Bamboo Release 2.1.2?
- Enable User Management debug logging 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 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
- Troubleshooting SQL Exceptions - Detailed Hibernate Logging
- Using Bamboo For Release Management
- Using Bamboo with Clover
- Working with Sun JAVA libraries
- Bamboo indicates that my Ant or Maven builds failed, even though they were successful
- How can I pass bamboo variables to my build script
- Finding the Support Entitlement Number (SEN)
- How do I manually set the version of new Subversion workspaces?
- Securing your repository connection
- **Backing 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. You can delete bamboo's working copy folder `<bamboo-home>/xml-data/build-dir` beforehand to reduce the size. Zip this backup.
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
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:
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: 
Triggerring a Build when another Build finishes

**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,

```xml
<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,

```xml
<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 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 straight forward 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>(Optional) reduce the size of this zip by deleting the xml-data/build-dir - this directory only contains working copies of checked out sources. zip &lt;bamboo-production-home&gt; directory, the embedded database will be part of this zip.</td>
<td>(Optional) reduce the size of this zip by deleting the xml-data/build-dir - this directory only contains working copies of checked out sources. zip &lt;bamboo-production-home&gt; directory. create a backup with the native tools provided by your DB.</td>
</tr>
</tbody>
</table>

3. restart production bamboo
4. transfer the home.zip to your cloned instance and unzip into <bamboo-clone-home>
5. (External DB only) create a new database for the cloned instance and import the db dump.
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. (External DB only) edit <bamboo-clone-home>/bamboo.cfg.xml and enter the new database connection details and credentials
8. point bamboo.home of your cloned instance to the unzipped <bamboo-clone-home> directory
9. 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

   1. Click on the icon highlighted in red.
   2. Enter an identifiable name for the connection, e.g. Bamboo_db.

2. Select JDBC Driver
1. Select **HSQLDB Embedded** from the drop down list.
2. Click on **Load Driver Files** button to open a file chooser in which you should select the driver file(s). If multiple files must be loaded then hold CTRL and select all of them at once. Driver files are normally packaged as JAR or ZIP files. In addition you can load a directory with driver classes.

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
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 'Operations' column.
4. The 'User Details' screen will be displayed. Type a new password in the 'Password' and 'Confirm Password' fields.
5. If you have configured SMTP email on your Bamboo server, the user will automatically receive an email containing their new password.
6. To get around the email problem, in the 'Email' field, enter an invalid email address. E.g. foobar@foobaremailaddress.foobar
7. Delete the users, Jabber Address so that he does not receive Jabber notifications on build events.
8. Click on the Save button.
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.

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:

```
2008-02-21 09:13:39,890 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table USER_COMMIT add constraint FKF8936C2BA5B29F foreign key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY
2008-02-21 09:13:39,891 ERROR \[main\] \[SchemaUpdate\] Constraint already exists in statement
\[alter table USER_COMMIT add constraint FKF8936C2BA5B29F foreign key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY\]
2008-02-21 09:13:39,892 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table USER_COMMIT add constraint FKF8936C2BF6E84F foreign key (AUTHOR_ID) references AUTHOR
2008-02-21 09:13:39,892 ERROR \[main\] \[SchemaUpdate\] Constraint already exists in statement
\[alter table USER_COMMIT add constraint FKF8936C2BE0C684F foreign key (AUTHOR_ID) references AUTHOR\]
2008-02-21 09:13:39,893 ERROR \[main\] \[SchemaUpdate\] Unsuccessful: alter table USER_COMMENT add constraint FK19DA09CBA5B29F foreign key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY
2008-02-21 09:13:39,894 ERROR \[main\] \[SchemaUpdate\] Constraint already exists in statement
\[alter table USER_COMMENT add constraint FK19DA09CBA5B29F foreign key (BUILDRESULTSUMMARY_ID) references BUILDRESULTSUMMARY\]
```

In Bamboo 2.0 we introduced compatibility with Oracle and MS SQL Server, this meant we had to make a few changes to the Bamboo Database schema and as a side-effect 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 CronTrigger Tutorial 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.

```
INFO wrapper 2009/01/28 15:24:34 Wrapper Process has not received any CPU time for 11 seconds. Extending timeouts.
```

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, `"-DDISABLE_AGENT_AUTO_CAPABILITY_DETECTION=true"`, to the Bamboo server. Read [Configuring System Properties](#) for information on how to do this.

**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.

```xml
<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"/>
  </properties>
  <failure type="junit.framework.AssertionFailedError" message="Should not have any errors. [Perforce client error:, Connect to server failed; ">
    jUnit.framework.AssertionFailedError: Should not have any errors. [Perforce client error:, Connect to server failed; check $P4PORT., TCP connect to keg failed., keg: host unknown.]
  </failure>
</testsuite>
```
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.battery.JUnitBattery.executeJUnit(JUnitBattery.java:242)
at org.apache.maven.surefire.battery.JUnitBattery.executeJUnit(JUnitBattery.java:216)
at org.apache.maven.surefire.Surefire.executeBattery(Surefire.java:215)
at org.apache.maven.surefire.Surefire.run(Surefire.java:163)
at org.apache.maven.surefire.Surefire.run(Surefire.java:87)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at org.apache.maven.surefire.SurefireBooter.runTestsInProcess(SurefireBooter.java:313)
at org.apache.maven.surefire.SurefireBooter.run(SurefireBooter.java:221)
at org.apache.maven.test.SurefirePlugin.execute(SurefirePlugin.java:371)
at org.apache.maven.plugin.DefaultPluginManager.executeMojo(DefaultPluginManager.java:412)
at org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoals(DefaultLifecycleExecutor.java:534)
at org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoalWithLifecycle(DefaultLifecycleExecutor.java:454)
at org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeGoalAndHandleFailures(DefaultLifecycleExecutor.java:306)
at org.apache.maven.lifecycle.DefaultLifecycleExecutor.executeTaskSegments(DefaultLifecycleExecutor.java:273)
at org.apache.maven.lifecycle.DefaultLifecycleExecutor.execute(DefaultLifecycleExecutor.java:140)
at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:322)
at org.apache.maven.DefaultMaven.execute(DefaultMaven.java:115)
at org.apache.maven.cli.MavenCli.main(MavenCli.java:256)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at org.codehaus.classworlds.Launcher.launchEnhanced(Launcher.java:315)
at org.codehaus.classworlds.Launcher.launch(Launcher.java:255)
at org.codehaus.classworlds.Launcher.mainWithExitCode(Launcher.java:430)
at org.codehaus.classworlds.Launcher.main(Launcher.java:375)
</failure>
</system-out>
PerforceSyncCommand.command: /usr/local/bin/p4
Click here to download the XML report.

A passed JUnit XML report, that is successfully parsed by Bamboo.

<?xml version="1.0" encoding="UTF-8" ?>
<testsuite errors="0" skipped="0" tests="1" time="0.045" failures="0"
  name="com.atlassian.bamboo.labels.LabelManagerImplTest">
  <properties>
    <property value="Java(TM) 2 Runtime Environment, Standard Edition" name="java.runtime.name"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/i386" name="sun.boot.library.path"/>
    <property value="1.5.0_07-b03" name="java.vm.version"/>
    <property value="Sun Microsystems Inc." name="java.vm.vendor"/>
    <property value="http://java.sun.com/" name="java.vendor.url"/>
    <property value=";" name="path.separator"/>
    <property value="Java HotSpot(TM) Client VM" name="java.vm.name"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="US" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="1386" name="os.arch"/>
    <property value="" name="java.io.tmpdir"/>
    <property value="Sun Microsystems Inc." name="java.vm.specification.vendor"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="Linux" name="os.name"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="1386" name="os.arch"/>
    <property value="" name="java.io.tmpdir"/>
    <property value="Sun Microsystems Inc." name="java.vm.specification.vendor"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="Linux" name="os.name"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
    <property value="1.5.0_07-b03" name="java.runtime.version"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN/bamboo-core" name="basedir"/>
    <property value="/usr/java/jdk1.5.0_07/jre/lib/endorsed" name="java.endorsed.dirs"/>
    <property value="" name="file.encoding.pkg"/>
    <property value="" name="user.country"/>
    <property value="unknown" name="sun.os.patch.level"/>
    <property value="/opt/bamboo-data/bamboohome/xml-data/build-dir/BAM-MAIN" name="user.dir"/>
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><code>-b</code></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><code>-kv</code></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.jmxremote.port=<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.

---

**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.

**Parameters**

Please add the following parameters to the run command line in `bamboo.sh`

```
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -Xloggc:/path/to/gc.log
```

**Note:** Remember to substitute `/path/to/gc.log` with a meaningful file path on your server.

**GC log file location**

The garbage collection traces and the heap dumps are in `<bamboo-install>/gc.log`

**Additional Note**

The `-XX:+PrintGCTimeStamps` flag, prints when GCs happen relative to the start of the application.

Some helpful links:


**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 -
   
   ```
   ```

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 Unices) 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).
```

### Jstack

Sun JDK 1.5 and above ship with native tool called `jstack` to perform thread dump. To use the tool find the Proccess ID and execute the command:

```
jstack <ProcessId>
```

### 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 incompatibile 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:

   ```markdown
   INSERT INTO USERS VALUES(1
   ```

   To find the administrator login entry:

   ```markdown
   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:

   ```sql
   INSERT INTO USERS
   VALUES(1,'admin','x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3p811:26:18.504000000','admin')
   ```

   This step makes 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:

   ```text
   x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3p837A==
   ```

   Paste the `admin` password hash between the `''` characters of their existing PASSWORD_HASH. The new administrator login entry should look like:

   ```sql
   INSERT INTO USERS
   VALUES(1,'USERNAME','x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3p837A==','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:

```sql
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 paste a hash, rather than the plain password, over the existing password. Below is the hash for the password `admin`

```text
x61Ey612Kl2gpFL56FT9weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTrw10Wq3+4qQyB+XURPWx1ONxp3Y3p837A==
```

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 = 'x61Ey612k12gpFl56FT3weDnpSo4AV8j8+qx2AuTHdRyY036xxzTTw10Wq3+4qQyB+XURPWx1ONxp3Y3pB37A==' 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

```
#!/bin/bash
echo "ERROR build xyz failed" >&2
```

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.

```
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

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.
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

One-click Clover Integration

Clover has been seamlessly integrated with Bamboo from Bamboo 2.4 and later. Clover reports can be activated in the Builder configuration screen. Please see Enabling the Clover Plugin for further details.

To configure Clover activity refer to Clover Reference Guides for your builder:

- Clover for Ant
- Clover for Maven 2

Classic Clover Integration

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 Configuring Tasks);
   - 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 the path to your clover.xml file "relative to your plan’s root directory" (e.g. your plan’s root directory is /home/bamboouser/bamboo-home/xml-data/build-dir/MY_PLAN/ and you would enter target/clover/site/clover.xml). Please do not specify the absolute path.

For further details, please see Configuring Tasks.

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.

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:

```xml
.......
<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 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
   ```

Bamboo indicates that my Ant or Maven builds failed, even though they were successful
If your plan's build logs indicate that your Maven or Ant builds are passing but Bamboo is reporting them as failed (or vice-versa), it could be that:

- Bamboo is not finding 'BUILD SUCCESS' in your build logs
- Bamboo is finding 'BUILD FAILED' in your build logs when it should not be doing so. (This marker is not used in Maven.)
- Your builds are returning a non-zero return code. (For example, the build log will indicate 'Build process for 'ABC Application - XYZ Build' returned with return code = 1.)

If your builds produce atypical or non-standard output, you can make Bamboo check for text other than 'BUILD SUCCESS' or 'BUILD FAILED' in your build logs. An additional system property is available to specify how far back in the logs Bamboo checks for these text markers.

<table>
<thead>
<tr>
<th>System Property</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>atlassian.bamboo.builder.successMarker</td>
<td>Specifies the text (or string) that Bamboo looks for in the build log to determine if the build was successful</td>
<td>BUILD SUCCESS</td>
</tr>
</tbody>
</table>
atlassian.bamboo.builder.failedMarker | Specifies the text (or string) that Bamboo looks for in the build log to determine if the build failed | BUILD FAILED
SUCCESS_MESSAGE_LINES | Specifies the number of lines from the end of the builder log in which to check for the values of atlassian.bamboo.builder.successMarker or atlassian.bamboo.builder.failedMarker | 250

For instructions on how to configure a system property, please refer to the Configuring System Properties page.

**How can I pass bamboo variables to my build script**

**Bamboo global and build specific variables** can be referred to in build scripts or maven pom.xml. Bamboo variables are not directly available in the builder execution context however. They can be passed as parameters to the builder.

**Maven**

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>  
<packaging>jar</packaging>  
<version>1.1.\$\{env.bambooBuildNumber\}-SNAPSHOT\$</version>  
...  
```

You can then specify the following in the 'Goal' field of your build plan:

```
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.

**Ant**

You can pass bamboo variables as ant parameters along with ant targets like

```
clean test -Dbuild.key=${bamboo.buildKey}
```

In your ant build script just refer to this variable

```
...  
<echo message=\"bamboo.buildKey = \${build.key}\"/>  
...  
```

**Finding the Support Entitlement Number (SEN)**

If you have a current Bamboo maintenance license under another account please supply the details of the licensee and the current Support Entitlement Number (SEN)

- Your Support Entitlement Number (SEN) is listed on the third page of your Atlassian Invoice.

OR

- Log into http://my.atlassian.com to find the SEN for a specific license
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. Bamboo automatically upgrades any source code it checks out, to be compatible with a particular version of Subversion. If you use an older Subversion client to access the code checked out by Bamboo, you will need to force any new Subversion workspaces to be created with the SVN version that you wish to retain. Otherwise, if you then use an older Subversion client to access this code, any Bamboo builds on that code may fail.

If you want to prevent Bamboo from automatically upgrading any source code checked out, you will need to run Bamboo with the following system property:

```
-Dbamboo.svn.wc.format=X.X
```

where \( X.X \) is the SVN version that you want to retain for your code. Valid values for this parameter are 1.3, 1.4, 1.5 and 1.6.

To change this parameter for your Bamboo instance:

- Add the parameter with a `-D` prefix and appropriate value, in your command line when starting Bamboo. e.g. `-Dbamboo.svn.wc.format=1.5`, or

⚠️ 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.

For example, setting this parameter to 1.5 tells Bamboo to;

- check out code to version 1.5 if no working copy exists, and
- not to automatically upgrade any already checked out code of an existing working copy to be compatible with Subversion 1.6.

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**

**Trouble shooting**
You can test the svn+ssh connection from the command line. First you need to tell the svn command line client which key file to use:

```
$ export SVN_SSH="ssh -i /absolute/path/to/private/key"
```

Then you can test the connection with

```
$ svn list svn+ssh://<svn-server>/Absolute/Path/To/Repository/[Module]
```
build log
build queue
build result
build strategy
build telemetry
capability
child
committer
custom capability
default image
elastic agent
elastic bamboo
elastic block store
elastic image
elastic instance
executable
favourites
global permission
Job
label
local agent
parent
permission
Plan
plan permission
project
queue
reason
remote agent
remote agent supervisor
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

A Bamboo agent is a service that provides capabilities to run Job builds. There are two types of Bamboo agents:

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

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 and can only run builds for Jobs whose requirements are met by the agent's capabilities.

For more information, see:

- Configuring Agents
- Configuring Agents and Capabilities
- Configuring a Job's Requirements

agent-specific capability

An agent-specific capability is a capability that 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 Configuring Agents and Capabilities and Configuring Capabilities for more information.

artifact

Artifacts are files created by a Job build (e.g. JAR files). Artifact definitions are used to specify which artifacts to keep from a build and are configured for individual Jobs.

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 the execution of either a Plan or a Job. The execution of a Plan is referred to as a 'Plan build' and that of a Job is a 'Job build'.

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.

**build log**

Every *build* has a *build log*. A *build log* is a permanent record of all the output generated by compiling the *Job*’s source-code and executing the tests.

**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.
- *'Incomplete'* — the build was not completed, e.g. it may have been stopped manually.

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 strategy**

The build strategy for a *Plan* determines how a build is triggered. The build strategy is defined when configuring the source repository for a plan.

**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 for more information.

**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 Jobs will be built by a particular agent. For example, if the builds for a particular Job 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 Job.

- To create a new custom capability in your Bamboo system, see Configuring a new Custom Capability.
- To specify a Job’s requirement for a custom capability, see Configuring a Job’s Requirements.

default image

Atlassian maintains a public ‘default’ elastic image (known as the ‘default image’) in the Amazon Simple Storage Service (S3). Bamboo’s Elastic Bamboo feature uses this image by default and in your list of elastic image configurations, this image will have ‘(default)’ appended to its name.

The ‘default image’ uses:

- the Fedora 8 (Linux) operating system,
- the Bamboo elastic agent for Bamboo 3.1.x

and has the following default packages/capabilities:

<table>
<thead>
<tr>
<th>Default packages/capabilities</th>
<th>Path/value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Builders</strong></td>
<td></td>
</tr>
<tr>
<td>Ant (Ant) (version 1.7.1)</td>
<td>/opt/apache-ant-1.7.1</td>
</tr>
<tr>
<td>Bash (Command)</td>
<td>/bin/bash</td>
</tr>
<tr>
<td>Grails 1.2.2 (Grails)</td>
<td>/opt/grails-1.2.2</td>
</tr>
<tr>
<td>Grails 1.3.1 (Grails)</td>
<td>/opt/grails-1.3.1</td>
</tr>
<tr>
<td>Maven 1 (Maven) (version 1.0.2)</td>
<td>/opt/maven-1.0.2</td>
</tr>
<tr>
<td>Maven 1.1 (Maven)</td>
<td>/opt/maven-1.1</td>
</tr>
<tr>
<td>Maven 2 (Maven 2.x) (version 2.0.11)</td>
<td>/opt/maven-2.0</td>
</tr>
<tr>
<td>Maven 2.1 (Maven 2.x) (version 2.1.0)</td>
<td>/opt/maven-2.1</td>
</tr>
<tr>
<td>Maven 2.2 (Maven 2.x) (version 2.2.1)</td>
<td>/opt/maven-2.2</td>
</tr>
<tr>
<td><strong>JDKs</strong></td>
<td></td>
</tr>
<tr>
<td>JDK 1.5 (version 5u22)</td>
<td>/opt/jdk-5 (for Java Home)</td>
</tr>
<tr>
<td>JDK 1.6 (version 6u21)</td>
<td>/opt/jdk-6 (for Java Home)</td>
</tr>
<tr>
<td><strong>Others</strong></td>
<td></td>
</tr>
<tr>
<td>Mercurial 1.4 Executable (version 1.4.1)</td>
<td>/usr/bin/hg</td>
</tr>
</tbody>
</table>

* The agent jar also contains the libraries required to connect to Subversion and CVS.

Be aware that the default packages/capabilities above may change with each major release of Bamboo. There is a new default image (with its own AMI ID) for each new version of Bamboo. However, older default images will still be available for use.

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 elastic instance of an 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 Job builds. Elastic Bamboo uses a remote agent AMI (Amazon Machine Image) to create instances of remote agents in the Amazon EC2. Job builds can be run on these 'elastic agents' in the same way that Job builds are run non-elastic agents.

elastic block store

The Amazon Elastic Block Store (EBS) provides 'EBS volumes' which can attach to EC2 instances. EBS volumes (and the 'EBS snapshots' created from these volumes) provide persistent storage for your elastic instances.

If you have relatively static resources required for building your Bamboo Jobs (such as, source code checkouts and Maven repository artifacts), you can add these to an EBS volume. From this volume, you can create an EBS snapshot, which effectively records the 'state' of an EBS volume at a given point in time.

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, an elastic image is equivalent to an operating system running on a computer's boot hard drive and elastic instances would be the software that runs on this operation system.

Each elastic image in the Amazon S3 has its own unique identifier, known as an AMI ID.

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 a running instance of an elastic image. One elastic instance is created whenever an elastic image is started. Hence, starting one elastic image multiple times, results in the creation of multiple elastic instances. Each time an elastic instance is created, one elastic agent is created on that instance.

Conceptually, an elastic instance can be thought of as a computer. The elastic agent's processes are run on this computer and the elastic image is the boot hard drive. Unlike computers, however, elastic instances are temporary and stateless. When an elastic instance is shut down:

- Any changes that an elastic instance makes to the boot hard drive (e.g. agent log file) will not persist
- Any customisations to the instance itself will also be lost.

✅ The Amazon Elastic Block Store can provide persistent storage for your elastic instances.

executable

An executable is a program external to Bamboo used to automate processes. Generally, executables compile source code to generate compiled executable files (referred to as artifacts in Bamboo). Ant, Maven, MSBuild or PHPUnit are just some examples of executables that can be used as part of your build process.

New executables can be defined as capabilities in Bamboo. Once an executable has been defined in Bamboo, it can be configured as part of a Task.

See Configuring a new Executable.

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.

Job
A Job is a single build unit within a Plan. One or more Jobs can be organised into one or more Stages. A Job is made up of one or more Tasks. A Job defines:

- what gets built (i.e. the source code repository) — this can be a custom, Job-specific source repository or the 'default source repository' of the Plan that contains this Job;
- which agent capabilities are required for the build (based on Job-specific requirements and requirements of the Job's Tasks);
- what Tasks make up the Job and the order in which they are executed;
- what artifacts the Job's build will produce;
- any labels with which the build result or build artifacts will be tagged;

Each new Plan created in Bamboo contains at least one Job known as the 'Default Job'.

Projects and plans can only be configured by Bamboo administrators (see Creating a Plan).

**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 Plan defines everything about your entire build process. Plans:

- consist of one or more Jobs, which are organised into one or more Stages;
- contain a single 'Default Job' in a single Stage, after creating a new Plan;
- define default settings for what gets built by Jobs in the Plan (i.e. the 'default source repository');
- define how the Plan's build is triggered;
- who will be notified of the Job's build result;
- define who has permission to view and perform various actions on the Plan and its Jobs.

Every plan belongs to a Project.

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 Plan and its Jobs. 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. Projects enable you to easily group and identify plans which are logically related to each other. They are especially useful when generating reports across multiple Plans.
queue

See build queue.

reason

A build's reason is the way in which the build was triggered.

There are a number of methods by which Bamboo can 'trigger' (i.e. execute) a Plan's build,

- **Build Strategy:**
  - Code updated — a Plan's build is triggered whenever one or more authors checks in code.
  - Scheduled build — a Plan's build is triggered at scheduled times or specified time intervals.
  - Manual build — a Plan's build is only ever triggered manually.
  - Initial clean build — a Plan's build is triggered immediately after a new Plan has been created.
- **Build Dependency:**
  - Dependency — a build can be triggered whenever another Plan's build is successfully completed.

The trigger method for a Plan's build is listed in the 'Reason' column on the Dashboard.

Note that these trigger methods can only be configured by a Bamboo administrator. For more information please see Triggering a Plan Build.

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 OS X:**
  - 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
  - 64 bit

requirement

A requirement is specified in a Job or a Task. It defines the capabilities which are required by an agent to build that Job. A Job inherits all of the requirements specified in its Tasks.

Together, capabilities and requirements control which agents can execute builds for particular Jobs. Each Job can only be built by agents whose capabilities meet the Job's requirements. See Configuring a Job's Requirements for more information.

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 Configuring Agents and Capabilities and Configuring Capabilities.

Stage

Stages group (or 'map') Jobs to individual steps within an entire Plan's build process. For example, you may have an overall Plan build process that comprises a compilation step, followed by several test steps, followed by a deployment step. You can create separate Bamboo Stages to represent each of these steps. Stages have the following characteristics:

- A Stage may contain one or more Jobs;
- Depending on the availability of Bamboo agents, all Jobs in a Stage can be processed in parallel;
- Stages are processed consecutively within a Plan, one at a time;
- All Jobs in a Stage must be built and succeed before Bamboo builds any Jobs in the next Stage. If any Job fails in a Stage, no further Stages in the Plan will be processed and the Plan's build will fail.

Each new Plan created in Bamboo contains at least one Stage (to house the Default Job) and is known as the 'Default Stage'. Stages can only be configured by Bamboo administrators.

Task

A Task is an operation that is run on a Bamboo working directory using an executable. Tasks are run sequentially within a single Job. Once a Task is defined in the Bamboo system, it can then be specified in Jobs by a Bamboo administrator. A Job can be configured to execute a number of Tasks, on the same working directory. For example, before executing a Maven goal, the user could substitute specific files within the working directory, substitute version numbers or execute a Script.

See Configuring Tasks.

triggering

There are a number of methods by which Bamboo can 'trigger' (i.e. execute) a Plan's build,

- **Build Strategy:**
  - Code updated — a Plan's build is triggered whenever one or more authors checks in code.
  - Scheduled build — a Plan's build is triggered at scheduled times or specified time intervals.
  - Manual build — a Plan's build is only ever triggered manually.
  - Initial clean build — a Plan's build is triggered immediately after a new Plan has been created.

- **Build Dependency:**
  - Dependency — a build can be triggered whenever another Plan's build is successfully completed.

The trigger method for a Plan's build is listed in the 'Reason' column on the Dashboard.

Note that these trigger methods can only be configured by a Bamboo administrator. For more information please see Triggering a Plan Build.

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.

Bamboo Resources

Resources for Evaluators

- Free Trial
- Feature Tour

Resources for Administrators

- Bamboo Knowledge Base
- Bamboo FAQ
- Tips of the Trade
- Guide to Installing an Atlassian Integrated Suite
- The big list of Atlassian gadgets

Downloadable Documentation

- Bamboo documentation in PDF, HTML or XML formats

Plugins
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.

Support

- Atlassian Support
- Support Policies

Training

- Atlassian Training

Forums

- Bamboo Announcements | subscribe
- Bamboo General Forum | subscribe
- Bamboo Developers Forum | subscribe

Feature Requests

- Issue Tracker and Feature Requests for Bamboo

Bamboo Development Hub

![Looking for existing plugins? See the existing plugins and extensions written by the community and partners at the Atlassian Plugin Exchange.](image)

Getting Started

There are two main ways to develop with Bamboo — using our REST APIs or developing a plugin. If you are integrating Bamboo with another application or scripting interactions with the Bamboo server, you will most likely want to use the REST APIs. If you wish to extend Bamboo's functionality, a plugin may be the answer. To get started writing plugins, we recommend you to download the Plugins SDK and follow the instructions to setup a plugin development environment.

Main Topics

**Atlassian Plugin SDK**
Get started by setting up your Atlassian plugin development environment.

**Gadgets**
Learn how to write Gadgets to expose or consume content in Atlassian applications.

**Plugin Modules**
Learn how to build plugin modules that integrate with your other applications, improve your builds and experience with Bamboo

- User Interface Plugin Modules
- Build Lifecycle Plugin Modules
- Notification Plugin Modules
- System Plugin Modules
- Bamboo Event Listeners

**REST API**
Bamboo exposes its data via REST services. Learn how to use the REST API's to integrate Bamboo with your other applications or script interactions with the Bamboo server.

Recent Changes by Version

- Changes for Bamboo 3.1
- Changes for Bamboo 3.0

Atlassian Development Hubs
Plugin Modules

**Build Lifecycle Plugin Modules**

- Build Agent Requirement Filter Module
- Build Complete Action Module
- Builder Plugin Module
- Build Processor Module
- Build Processor Server Module
- Build Trigger Condition Module
- Command Decorator Module
Post-Build Completed Action Module
Post-Chain Action Module
Post-Job Action Module
Post-Stage Action Module
Pre-Build Action Module
Pre-Build Queued Action
Pre-Chain Action Module
Pre-Job Action Module
Pre-Stage Action Module
Source Repository Module
Task Plugin Module
Trigger Reason Module

**Notification Plugin Modules**

Notification Condition Module
Notification Recipient Module
Notification Type Module

**User Interface Plugin Modules**

Additional Build Configuration Module
Report Module
Web Item Module
Web Panel Module
Web Repository Viewer Module
Web Section Module
XWork Plugin Module

**System Plugin Modules**

Capability Type Module
Component Module
Index Reader Module
Plan Deletion Interceptor Action Module
Post Build Index Writer Module
Post Chain Index Writer Module
Servlet Context Listener Plugin Module
Bamboo Plugin Guide

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:

```xml
<!-- the plugin key must be unique, think of it as the 'package' of the plugin -->
<atlassian-plugin key="com.atlassian.plugin.sample" name="Sample Plugin">
  <!-- a short block describing the plugin itself -->
  <plugin-info>
    <description>This is a brief textual description of the plugin</description>
    <!-- the version of the plugin -->
    <version>1.1</version>
    <!-- the versions of the application this plugin is for -->
    <application-version min="3.0" max="3.0"/>
    <!-- details of the plugin vendor -->
    <vendor name="Atlassian Software Systems Pty Ltd" url="http://www.atlassian.com"/>
  </plugin-info>
  . . . 1 or more plugin modules . . .
</atlassian-plugin>
```

Each plugin has a plugin key which is unique among all plugins (eg "com.atlassian.plugin.sample"). Semantically this equates to the package of a Java class. Each module within the plugin also has a module key which is unique within the plugin (eg "myreport"). Semantically this equates to the class name of a Java class.

The plugin key + module key are combined to make the complete key of the plugin module (combining the examples above, the complete key would be "com.atlassian.plugin.sample:myreport"). Note: a : is used to separate the plugin key from the module key.

How does the Bamboo plugin system work?

A Chain (what users see as a Plan in the UI) consists of one or more Stages and each Stage consists of one or more Jobs. Before progressing to the next Stage, every Job in the previous Stage must have completed successfully. There is no order to Jobs within a Stage so when a Stage is executed, all Jobs in that Stage are placed in the queue immediately.

All triggering strategies now work at the Chain level. Chains do not take Job information into account when deciding whether to trigger. If repository polling or repository trigger strategies are selected, only changes detected against the Chain's are used to determine if the build will be triggered.

Below is a diagram which shows Bamboo's build process flow, as well as the available plugin modules you can build.

Build Process For Chain
Build Process Per Job

Further Notes
1. **Build is Triggered** — Changes are detected via polling (against the Chain's repository configuration only), manual builds, etc.
2. **Preperation for Build** — The Build Number is determined then ResultsSummary objects and BuildContexts are created for the Chain and every Job within the Chain. The build number information, latest revision keys and ResultSummaries are persisted to the database.
3. **Job Queued** — The server decides which agents can execute the job and queues the job.
4. **Agent Picks Up Job** — The capability context is set. The job is removed from queue. The agent begins looping through the build tasks.
5. **Retrieve Source** — 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.
6. **Prepare Job** — The agent runs the PrepareBuildTask. This begins streaming the logs back to the server. The agent also runs the CustomPreBuildAction plugin point.
7. **Executes the Job** — Timer begins. The agent runs the Builder#executeBuild. The CommandDecorator plugin will be run against the command line just before execution. After the Builder has been run, all the CustomBuildProcessors are run. Timer is stopped.
8. **Process Job Result** — The server runs CustomBuildProcessorServer, checks if the job has passed and saves and indexes the result. The server also fires off the BuildCompletedEvent and PostBuildCompletedEvent events, and executes any CustomPostBuildCompletedActions.
9. **On BuildCompletedEvent (performed by the server)** — Notifications are sent in this phase.

The following types of plugin modules are supported by Bamboo:

### Build Lifecycle Plugin Modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo Version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildAgentRequirementFilter</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td>Build Agent Requirement Filter Module</td>
<td>Customises the list of Agents a plan can be built on.</td>
</tr>
<tr>
<td>buildCompleteAction</td>
<td>1.x, 2.x</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>builder</td>
<td>1.x</td>
<td>1.0</td>
<td>Builder Plugin Module</td>
<td>Add new builders to Bamboo.</td>
</tr>
<tr>
<td>buildProcessor</td>
<td>1.x</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>1.x</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>command decorator</td>
<td>1.x</td>
<td>2.3</td>
<td>Command Decorator Module</td>
<td>Allows additional parameters to be added to the command line of a builder before a build occurs.</td>
</tr>
<tr>
<td>preBuildQueuedAction</td>
<td>1.x, 2.x</td>
<td>2.1</td>
<td>Pre-Build Queued Action Module</td>
<td>Prepends a custom synchronous process to the build. Occurs before a build is queued.</td>
</tr>
<tr>
<td>preBuildAction</td>
<td>1.x</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>postBuildCompletedAction</td>
<td>2.x</td>
<td>2.6</td>
<td>Post-Build Completed Action Module</td>
<td>Add a custom asynchronous 'build context'-aware action after the build process has completed and the build results have been saved.</td>
</tr>
<tr>
<td>preChainAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Chain Action Module</td>
<td>Prepends a custom synchronous process before the chain starts executing</td>
</tr>
<tr>
<td>postChainAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Chain Action Module</td>
<td>Prepends a custom synchronous process after the chain finishes executing</td>
</tr>
<tr>
<td>preStageAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Stage Action Module</td>
<td>Prepends a custom synchronous process before the stage starts executing</td>
</tr>
<tr>
<td>postStageAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Stage Action Module</td>
<td>Prepends a custom synchronous process after the stage finishes executing</td>
</tr>
<tr>
<td>preJobAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Job Action Module</td>
<td>Prepends a custom synchronous process before the job starts executing</td>
</tr>
<tr>
<td>postJobAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Job Action Module</td>
<td>Prepends a custom synchronous process after the job finishes executing</td>
</tr>
<tr>
<td>Repository</td>
<td>1.x</td>
<td>1.1</td>
<td><strong>Source Repository Module</strong></td>
<td>Add a custom source repository.</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>triggerReason</td>
<td>1.x</td>
<td>2.0</td>
<td><strong>Trigger Reason Module</strong></td>
<td>Add a new trigger reason.</td>
</tr>
</tbody>
</table>

**Notification Plugin Modules**

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notification-condition</td>
<td>1.x, 2.x</td>
<td>1.1 to 2.2</td>
<td><strong>Notification Condition Module</strong></td>
<td>Add new notification condition.</td>
</tr>
<tr>
<td>notificationType</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td><strong>Notification Type Module</strong></td>
<td>Add a new notification type</td>
</tr>
<tr>
<td>notificationRecipient</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td><strong>Notification Recipient Module</strong></td>
<td>Add a new notification recipient type.</td>
</tr>
</tbody>
</table>

**Bamboo Event Listeners**

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bambooEventListener</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td><strong>Bamboo Event Listeners</strong></td>
<td>Register an event listener against any Bamboo Events.</td>
</tr>
</tbody>
</table>

**User Interface Plugin Modules**

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalBuildConfigurationPlugin</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td><strong>Additional Build Configuration Module</strong></td>
<td>Register additional configuration UI on the Miscellaneous tab of the Configuration of a Job or Plan.</td>
</tr>
<tr>
<td>report</td>
<td>2.x</td>
<td>1.0</td>
<td><strong>Report Module</strong></td>
<td>Defines a report of build telemetry data.</td>
</tr>
<tr>
<td>web-item</td>
<td>1.x, 2.x</td>
<td>1.0</td>
<td><strong>Web Item Module</strong></td>
<td>Add new links to the Bamboo interface.</td>
</tr>
<tr>
<td>web-section</td>
<td>1.x, 2.x</td>
<td>1.0</td>
<td><strong>Web Section Module</strong></td>
<td>Add a new section to the Bamboo interface.</td>
</tr>
<tr>
<td>web resource</td>
<td>1.x, 2.x</td>
<td>2.3</td>
<td><strong>Web Resources</strong></td>
<td>Permits web resources to be included with a plugin.</td>
</tr>
<tr>
<td>webRepositoryViewer</td>
<td>1.x, 2.x</td>
<td>2.5</td>
<td><strong>Web Repository Viewer Module</strong></td>
<td>Renders commit information to include web repository links and information.</td>
</tr>
<tr>
<td>xwork</td>
<td>1.x, 2.x</td>
<td>1.0</td>
<td><strong>XWork Plugin Module</strong></td>
<td>XWork actions and views bundled with the plugin. This enables building generic user interfaces.</td>
</tr>
</tbody>
</table>

**System Plugin Modules**

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capabilityTypeModule</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td><strong>Capability Type Module</strong></td>
<td>The Capability Type Module is used for registering custom capabilities.</td>
</tr>
<tr>
<td>component</td>
<td>1.x</td>
<td>2.6</td>
<td><strong>Component Module</strong></td>
<td>Enables you to add components to Bamboo’s internal component system (powered by Spring).</td>
</tr>
<tr>
<td>downloadable resource</td>
<td>1.x, 2.x</td>
<td>2.3</td>
<td><strong>Downloadable Plugin Resources</strong></td>
<td>Permits downloadable resources to be included with a plugin.</td>
</tr>
<tr>
<td>indexReader</td>
<td>1.x, 2.x</td>
<td>1.0</td>
<td><strong>Index Reader Module</strong></td>
<td>Reads out custom index information written by the postBuildIndexWriter from the index.</td>
</tr>
</tbody>
</table>
### Setting up a Bamboo Plugin Project

Refer to the [Getting started with Atlassian plugins](https://docs.atlassian.com/developer/guides/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](https://store.atlassian.com/) you can obtain the Bamboo Source code. Refer to the [Building Bamboo from source](https://docs.atlassian.com/developer/guides/bamboo/) 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

```shell
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](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](https://docs.atlassian.com/developer/guides/maven/) 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-repository-plugin.xml` - the built reports of builds grouped by time periods under the Reports tab.
- `system-repo-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
- Differences between Plugins1 and Plugins2
### Bamboo Plugin Module Types

The following types of plugin modules are supported by Bamboo:

#### Build Lifecycle Plugin Modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildAgentRequirementFilter</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td>Build Agent Requirement Filter Module</td>
<td>Customises the list of Agents a plan can be built on.</td>
</tr>
<tr>
<td>buildCompleteAction</td>
<td>1.x, 2.x</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>builder</td>
<td>1.x</td>
<td>1.0</td>
<td>Builder Plugin Module</td>
<td>Add new builders to Bamboo.</td>
</tr>
<tr>
<td>buildProcessor</td>
<td>1.x</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>1.x</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>command decorator</td>
<td>1.x</td>
<td>2.3</td>
<td>Command Decorator Module</td>
<td>Allows additional parameters to be added to the command line of a builder before a build occurs.</td>
</tr>
<tr>
<td>preBuildQueuedAction</td>
<td>1.x, 2.x</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.x</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>postBuildCompletedAction</td>
<td>2.x</td>
<td>2.6</td>
<td>Post-Build Completed Action Module</td>
<td>Add a custom asynchronous 'build context'-aware action after the build process has completed and the build results have been saved.</td>
</tr>
<tr>
<td>preChainAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Chain Action Module</td>
<td>Prepends a custom synchronous process before the chain starts executing</td>
</tr>
<tr>
<td>postChainAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Chain Action Module</td>
<td>Prepends a custom synchronous process after the chain finishes executing</td>
</tr>
<tr>
<td>preStageAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Stage Action Module</td>
<td>Prepends a custom synchronous process before the stage starts executing</td>
</tr>
<tr>
<td>postStageAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Stage Action Module</td>
<td>Prepends a custom synchronous process after the stage finishes executing</td>
</tr>
<tr>
<td>preJobAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Pre-Job Action Module</td>
<td>Prepends a custom synchronous process before the job starts executing</td>
</tr>
<tr>
<td>postJobAction</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post-Job Action Module</td>
<td>Prepends a custom synchronous process after the job finishes executing</td>
</tr>
<tr>
<td>repository</td>
<td>1.x</td>
<td>1.1</td>
<td>Source Repository Module</td>
<td>Add a custom source repository.</td>
</tr>
<tr>
<td>triggerReason</td>
<td>1.x</td>
<td>2.0</td>
<td>Trigger Reason Module</td>
<td>Add a new trigger reason.</td>
</tr>
</tbody>
</table>

#### Notification Plugin Modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notification-condition</td>
<td>1.x, 2.x</td>
<td>1.1 to 2.2</td>
<td>Notification Condition Module</td>
<td>Add new notification condition.</td>
</tr>
</tbody>
</table>
### Bamboo Event Listeners

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bambooEventListener</td>
<td>1.x, 2.x</td>
<td>2.2</td>
<td>Bamboo Event Listeners</td>
<td>Register an event listener against any Bamboo Events.</td>
</tr>
</tbody>
</table>

### User Interface Plugin Modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalBuildConfigurationPlugin</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Additional Build Configuration Module</td>
<td>Register additional configuration UI on the Miscellaneous tab of the Configuration of a Job or Plan</td>
</tr>
<tr>
<td>report</td>
<td>2.x</td>
<td>1.0</td>
<td>Report Module</td>
<td>Defines a report of build telemetry data.</td>
</tr>
<tr>
<td>web-item</td>
<td>1.x, 2.x</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.x, 2.x</td>
<td>1.0</td>
<td>Web Section Module</td>
<td>Add a new section to the Bamboo interface.</td>
</tr>
<tr>
<td>web resource</td>
<td>1.x, 2.x</td>
<td>2.3</td>
<td>Web Resources</td>
<td>Permits web resources to be included with a plugin.</td>
</tr>
<tr>
<td>webRepositoryViewer</td>
<td>1.x, 2.x</td>
<td>2.5</td>
<td>Web Repository Viewer Module</td>
<td>Renders commit information to include web repository links and information.</td>
</tr>
<tr>
<td>xwork</td>
<td>1.x, 2.x</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>
</tbody>
</table>

### System Plugin Modules

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Plugin Framework Version</th>
<th>Since Bamboo version...</th>
<th>Documentation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capabilityTypeModule</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Capability Type Module</td>
<td>The Capability Type Module is used for registering custom capabilities</td>
</tr>
<tr>
<td>component</td>
<td>1.x</td>
<td>2.6</td>
<td>Component Module</td>
<td>Enables you to add components to Bamboo’s internal component system (powered by Spring).</td>
</tr>
<tr>
<td>downloadable resource</td>
<td>1.x, 2.x</td>
<td>2.3</td>
<td>Downloadable Plugin Resources</td>
<td>Permits downloadable resources to be included with a plugin.</td>
</tr>
<tr>
<td>indexReader</td>
<td>1.x, 2.x</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>postBuildIndexWriter</td>
<td>1.x, 2.x</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>postChainIndexWriter</td>
<td>1.x, 2.x</td>
<td>2.7</td>
<td>Post Chain Index Writer Module</td>
<td>Writes custom chain data into the index for report generation.</td>
</tr>
</tbody>
</table>

### User Interface Plugin Modules

These modules allow developers to add custom User Interfaces to Bamboo

**Additional Build Configuration Module**

Additional Build Configuration Module allows developers to register additional configuration UI on the Miscellaneous tab of the Configuration of a Job or Plan.

**Report Module**
This defines a report module. A report module will appear under the Reports tab.

**Web Item Module**

The Web Item module allows you to define a link in the Bamboo system. (Usually in some form of menu).

**Web Panel Module**

Web Panel plugin modules allow plugins to define panels, or sections, on an HTML page. A panel is a set of HTML that will be inserted into a page.

**Web Repository Viewer Module**

A plugin module which defines a web repository viewer in Bamboo. This module is responsible for rendering commit information in Bamboo. For example, the Fisheye Web Repository Viewer will render the commit, providing links to the web repository.

**Web Section Module**

The Web Section 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.

**XWork Plugin Module**

Each XWork module is deployed as a plugin module of type xwork and contains one or more XWork package elements.

### Additional Build Configuration Module

<table>
<thead>
<tr>
<th>Available:</th>
<th>Bamboo 2.7 and later</th>
</tr>
</thead>
</table>

**On this page:**

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example
  - Plugin Descriptor
  - Freemarker Edit and View Templates
  - Display Conditions

**Purpose of this Module Type**

Additional Build Configuration Module allows developers to register additional configuration UI on the Miscellaneous tab of the Configuration of a Job or Plan.

**Configuration**

The root element for the Additional Build Configuration module is `additionalBuildConfigurationPlugin`. It allows the following attributes and child elements for configuration:

#### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.</td>
</tr>
</tbody>
</table>
name

The human-readable name of the plugin module. Only used in the plugin’s administrative user interface.

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
</tr>
<tr>
<td>resource</td>
<td>🟢</td>
<td>Using the &quot;edit&quot; and &quot;view&quot; attributes it is possible to provide a custom template for editing and viewing your custom plugin configuration</td>
</tr>
</tbody>
</table>

Interface

Additional Build Configuration modules must implement the MiscellaneousBuildConfigurationPlugin interface and extend BaseBuildConfigurationAwarePlugin.

Example

**Plugin Descriptor**

Here is an example atlassian-plugin.xml file containing a Post Chain Index Writer Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Additional Build Configuration Module module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <additionalBuildConfigurationPlugin key="concurrentBuild" name="Concurrent Build Configuration" class="com.atlassian.bamboo.build.configuration.ConcurrentBuildsPlanConfigurationPlugin">
    <description>Plugin to configure maximum number of concurrent builds for a plan</description>
    <resource type="freemarker" name="edit" location="build/concurrent/editConcurrentBuildsConfigForPlan.ftl"/>
    <resource type="freemarker" name="view" location="build/concurrent/viewConcurrentBuildsConfigForPlan.ftl"/>
  </additionalBuildConfigurationPlugin>
</atlassian-plugin>
```

**Freemarker Edit and View Templates**

Example "edit" template:

```freemarker
[@ui.bambooSection titleKey="build.concurrent.title"]
[@ww.checkbox labelKey='build.concurrent.overrideDefault' name='custom.concurrentBuilds.overrideNumberOfConcurrentBuilds' toggle='true' ]

[@ui.bambooSection dependsOn='custom.concurrentBuilds.overrideNumberOfConcurrentBuilds' showOn='true']
[@ww.textfield labelKey='build.concurrent.maxnumber' name='custom.concurrentBuilds.numberOfConcurrentBuilds' ]
[/@ui.bambooSection]
[/@ui.bambooSection]
```

Example "view" template:
Display Conditions

By overriding isApplicableTo it is possible to control the conditions on which the configuration UI is shown. One typical implementation is to check and only display on a plans type.

To display on all TopLevelPlan’s (Chains and Builds):

```java
public boolean isApplicableTo(@NotNull Plan plan) {
    return plan instanceof TopLevelPlan;
}
```

To display only on Jobs:

```java
public boolean isApplicableTo(@NotNull Plan plan) {
    return plan instanceof Job;
}
```

Report Module

Available: Bamboo 1.0 and later

On this page:

- Description
- Sample Module Descriptor Element

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.reportscollector.ReportCollector interface. This takes in a list of builds and generates a DataSet.
- A ReportLineChart object extending the com.atlassian.bamboo.reportscharts.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.reportscharts.BuildSummarySuccessRatioLineChart</chartClass>
</report>
```

Web Item Module
**Description**

The **Web Item** module 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 the following locations:

<table>
<thead>
<tr>
<th>Menu</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard Tabs (since 3.0)</td>
<td>system.dashboard/dashboard</td>
</tr>
<tr>
<td>Plan Sub Menu (tabs on the View Plan page)</td>
<td>chain.subMenu/chain</td>
</tr>
<tr>
<td>Plan Results Sub Menu (tabs on the View Plan Result page)</td>
<td>chainResults.subMenu/chainResults</td>
</tr>
<tr>
<td>Job Sub Menu (tabs on the View Job page)</td>
<td>build.subMenu/build</td>
</tr>
<tr>
<td>Job Results Sub menu (tabs on the View Job Result page)</td>
<td>results.subMenu/results</td>
</tr>
<tr>
<td>Administration Menu</td>
<td>system.admin/plugins</td>
</tr>
</tbody>
</table>

When putting items in the Administration Menu, you can also add your own section to the menu (rather than using the suggested plugins section above). For more information see Web Section Module.

**Bamboo-specific notes about Web Items Module Parameters**

Please take note of the following Bamboo-specific information that is relevant to the Module components described in the 'Configuration' section below.

**Attributes:**

- **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

**Elements:**

- **condition** — by implementing the com.atlassian.plugin.web.Condition class you can add rules to determine whether the link will be displayed or not.
- **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

Both the **link** and the **key** id can make use of parameters passed to the page. For example:

```xml
<link>/build/viewBuildFiles.action?buildKey=${buildKey}</link>
```

where `${buildKey}` is the parameter name.

**Context Aware Navigation**

In Bamboo 3.0 we have added the Plan Navigator. When moving between Jobs and the Plan (as well as Job Results and Plan Result), the navigator remembers your context i.e. which tab (Web Item) you are on and will attempt to keep you on the same tab. If you are adding a new tab to a page which has the Plan Navigator, you may also wish to manually define the corresponding Job/Plan/Result URL. You can find more information on how to do this on the XWork Plugin Module page.
Purpose of this Module Type

Web Item plugin modules allow plugins to define new links in application menus.

Configuration

The root element for the Web Item plugin module is `<web-item>`. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td></td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>.</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Used only in the plugin’s administrative user interface.</td>
<td></td>
</tr>
<tr>
<td>section</td>
<td></td>
<td>Location into which this web item should be placed. For non-sectioned locations, this is just the location key. For sectioned locations it is the location key, followed by a slash (&quot;/&quot;). The name of the web section in which it should appear.</td>
<td>N/A</td>
</tr>
<tr>
<td>system</td>
<td></td>
<td>Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-OSGi plugins.</td>
<td>false</td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td>Determines the order in which web items appear. Items are displayed top to bottom or left to right in order of ascending weight. The 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom. The weights for most applications' system sections start from 100, and the weights for the links generally start from 10. The weight is incremented by 10 for each in sequence so that there is ample space to insert your own sections and links.</td>
<td>1000</td>
</tr>
</tbody>
</table>

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td></td>
<td>Defines a condition that must be satisfied for the web item to be displayed. If you want to 'invert' a condition, add an attribute 'invert=&quot;true&quot;' to it. The web item will then be displayed if the condition returns false (not true).</td>
<td>N/A</td>
</tr>
<tr>
<td>conditions</td>
<td></td>
<td>Defines the logical operator type to evaluate its condition elements. By default 'AND' will be used.</td>
<td>AND</td>
</tr>
<tr>
<td>context-provider</td>
<td></td>
<td>Allows dynamic addition to the velocity context available for various web item elements (in XML descriptors only). Currently only one context-provider can be specified per web item and section.</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. I.e. the description of the web item.</td>
<td></td>
</tr>
<tr>
<td>icon</td>
<td></td>
<td>Defines an icon to display with or as the link. Note: In some cases the icon element is required. Try adding it if your web section is not displaying properly.</td>
<td>N/A</td>
</tr>
<tr>
<td>label</td>
<td></td>
<td>Is the i18n key that will be used to look up the textual representation of the link.</td>
<td>N/A</td>
</tr>
<tr>
<td>link</td>
<td></td>
<td>Defines where the web item should link to. The contents of the link element will be rendered using Velocity, allowing you to put dynamic content in links. For more complex examples of links, see below.</td>
<td>N/A</td>
</tr>
<tr>
<td>param</td>
<td></td>
<td>Parameters for the plugin module. Use the 'key' attribute to declare the parameter key, then specify the value in either the 'value' attribute or the element body. This element may be repeated. An example is the configuration link described in Adding a Configuration UI for your Plugin. This is handy if you want to use additional custom values from the UI.</td>
<td>N/A</td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>A resource for this plugin module. This element may be repeated. A 'resource' is a non-Java file that a plugin may need in order to operate. Refer to Adding Plugin and Module Resources for details on defining a resource.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
tooltip

| N/A |

Is the i18n key that will be used to look up the textual mouse-over text of the link.

Label Elements

Label elements may contain optional parameters, as shown below:

- The parameters allow you to insert values into the label using Java's `MessageFormat` syntax.
- Parameter names must start with `param` and will be mapped in alphabetical order to the substitutions in the format string. I.e. `param0` is {0}, `param1` is {1}, `param2` is {2}, etc.
- Parameter values are rendered using Velocity, allowing you to include dynamic content.

Tooltip Elements

Tooltip elements have the same attributes and parameters as the label elements. See above.

Link Elements

Link elements may contain additional information:

- The `linkId` is optional, and provides an XML id for the link being generated.
- The `absolute` is optional and defaults to false unless the link starts with `http://` or `https://`

The body of the link element is its URL. The URL is rendered with Velocity, so you can include dynamic information in the link. For example, in Confluence, the following link would include the page ID:

Icon Elements

Icon elements have a `height` and a `width` attribute. The location of the icon is specified within a link element:

Param Elements

Param elements represent a map of key/value pairs, where each entry corresponds to the param elements attribute: `name` and `value` respectively.

The value can be retrieved from within the Velocity view with the following code, where $item is a `WebItemModuleDescriptor`:

If the `value` attribute is not specified, the value will be set to the body of the element. I.e. the following two param elements are equivalent:

Context-provider Element

The context-provider element adds to the Velocity context available to the web section and web item modules. You can add what you need to the context, to build more flexible section and item elements. Currently only one context-provider can be specified per module. Additional context-providers are ignored.

The context-provider element must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement `com.atlassian.plugin.web.ContextProvider`, and
- will be auto-wired by Spring before any additions to the Velocity context.

For example, the following context-provider will add `historyWindowHeight` and `filtersWindowHeight` to the context.

In the following example, `HeightContextProvider` extends `AbstractJiraContextProvider`, which is only available in JIRA and happens to implement `ContextProvider`. The `AbstractJiraContextProvider` conveniently extracts the `User` and `JiraHelper` from the context map, which you would otherwise have to do manually.

The newly added context entries `historyWindowHeight` and `filtersWindowHeight` can be used in the XML module descriptors just like normal velocity context variables, by prefixing them with the dollar symbol ($):
Condition and Conditions Elements

Conditions can be added to the web section, web item and web panel modules, to display them only when all the given conditions are true.

Condition elements must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement `com.atlassian.plugin.web.Condition`, and
- will be auto-wired by Spring before any condition checks are performed.

Condition elements can take optional parameters. These parameters will be passed in to the condition's `init()` method as a map of string key/value pairs after autowiring, but before any condition checks are performed. For example:

```
To invert a condition, add the attribute "invert="true"" to the condition element. This is useful where you want to show the section if a certain condition is not satisfied.

Conditions elements are composed of a collection of condition/conditions elements and a type attribute. The type attribute defines what logical operator is used to evaluate its collection of condition elements. The type can be one of AND or OR.

For example: The following condition is true if the current user is a system administrator OR a project administrator:
```

Example

Here is an example `atlassian-plugin.xml` file containing a single web item:

```
Web Panel Module

Web Panel plugin modules allow plugins to define panels, or sections, on an HTML page. A panel is a set of HTML that will be inserted into a page.

On This page:

- Purpose of this Module Type
- Configuration
- Attributes
- Elements
- Resource Element
- Context-provider Element
- Condition and Conditions Elements
- Web Panel Examples
- Bamboo specific notes for Web Panels
  - Locations
  - Default Context Objects
  - Freemarker resource type
  - Adding Web Panel locations to your own plugins
    - Example

Purpose of this Module Type

Web Panel plugin modules allow plugins to define panels, or sections, on an HTML page. A panel is a set of HTML that will be inserted into a page.

Configuration

The root element for the Web Panel plugin module is `web-panel`. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module and which is responsible for providing the web panel's HTML. In most cases you will not need to provide a custom class to generate the content, as you can simply point to a static HTML file or a (Velocity) template. See the plugin framework guide to creating plugin module instances. If you omit this attribute, you MUST provide a resource element and vice versa, to ensure there is always exactly one source for the web panel's content.</td>
<td></td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
</tbody>
</table>
The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key `fred` in a plugin with key `com.example.modules` will have a complete key of `com.example.modules:fred`.

The human-readable name of the plugin module. Used only in the plugin’s administrative user interface.

Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-OSGi plugins.

Determines the order in which web panels appear. Web panels are displayed top to bottom or left to right in order of ascending weight. The 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom. The weights for most applications' system sections start from 100, and the weights for the links generally start from 10. The weight is incremented by 10 for each in sequence so that there is ample space to insert your own panels.

The location in the host application where the web panel must be rendered. Note that every host application declares its own set of web panel plugin points. Currently a web panel can only be associated with a single location.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td></td>
<td>Defines a condition that must be satisfied for the web panel to be displayed. If you want to 'invert' a condition, add an attribute 'invert=&quot;true&quot;' to it. The web item will then be displayed if the condition returns false (not true).</td>
<td>N/A</td>
</tr>
<tr>
<td>conditions</td>
<td></td>
<td>Defines the logical operator type to evaluate its condition elements. By default 'AND' will be used.</td>
<td>AND</td>
</tr>
<tr>
<td>context-provider</td>
<td></td>
<td>Allows dynamic addition to the Velocity context available for various web panel elements (in XML descriptors only). Currently only one context-provider can be specified per web panel.</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. i.e. the description of the web panel.</td>
<td></td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>A resource element is used to provide a web panel with content. It can be used in a way similar to normal resources, using the resource's location attribute to point to a static HTML file or (Velocity) template file that is provided by the plugin's JAR file. To differentiate between static HTML and Velocity templates that need to be rendered, always specify the <code>type</code> attribute. See the examples further down on this page. It is also possible to embed the contents (both static HTML or velocity) directly in the <code>atlassian-plugin.xml</code> file by encoding it in the resource element's body and then omitting the <code>location</code> attribute. Note that if you omit the resource element you MUST provide the module descriptor's <code>class</code> attribute, and vice versa, to ensure there is always exactly one source for the web panel's content.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Resource Element**

Unless the module descriptor's `class` attribute is specified, a web panel will contain a single resource child element that contains the contents of the web panel. This can be plain HTML, or a (Velocity) template to provide dynamic content.

A web panel's resource element can either contain its contents embedded in the resource element itself, as part of the `atlassian-plugin.xml` file, or it can link to a file on the classpath when the `location` attribute is used.

A resource element's `type` attribute identifies the format of the panel's content (currently "static" and "velocity" are provided by Atlassian Plugin Framework 2.5.0 and atlassian-template-renderer 2.5.0 respectively) which allows the plugin framework to use the appropriate `com.atlassian.plugin.web.renderer.WebPanelRenderer`.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>Used to indicate that the web panel's contents must not be processed, but included in the page as is.</td>
</tr>
<tr>
<td>velocity</td>
<td>Used to indicate that the web panel contains Velocity markup that needs to be parsed.</td>
</tr>
</tbody>
</table>

The template rendering system is extensible. You can add custom renderers by creating plugins. For more information on this, check out the Web Panel Renderer Plugin Module.

**Context-provider Element**

Available: Atlassian Plugins 2.5, Confluence 2.5, Bamboo 3.0, JIRA 4.2 and later
The context-provider element adds to the Velocity context available to the web section and web item modules. You can add what you need to the context, to build more flexible section and item elements. Currently only one context-provider can be specified per module. Additional context-providers are ignored.

The context-provider element must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement com.atlassian.plugin.web.ContextProvider, and
- will be auto-wired by Spring before any additions to the Velocity context.

For example, the following context-provider will add historyWindowHeight and filtersWindowHeight to the context.

In the following example, HeightContextProvider extends AbstractJiraContextProvider, which is only available in JIRA and happens to implement ContextProvider. The AbstractJiraContextProvider conveniently extracts the User and JiraHelper from the context map, which you would otherwise have to do manually.

The newly added context entries historyWindowHeight and filtersWindowHeight can be used in the XML module descriptors just like normal velocity context variables, by prefixing them with the dollar symbol ($):

Condition and Conditions Elements

Conditions can be added to the web section, web item and web panel modules, to display them only when all the given conditions are true.

Condition elements must contain a class attribute with the fully-qualified name of a Java class. The referenced class:

- must implement com.atlassian.plugin.web.Condition, and
- will be auto-wired by Spring before any condition checks are performed.

Condition elements can take optional parameters. These parameters will be passed in to the condition's init() method as a map of string key/value pairs after autowiring, but before any condition checks are performed. For example:

To invert a condition, add the attribute 'invert="true"' to the condition element. This is useful where you want to show the section if a certain condition is not satisfied.

Conditions elements are composed of a collection of condition/conditions elements and a type attribute. The type attribute defines what logical operator is used to evaluate its collection of condition elements. The type can be one of AND or OR.

For example: The following condition is true if the current user is a system administrator OR a project administrator:

Web Panel Examples

The values of the location attributes in the examples below are not real. They are just illustrative of the kind of location that Confluence, Bamboo and FishEye may make available in future releases.

A web panel that contains static, embedded HTML:

A web panel that contains an embedded Velocity template:

A web panel containing a Velocity template that is on the classpath (part of the plugin's JAR file):

As mentioned previously, it is also possible to provide your own custom class that is responsible for producing the panel's HTML, by using the descriptor's class attribute (which makes the resource element redundant):

Note that com.example.FooWebPanel MUST implement WebPanel.

Bamboo specific notes for Web Panels

Locations

Bamboo supports the following web panel locations:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan.navigator</td>
<td>Renders below the plan navigator on all Plan, Job, Plan result and Job result screens</td>
<td>3.0</td>
</tr>
</tbody>
</table>
job.configuration.artifact.definitions | Renders below the Artifact definitions table in the Job configuration | 3.0
job.configuration.artifact.subscriptions | Renders below the Shared Artifacts table in the Job configuration | 3.0
plan.result.artifacts | Renders below the default content of the Plan result artifacts tab | 3.0
job.result.artifacts | Renders below the default content of the Job result artifacts tab | 3.0

**Default Context Objects**

The following objects are available by default without having to specify your own context-provider.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan</td>
<td>Refers to the current Plan</td>
<td>Any page that shows a Plan, Job, Plan Result or Job Result</td>
</tr>
<tr>
<td>resultSummary</td>
<td>Refers to the current ResultsSummary</td>
<td>Any page that shows a Plan Result or Job Result</td>
</tr>
</tbody>
</table>

**Freemarker resource type**

In addition to the standard velocity and static Web Panel resource types Bamboo introduces a freemarker type used to render Bamboos preferred templating engine Freemarker.

```xml
<web-panel key="feedbackPanel" location="plan.navigator">
  <resource name="view" type="freemarker" location="/fragments/feedback/feedbackWebPanel.ftl" />
</web-panel>
```

**Adding Web Panel locations to your own plugins**

Bamboo provides easy support for adding your own Web Panels via the ui.renderWebPanels freemarker macro.

Example

```freemarker
[@ui.renderWebPanels 'my.webpanel.location' />
```

Once specified in any Freemarker template any plugin can specify my.webpanel.location in their web-panel location declaration and be expected to render along side your template.

**Web Repository Viewer Module**

**Available:** Bamboo 2.5 and later

On this page:

- Description
- Interface
- Sample Module Descriptor Element

**Description**

A plugin module which defines a web repository viewer in Bamboo. This module is responsible for rendering commit information in Bamboo. For example, the Fisheye Web Repository Viewer will render the commit, providing links to the web repository.

**Interface**

Web Repository Viewer modules must implement the com.atlassian.bamboo.webrepository.WebRepositoryViewer interface. An abstract class is available that implements a lot of the default functionality for you, so you can extend com.atlassian.bamboo.webrepository.AbstractWebRepositoryViewer.

If you would like to render links to commits in emails, you will need to also implement the com.atlassian.bamboo.webrepository.CommitUrlProvider interface.

**Sample Module Descriptor Element**
Web Section Module

On this page:

- Description
- Bamboo-specific Notes
- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
  - Label Elements
  - Tooltip Elements
  - Param Elements
  - Context-provider Element
  - Condition and Conditions elements
- Example

Description

The Web Section 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.

Bamboo-specific Notes

The Administration Menu is currently the only place in Bamboo that allows the addition of web sections. The location attribute of these web sections will be `system.admin`.

<table>
<thead>
<tr>
<th>Menu</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration Menu</td>
<td>system.admin</td>
</tr>
</tbody>
</table>

However, Bamboo provides a number of other in-built web sections to add other Web Item Modules to. You can find more information on these on the Web Item Module page.

Purpose of this Module Type

Web Section plugin modules allow plugins to define new sections in application menus. Each section can contain one or more links. To insert the links themselves, see the Web Item Plugin Module.

Configuration

The root element for the Web Section plugin module is `web-section`. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
</tbody>
</table>
The identifier of the plugin module. This key must be unique within the plugin where it is defined.

Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.

The human-readable name of the plugin module. Only used in the plugin's administrative user interface.

Location into which this web item should be placed. For non-sectioned locations, this is just the location key. For sectioned locations it is the location key, followed by a slash ('/'), and the name of the web section in which it should appear.

Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-O SGi plugins.

Determines the order in which web items appear. Items are displayed top to bottom or left to right in order of ascending weight. The 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom. The weights for most applications' system sections start from 100, and the weights for their links generally start from 10. The weight is incremented by 10 for each in sequence so that there is ample space to insert your own sections and links.

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td></td>
<td>Defines a condition that must be satisfied for the web item to be displayed. If you want to 'invert' a condition, add an attribute 'invert=&quot;true&quot;' to it. The web item will then be displayed if the condition returns false (not true).</td>
<td>N/A</td>
</tr>
<tr>
<td>conditions</td>
<td></td>
<td>Defines the logical operator type used to evaluate the condition elements. By default 'AND' will be used.</td>
<td>AND</td>
</tr>
<tr>
<td>context-provider</td>
<td></td>
<td>Allows dynamic addition to the Velocity context available for various web item elements (in XML descriptors only). Currently only one context-provider can be specified per web item and section.</td>
<td>N/A</td>
</tr>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>✓</td>
<td>Is the i18n key that will be used to look up the textual representation of the link.</td>
<td>N/A</td>
</tr>
<tr>
<td>param</td>
<td></td>
<td>Parameters for the plugin module. Use the 'key' attribute to declare the parameter key, then specify the value in either the 'value' attribute or the element body. This element may be repeated. An example is the configuration link described in Adding a Configuration UI for your Plugin. Defines a key/value pair available from the web item. This is handy if you want to use additional custom values from the UI.</td>
<td>N/A</td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>A resource for this plugin module. This element may be repeated. A 'resource' is a non-Java file that a plugin may need in order to operate. Refer to Adding Plugin and Module Resources for details on defining a resource.</td>
<td>N/A</td>
</tr>
<tr>
<td>tooltip</td>
<td></td>
<td>Is the i18n key that will be used to look up the textual mouse-over text of the link.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Label Elements

Label elements may contain optional parameters, as shown below:

- The parameters allow you to insert values into the label using Java's MessageFormat syntax.
- Parameter names must start with param and will be mapped in alphabetical order to the substitutions in the format string. I.e. param0 is [0], param1 is [1], param2 is [2], etc.
- Parameter values are rendered using Velocity, allowing you to include dynamic content.

Tooltip Elements

Tooltip elements have the same attributes and parameters as the label elements. See above.

Param Elements

Param elements represent a map of key/value pairs, where each entry corresponds to the param elements attribute: name and value respectively.

The value can be retrieved from within the Velocity view with the following code, where $item is a WebItemModuleDescriptor:
If the `value` attribute is not specified, the value will be set to the body of the element. I.e. the following two param elements are equivalent:

Context-provider Element

The context-provider element adds to the Velocity context available to the web section and web item modules. You can add what you need to the context, to build more flexible section and item elements. Currently only one context-provider can be specified per module. Additional context-providers are ignored.

The `context-provider` element must contain a class attribute with the fully-qualified name of a Java class. The referenced class:
- must implement `com.atlassian.plugin.web.ContextProvider`, and
- will be auto-wired by Spring before any additions to the Velocity context.

For example, the following context-provider will add `historyWindowHeight` and `filtersWindowHeight` to the context.

In the following example, `HeightContextProvider` extends `AbstractJiraContextProvider`, which is only available in JIRA and happens to implement `ContextProvider`. The `AbstractJiraContextProvider` conveniently extracts the `User` and `JiraHelper` from the context map, which you would otherwise have to do manually.

The above `HeightContextProvider` can be used by nesting the following element in a web item module.

The newly added context entries `historyWindowHeight` and `filtersWindowHeight` can be used in the XML module descriptors just like normal velocity context variables, by prefixing them with the dollar symbol ($):

Condition and Conditions elements

Conditions can be added to the web section, web item and web panel modules, to display them only when all the given conditions are true.

Condition elements must contain a class attribute with the fully-qualified name of a Java class. The referenced class:
- must implement `com.atlassian.plugin.web.Condition`, and
- will be auto-wired by Spring before any condition checks are performed.

Condition elements can take optional parameters. These parameters will be passed in to the condition's `init()` method as a map of string key/value pairs after autowiring, but before any condition checks are performed. For example:

To invert a condition, add the attribute 'invert="true"' to the condition element. This is useful where you want to show the section if a certain condition is not satisfied.

Conditions elements are composed of a collection of condition/conditions elements and a type attribute. The type attribute defines what logical operator is used to evaluate its collection of condition elements. The type can be one of `AND` or `OR`.

For example: The following condition is true if the current user is a system administrator OR a project administrator:

Example

Here is an example `atlassian-plugin.xml` file containing a single web section, using a condition that will be available in JIRA:

XWork Plugin Module

On this page:
- Description
- Sample Module Descriptor Element
- Context Aware Navigation
- Using Bamboo's Decorators

Description
Each XWork module is deployed as a plugin module of type **xwork** and contains one or more XWork **package** elements.

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.

Below is an example **atlassian-plugin.xml** file containing a single XWork module.

### 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>
```

### Context Aware Navigation

**Since 3.0**

⚠️ Only applicable to people adding **Web Items** to Plan/Job/Result tabs

Some pages in Bamboo have a Plan Navigator. When moving between Jobs and the Plan (as well as Job Results and Plan Result), the navigator remembers our context i.e. which tab you are on and will attempt to keep you on the same tab. There is some default behaviour available to determine which page to navigate to (and hence which tab), however, if you would like your plugin to dictate exactly where to go, you can manually define the corresponding page to navigate to via your xwork action definition.

For each xwork action you can define the `jobEquiv` and `chainEquiv` parameters. The `chainEquiv` will be used when navigating to the Plan (or Plan Result), the `jobEquiv` when navigating to the Job (or Job Result). Note that even though you are already on a Job level page, you can set the `jobEquiv` to be something different for other Jobs in the Plan.

You can use the markers `${planKey}` and `${buildNumber}` as place holders in the equivalent URL.

**Examples:**

```xml
<action name="viewChainAuditLog"
   class="com.atlassian.bamboo.ww2.actions.chains.admin.ViewChainAuditLog">
   <param name="jobEquiv">/build/admin/edit/editBuildDetails.action?buildKey=${planKey}</param>
   <param name="chainEquiv">chain/admin/config/viewChainAuditLog.action?buildKey=${planKey}</param>
   <result name="success" type="freemarker">/chain/edit/viewChainChangeHistory.ftl</result>
</action>
```

```xml
<action name="viewBuildResultsSuccessfulTests"
   class="com.atlassian.bamboo.build.ViewBuildResultsSuccessfulTests">
   <param name="chainEquiv">/browse/${planKey}-${buildNumber}/test</param>
   <param name="jobEquiv">/browse/${planKey}-${buildNumber}/test</param>
   <result name="success" type="freemarker">/build/result/viewBuildResultsSuccessfulTests.ftl</result>
   <result name="error" type="freemarker">/error.ftl</result>
</action>
```

If Bamboo can not figure out where to take the user, it will default to the first tab on the page.

### Using Bamboo's Decorators

**Since 3.0**

---

657
Bamboo will decorate any HTML returned by your action. In most cases this will just be the Header and Footer, however there are a few decorators which do much more than this. There are two ways to control the decoration occurring on your page.

1. **URL Mapping** - Each decorator is mapped to a specific URL pattern, if your action URL matches that pattern it will use that decorator.
2. **Manual Specification** - If your action URL does not match the required pattern, you will need to manually define which decorator you want Bamboo to use.

```
<head>
  <meta name="decorator" content="<decorator-name>"/>
</head>
```

### Available Decorators

<table>
<thead>
<tr>
<th>Decorator Name</th>
<th>Default URL Mappings</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan</td>
<td><code>&lt;pattern&gt;/build/*.action&lt;/pattern&gt;</code></td>
<td>For Plans and Job, provides Breadcrumbs, Plan Navigator and Tab menu</td>
</tr>
<tr>
<td>result</td>
<td><code>&lt;pattern&gt;/chain/result/*.action&lt;/pattern&gt;</code></td>
<td>For Plan Results and Job Results, provides Breadcrumbs, Status Ribbon, Plan Navigator and Tab menu</td>
</tr>
<tr>
<td>atl.admin</td>
<td><code>&lt;pattern&gt;/admin/*.action&lt;/pattern&gt;</code></td>
<td>For Administration Pages, provides Admin menu</td>
</tr>
<tr>
<td>atl.general</td>
<td>Everything</td>
<td>The default decorator (Header + Footer only)</td>
</tr>
<tr>
<td>none</td>
<td>Will not provide any decoration</td>
<td></td>
</tr>
</tbody>
</table>

### Tabs and Decorators

If the decorator is providing a row of tabs to your page, you will need to manually define on their page which tab is selected as follows:

```
<head>
  <meta name="tab" content="<web-item-name>"/>
</head>
```

People using Web Items to add to the Plan/Job/Result tabs should use the plan or result decorator

### Build Lifecycle Plugin Modules

These modules allow developers to extend Bamboos build processes with custom actions that change the outcome of the build, collect data, etc.

**Build Agent Requirement Filter Module**

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.

**Build Complete Action Module**

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.

**Builder Plugin Module**

A plugin module which defines a builder in Bamboo, such as Maven, Maven2, or Ant.

**Build Processor Module**

The BuildProcessor module allows you to define a custom process that runs during the build.

**Build Processor Server Module**

The BuildProcessorServer module allows you to define a custom process that runs during the build.
**Build Trigger Condition Module**

The `buildTriggerCondition` module provides the ability to gate whether a build should be triggered or not. It uses a preference system so multiple trigger condition plugins can be used to determine the desired result. A build which fails the trigger condition will not be executed, there will be no trace of the existence of this build.

**Command Decorator Module**

The `Command Decorator` module allows you to add additional parameters to the command line of a builder before a build occurs.

**Post-Build Completed Action Module**

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 been completed before the `PostBuildCompletedAction` is fired. Hence, build complete actions cannot impact the state of the build.

**Post-Chain Action Module**

Post-Chain Action modules allow plugins to define custom functionality that runs after the Plan finishes executing. This module will be run regardless of BuildState or LifeCycle state of the `ChainResultsSummary` (For example, if the build completed successfully or failed). The module provides access to the `Chain` object, the `ChainResultsSummary` and the `ChainExecution`.

**Post-Job Action Module**

Post-Job Action modules allow plugins to define custom functionality that runs after all the results for Jobs belonging to the stage have been returned to the server and saved to the database.

**Post-Stage Action Module**

Post-Stage Action modules allow plugins to define custom functionality that runs after the jobs for the given stage have finished executing. The module provides access to the `ChainResultsSummary`, `ChainStageResult` (where by individual `BuildResultsSummary` objects for the each `Job` can be accessed) and the `StageExecution`.

**Pre-Build Action Module**

The `PreBuildAction` module allows you to define a custom process that runs before your build begins.

**Pre-Build Queued Action**

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.

**Pre-Chain Action Module**

Pre-Chain Action modules allow plugins to define custom functionality that runs before the Plan executes. The module provides access to the `Chain` object and the `ChainExecution`.

**Pre-Job Action Module**

Pre-Job Action modules allow plugins to define custom functionality that runs on the server before the Job is queued. This module shares similarities but differs from the `Pre-Build Queued Action` because it gives access to both the `StageExecution` and `BuildContext` rather than just the `BuildContext`.

**Pre-Stage Action Module**

Pre-Stage Action modules allow plugins to define custom functionality that runs before the jobs for the given stage are queued for building. The module provides access to the `StageExecution` object.

**Source Repository Module**

A plugin module which defines a repository in Bamboo, such as CVS, Subversion, or Perforce.

**Task Plugin Module**

A task plugin represents a unit of work during a Build. Each Job can now have multiple Tasks which run in order on a single Agent. If a Task fails the following tasks will not execute, with the exception of Final Tasks which will always execute.

**Trigger Reason Module**

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
Build Agent Requirement Filter Module

**Available:** Bamboo 2.2 and later

On this page:
- Description
- Interface
- Sample Module Descriptor Element

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="labManagerAgentFilter" name="LabManager Agent filter"
class="com.atlassian.bamboo.plugins.labmanager.LabManagerAgentFilter">
  <description>Ensures lab manager builds don’t run on normal agents</description>
</buildAgentRequirementFilter>
```

Build Complete Action Module

**Available:** Bamboo 1.0 and later

We advise against using this plugin point as it has the tendency to suffer from thread safety issues. Please use the Post Build Completed Action or the Build Processor Server instead.

On this page:
- Description
- Interface
- Sample Module Descriptor Element

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">
  <description>An automatic labelling plugin.</description>
</buildCompleteAction>
```
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

```xml
<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>
```

Extensions
Your builder plugin also has the option to implement the below interfaces:

- **PostConfigurableBuilder** - Perform any post actions that might need to occur after you save the builder configuration. This will get run on all builders regardless of which one is actually selected. Example usage is to clean up unused data if you are not the selected builder or add extra info if you are the selected builder.

Build Processor Module

Available: Bamboo 1.0 and later

On this page:
- Description
- Interface
- Sample Module Descriptor Element

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 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
Build Processor Server Module

Available: Bamboo 2.0 and later

On this page:

- Description
- Interface
- Sample Module Descriptor Element

Description

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

```xml
<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>
```

Build Trigger Condition Module

Available: Bamboo 3.0 and later

On this page:

- Description
- Interface
- Sample Module Descriptor Element

Description

The **buildTriggerCondition** module provides the ability to gate whether a build should be triggered or not. It uses a preference system so multiple trigger condition plugins can be used to determine the desired result. A build which fails the trigger condition will not be executed, there will be no trace of the existence of this build.
The following ExecutionPreferences are available (in order of priority - highest first):

- MUST_STOP
- MUST_RUN
- SHOULD_STOP
- SHOULD_RUN
- NONE

SHOULD_RUN is the default used by Bamboo. Please carefully consider which preference you use, to allow for better integration with future plugins. For example you could have:

- Plugin A: Only will build if another build is green, will return SHOULD_STOP as its preference (if the other build is red)
- Plugin B: Ensures that there is always a build once a day so can over-ride Plugin A by return MUST_RUN
- Plugin C: Checks for resource availability and therefore can return MUST_STOP when a resource is not available, and override any other plugins which might want the plan to run.

Interface

Build Trigger Condition modules must implement the com.atlassian.bamboo.build.BuildTriggerCondition interface.

Sample Module Descriptor Element

```xml
<buildTriggerCondition key="plansGreenCondition" name="Plans Green Condition"
    class="com.atlassian.bamboo.buildtrigger.PlansGreenCondition">
    <description>Checks that a given list of plans is green before triggering a build.</description>
    <resource type="freemarker" name="edit" location="editPlansGreenCondition.ftl"/>
    <resource type="freemarker" name="view" location="viewPlansGreenCondition.ftl"/>
</buildTriggerCondition>
```

Command Decorator Module

| Available     | Bamboo 2.3 and later |

On this page:

- Description
- Interface
- Sample Module Descriptor Elements

Description

The Command Decorator module allows you 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>
```

Post-Build Completed Action Module

| Available     | Bamboo 2.6 and later |
### 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 been completed before the **PostBuildCompletedAction** is fired. Hence, build complete actions cannot impact the state of the build.

### Interface

Post Build Completed Action modules must implement the [CustomPostBuildCompletedAction](#) interface.

### Sample Module Descriptor Element

```xml
<postBuildCompletedAction 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>
</postBuildCompletedAction>
```

### Post-Chain Action Module

Available: Bamboo 2.7 and later

#### Purpose of this Module Type

Post-Chain Action modules allow plugins to define custom functionality that runs after the Plan finishes executing. This module will be run regardless of BuildState or LifeCycle state of the [ChainResultsSummary](#) (For example, if the build completed successfully or failed). The module provides access to the [Chain](#) object, the [ChainResultsSummary](#) and the [ChainExecution](#).

#### Configuration

The root element for the Pre-Chain Action module is **postChainAction**. It allows the following attributes and child elements for configuration:

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✔️</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>✔️</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
name

The human-readable name of the plugin module.
Only used in the plugin's administrative user interface.

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
</tr>
</tbody>
</table>

Interface

Post-Chain Action modules must implement the PostChainAction interface.

Example

Here is an example atlassian-plugin.xml file containing a Post-Chain Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Post-Chain Action module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>

  <postChainAction key="myPostChainAction" class="com.atlassian.example.bamboo.actions.MyPostChainAction">
    <description>Tears down and saves data for MyBambooPlugin if the Chain has the plugin enabled</description>
  </postChainAction>
</atlassian-plugin>
```

Post-Job Action Module

Available: Bamboo 2.7 and later

On this page:

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example

Purpose of this Module Type

Post-Job Action modules allow plugins to define custom functionality that runs after all the results for Jobs belonging to the stage have been returned to the server and saved to the database.

Configuration

The root element for the Post-Job Action module is postJobAction. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
</table>
class

The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.

key

The identifier of the plugin module. This key must be unique within the plugin where it is defined.

Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.

name

The human-readable name of the plugin module.

Only used in the plugin's administrative user interface.

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
<td></td>
</tr>
</tbody>
</table>

Interface

Post-Job Action modules must implement the PostJobAction interface.

Example

Here is an example atlassian-plugin.xml file containing a Post-Job Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Post-Job Action module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <postJobAction key="myPostJobAction" class="com.atlassian.example.bamboo.actions.MyPostJobAction">
    <description>Tears down the Job once the result has been returned to the server and saved to the database</description>
  </postJobAction>
</atlassian-plugin>
```

Post-Stage Action Module

Available: Bamboo 2.7 and later

On this page:

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example

Purpose of this Module Type

Post-Stage Action modules allow plugins to define custom functionality that runs after the jobs for the given stage have finished executing. The module provides access to the ChainResultsSummary, ChainStageResult (where by individual BuildResultsSummary objects for the each Job can be accessed) and the StageExecution.
Configuration

The root element for the Post-Stage Action module is `postStageAction`. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>.</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin's administrative user interface.</td>
<td></td>
</tr>
</tbody>
</table>

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
<td></td>
</tr>
</tbody>
</table>

Interface

Post-Stage Action modules must implement the `PostStageAction` interface.

Example

Here is an example `atlassian-plugin.xml` file containing a Post-Stage Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Post-Stage Action module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <postStageAction key="myPostStageAction"
                   class="com.atlassian.example.bamboo.actions.MyPostStageAction">
    <description>Tears down the stage after execution has completed</description>
  </postStageAction>
</atlassian-plugin>
```

Pre-Build Action Module

**Available:** Bamboo 1.1 and later

On this page:

- Description
- Interface
- Sample Module Descriptor Element
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>
```

Pre-Build Queued Action

**Available:** Bamboo 2.1 and later

On this page:
- Description
- Interface
- Sample Module Descriptor Element

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">
  <description>Starts a virtual machine</description>
</preBuildQueuedAction>
```

Pre-Chain Action Module

**Available:** Bamboo 2.7 and later

On this page:
- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example
**Purpose of this Module Type**

Pre-Chain Action modules allow plugins to define custom functionality that runs before the Plan executes. The module provides access to the **Chain** object and the **ChainExecution**.

**Configuration**

The root element for the Pre-Chain Action module is **preChainAction**. It allows the following attributes and child elements for configuration:

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>depends on the module type. For example, Confluence theme, layout and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>colour-scheme modules can use classes already provided in Confluence. So</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>you can write a theme-plugin without any Java code. But for macro and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>listener modules you need to write your own implementing class and include</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>it in your plugin. See the plugin framework guide to creating plugin module</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>instances.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td></td>
<td>The identifier of the plugin module. This key must be unique within the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>plugin where it is defined. Sometimes, in other contexts, you may need to</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>uniquely identify a module. Do this with the <strong>complete module key</strong>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A module with key fred in a plugin with key com.example.modules will have a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>complete key of com.example.modules:fred.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin's</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>administrative user interface.</td>
<td></td>
</tr>
</tbody>
</table>

**Elements**

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>to declare a localisation key for the value instead of text in the element</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>body. Use this element to describe the section.</td>
<td></td>
</tr>
</tbody>
</table>

**Interface**

Pre-Chain Action modules must implement the **PreChainAction** interface.

**Example**

Here is an example `atlassian-plugin.xml` file containing a Pre-Chain Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Pre-Chain action module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <preChainAction key="myPreChainAction"
                 class="com.atlassian.example.bamboo.actions.MyPreChainAction">
    <description>Sets up the MyBambooPlugin if the Chain has the plugin enabled</description>
  </preChainAction>
</atlassian-plugin>
```

**Pre-Job Action Module**

**Available:** Bamboo 2.7 and later
Purpose of this Module Type

Pre-Job Action modules allow plugins to define custom functionality that runs on the server before the Job is queued. This module shares similarities but differs from the Pre-Build Queued Action because it gives access to both the StageExecution and BuildContext rather than just the BuildContext.

Configuration

The root element for the Pre-Job module is preJobAction. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin’s administrative user interface.</td>
</tr>
</tbody>
</table>

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The ‘key’ attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
</tr>
</tbody>
</table>

Interface

Pre-Job Action modules must implement the PreJobAction interface.

Example

Here is an example atlassian-plugin.xml file containing a Pre-Job Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Pre-Job Action module test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>

  <preJobAction key="myPreJobAction"
                class="com.atlassian.example.bamboo.actions.MyPreJobAction">
    <description>Modifies the Job before it is queued</description>
  </preJobAction>
</atlassian-plugin>
```
### Pre-Stage Action Module

**Available:** Bamboo 2.7 and later

### Purpose of this Module Type

Pre-Stage Action modules allow plugins to define custom functionality that runs before the jobs for the given stage are queued for building. The module provides access to the `StageExecution` object.

### Configuration

The root element for the Pre-Stage Action module is `preStageAction`. It allows the following attributes and child elements for configuration:

#### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin's administrative user interface.</td>
</tr>
</tbody>
</table>

#### Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The ‘key’ attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
</tr>
</tbody>
</table>

### Interface

Pre-Stage Action modules must implement the `PreStageAction` interface.

### Example

Here is an example `atlassian-plugin.xml` file containing a Pre-Stage Action module:
Source Repository Module

Available: Bamboo 1.1 and later

On this page:
- Description
- Interface
- Sample Module Descriptor Element
- Extensions
  - RepositoryEventAware
  - PostConfigurableRepository
  - CustomVariableProviderRepository
  - Supporting "Import from Maven"
  - Examples

Description

A plugin module which defines a repository in Bamboo, such as CVS, Subversion, or Perforce.

Interface

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.

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>
```

Extensions

RepositoryEventAware

Description

The RepositoryEventAware interface allows you to instruct the repository to perform a custom action before and/or after the checkout/update occurs.

PostConfigurableRepository

The PostConfigurableRepository allows any custom configuration to be performed after the repository has been created or another has been selected. It is typically used to add and remove configuration after changes have been made to the repository configuration by a user.

CustomVariableProviderRepository

The CustomVariableProviderRepository allows to define custom variables that can be used to configure the plan.
Supporting "Import from Maven"

By implementing MavenPomAccessorCapableRepository on your repository module this interface allows your Source Repository Module to be used from the Import a Maven 2 Project feature.

The MavenPomAccessorCapableRepository.getMavenPomAccessor() should return your own implementation of the MavenPomAccessorAbstract class that provides the ability to checkout and find the Maven pom from the repository provided by the user.

Examples

Add a line similar to the one below to your repository descriptor in the atlassian-plugin.xml. This is similar to the "edit" resource

```
<resource type="freemarker" name="mavenPomCheckoutAccessEdit" location="templates/plugins/repository/p4RepositoryMavenPomCheckoutAccessEdit.ftl"/>
```

Here is the example Freemarker template:

```
[#if repository.perforceExecutableSet]
     <div class="infoBox">
       [@ww.text name='repository.p4.executableExists']
       [@ww.param]${repository.p4Executable}[/@ww.param]
     [/@ww.text]
   </div>
[#else]
     <div class="warningBox">
       [@ww.text name='repository.p4.noExecutable']
       [@ww.param]${req.contextPath}/admin/agent/configureSharedLocalCapabilities.action[/@ww.param]
     [/@ww.text]
   </div>
[/#if]

     [@ww.textfield labelKey='repository.p4.port' name='repository.p4.port' required='true'
     helpKey='perforce.fields' /
     [@ww.textfield labelKey='repository.p4.client' name='repository.p4.client' required='true'
     helpKey='perforce.fields' /
     [@ww.textfield labelKey='repository.p4.depot' name='repository.p4.depot' required='true'
     helpKey='perforce.fields' /
     [@ww.textfield labelKey='repository.p4.mavenPomCheckout.pomRelativePath'
     name='repository.p4.mavenPomCheckout.pomRelativePath' /
     [@ww.textfield labelKey='repository.p4.username' name='repository.p4.user'
     helpKey='perforce.fields' /
     [@ww.hidden name="temporary.p4.passwordChange" value="true" /
     [@ww.password labelKey='repository.p4.password' name='temporary.p4.password'
     helpKey='perforce.fields' required='false'/]
```

Task Plugin Module

Available: Bamboo 3.1 and later

On this page:
- Purpose of this Module Type
- Example
- Interface
- Attributes
- Elements

Purpose of this Module Type

A task plugin represents a unit of work during a Build. Each Job can now have multiple Tasks which run in order on a single Agent. If a Task fails the following tasks will not execute, with the exception of Final Tasks which will always execute.

Tasks are designed to be easily extensible. To get a basic Task up and running, you simply need to implement a single Java interface. However, there are optional extension points that allow you to add a configuration screen and other powerful features to your Task.

You can find in-depth details on creating a Task on the Tasks Overview page.

If you are looking to upgrade a Builder to a Task please see the LegacyBuilderToTaskConverter Plugin Module

Example
Minimal TaskType plugin

```xml
<taskType key="task.myExampleTask" name="Awesome Task"/>
```

Advanced TaskType plugin

```xml
<taskType key="task.builder.maven" name="Maven 1.x"
class="com.atlassian.bamboo.plugins.maven.task.Maven1BuildTask">
  <description>Execute one or more Maven 1 goals as part of your build</description>
  <category name="builder"/>
  <category name="test"/>
  <executable key="maven" nameKey="builder.maven.executableName"
    pathHelpKey="builder.maven.helpPath"/>
  <capabilityDefaultsHelper
class="com.atlassian.bamboo.plugins.maven.task.Maven1CapabilityDefaultsHelper"/>
  <configuration
class="com.atlassian.bamboo.plugins.maven.task.configuration.Maven1BuildTaskConfigurator" />
  <resource type="freemarker" name="edit"
    location="com/atlassian/bamboo/plugins/maven/task/configuration/maven1BuildTaskEdit.ftl"/>
  <resource type="freemarker" name="view"
    location="com/atlassian/bamboo/plugins/maven/task/configuration/maven1BuildTaskView.ftl"/>
</taskType>
```

Interface

All TaskType modules must implement the TaskType interface.

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.</td>
<td>Selecting the TaskType</td>
</tr>
<tr>
<td>name</td>
<td>✓</td>
<td>The human-readable name of the plugin module. This name will be displayed when configuring Tasks</td>
<td>Executing The Task</td>
</tr>
<tr>
<td>class</td>
<td>✓</td>
<td>Your TaskType class. It must implement TaskType and provide the logic for executing a Task</td>
<td></td>
</tr>
</tbody>
</table>

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description will be displayed to the user when selecting the TaskType to add to their Job.</td>
<td>Selecting the TaskType</td>
</tr>
<tr>
<td>category</td>
<td></td>
<td>Determines which categories the TaskType will be displayed in, in the TaskType picker. You can have multiple category definitions. Available categories are &quot;builder&quot;, &quot;test&quot;, &quot;deployment&quot;</td>
<td>Selecting the TaskType</td>
</tr>
<tr>
<td>executable</td>
<td></td>
<td>Allows you to define an Executable Type that users can add as Capabilities on their Agents. You can then add Requirements on these Capabilities</td>
<td>Tasks, Requirements And Capabilities</td>
</tr>
<tr>
<td>capabilityDefaultsHelper</td>
<td></td>
<td>Allows you to automatically detect Executables that the agent/server may have and add them as Capabilities</td>
<td>Tasks, Requirements And Capabilities</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
<td>Provide a way for users to configure your Task.</td>
<td>Configuring The Task</td>
</tr>
</tbody>
</table>
### LegacyBuilderToTaskConverter Plugin Module

Converting builders to tasks

Plugin implementers that reimplement their Builders as Task can also provide a utility that will be used to automatically convert Plans to use Tasks instead of Builders. This can be achieved by implementing LegacyBuilderToTaskConverter interface and adding the implementation to a plugin. The converter should extract the relevant parameters out of BuildConfiguration and create a list of appropriate TaskDefinitions (order according to desired execution order).

Plugin configuration example:

```
<builder2TaskConverter key="AntConverter" name="Ant Builder Converter"
class="com.atlassian.bamboo.plugins.ant.task.conversion.AntBuilder2TaskConverter">
  <description>A configuration converter for Ant builders</description>
  <builderKeyPattern>com.atlassian.bamboo.plugin.system.builder:ant</builderKeyPattern>
</builder2TaskConverter>
```

**builderKeyPattern** - regular expression; converter is used if builder plugin key matches the expression

### Tasks Overview

- Introduction
- The TaskType Module Definition
- Selecting the TaskType
- The Data Model.
- Executing The Task
  - Executing an external process
  - Reading Logs
  - Parsing Tests
  - Helpful Utilities
- Configuring The Task
  - Utilities
- Tasks, Requirements And Capabilities
- other
- Updating your builder configuration
- Agent Services

### Introduction

In 3.1 we have introduced Tasks. Each Job can now have multiple Tasks.

- Tasks have replaced the Builder.
- Tasks are run in order on a single Agent.
- If a task fails following tasks will not be executed (Final Tasks are an exception)
- Final tasks will always be executed regardless of the state of previous tasks so can be used to e.g. clean up tasks
- Tasks are executed after the Pre Build Plugins and Before the Post Build Plugins.
- What you choose to do in a task is entirely up to you!

### Special Considerations When Developing Tasks

- People can add multiple of your TaskType plugin to the one Job.
- People can put your task in the final tasks section so you task needs to be defensive to the situation where previous setup may not exist.
- It must be a plugins 1 plugin because it needs to be capable of running on remote agents

### The TaskType Module Definition

Tasks are designed to be easily extensible. To get a basic Task up and running, you simply need to implement a single Java interface. However, there are optional extension points that allow you to add a configuration screen and other powerful features to your Task.

#### Minimal TaskType Plugin

```
<taskType key="task.awesome" name="Awesome Task"
class="com.atlassian.bamboo.plugins.MyAwesomeTaskType" />
```

#### Advanced TaskType Plugin
**Selecting the TaskType**

When configuring the Job, the user is offered a TaskType picker. Your TaskType module will appear in this list.

![TaskType Picker](image)

The TaskType module name and description will be visible to assist the user in selecting the desired TaskType. The description is optional however we highly recommend including one.

You can also optionally provide categories to group the TaskType in the picker.

**Available Categories**

- builder
- test
- deployment

When a user selects a TaskType, they will have the opportunity to configure it (if required) and then it will be saved as a TaskDefinition. TaskDefinitions are stored against the BuildDefinition of a Job.
The Data Model.

TaskDefinition objects are what we store in the BuildDefinition and will be used whenever you are configuring a Task. When you are executing the TaskType we have provided lots of other useful information, so you will be dealing instead with the TaskContext. Both the TaskDefinition and TaskContext extend the TaskIdentifier interface which may be used when only the core information is required.

```
TaskIdentifier
long getId()
String getPluginKey()
String getUsuarioDescription()
boolean isFinalising()
```

```
TaskContext
BuildContext getBuildContext()
BuildLogger getBuildLogger()
ConfigurationMap getConfigurationMap()
File getRootDirectory()
File getWorkingDirectory()
```

Executing The Task

The minimal requirement for a TaskType Module is a class implementing the TaskType interface.

```java
@NotNull
TaskResult execute(@NotNull final TaskContext taskContext) throws TaskException;
```

The TaskContext in which you can find any build related information and any configuration information.

TaskResult The TaskResult allows you to store any resultData for later use and the TaskState, to indicate whether the task Failed or Passed. Bamboo will use this TaskState to determine whether to continue executing other tasks.

TaskException If something goes wrong you can also throw the TaskException

Things to consider

- TaskTypes are run on the Agent so must not use stuff your not allowed to use

Executing an external process

ExternalProcess ProcessService
ExternalProcessBuilder

Reading Logs
LogInterceptor

Parsing Tests
TestCollationService
Helpful Utilities

EnvironmentVariableAccessor

Configuring The Task

If your TaskType requires configuring you can provide a configuration class. The configuration class must implement TaskConfigurator. You will also need to provide freemarker files for both edit and view of the configuration.

```java
private static final List<String> FIELDS_TO_COPY = ImmutableList.of("goal", "environment");

@NotNull
@Override
public Map<String, String> generateTaskConfigMap(@NotNull final ActionParametersMap params,
                                                  @Nullable final TaskDefinition previousTaskDefinition)
        {
            final HashMap<String,String> config = Maps.newHashMap();
            taskConfiguratorHelper.populateTaskConfigMapWithActionParameters(config, params,
                                                                            FIELDS_TO_COPY);
            return config;
        }

@Override
public void populateContextForEdit(@NotNull Map<String, Object> context, @NotNull
                                     TaskDefinition taskDefinition)
        {
            taskConfiguratorHelper.populateContextWithConfiguration(context, taskDefinition,
                                                                       FIELDS_TO_COPY);
        }
```

The TaskConfiguratorHelper also contains utilities to assist some generic fields such as selecting a JDK, and setting the location of where to find Test Results. However these methods rely on specifically named parameters, so if you want to make use of this functionality you should use TaskConfigConstants for your field naming.

Tasks, Requirements And Capabilities

TaskExecutableType – private class, but public functionality
TaskRequirementSupport
CapabilityDefaultsHelper
TaskConfiguratorHelper: add system requirement.

other

TaskManager

TaskModuleDescriptor?? public or not

TaskProcessCommandDecorator

Updating your builder configuration

TaskProcessCommandDecoratorModuleDescriptor ?? public or not

Agent Services

These Services and Managers are marked with @RemoteAgentSupported
Trigger Reason Module

Available: Bamboo 2.0 and later

On this page:
- Description
- Interface
- Sample Module Descriptor Element

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

```
<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>
```

Notification Plugin Modules

Notification plugin modules allow developers to add new types of email and instant messaging notifications to the Bamboo

Notification Condition Module

Deprecated since Bamboo 2.2 Please use the Notification Type Module instead (see Building a Notification Plugin as well)

Notification Recipient Module

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

Notification Type Module

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?).

Notification Condition Module
On this page:

- Description
- Interface
- Sample Module Descriptor Element
- Other Information regarding the NotificationCondition class

Description

Deprecated since Bamboo 2.2 Please use the Notification Type Module instead (see Building a Notification Plugin as well)

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.

Notification Recipient Module

On this page:

- Description
  - Notification Transport
- Interface
- Sample Module Descriptor

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

**Available:** Bamboo 2.2 and later

On this page:
- Description
- Interface
- Sample Module Descriptor

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?).

NotificationTypes can be scoped to determine where it can be configured. There are three options (though poorly named):

- **Chain** - For Plan level notifications
- **Plan** - for Job level notifications
- **System** - for global notifications (configured in the administration section of Bamboo)

If you don't provide a scope, it will default to a Job level notification. Also, note that whilst there are Job and Plan level notifications, they are both configured at the plan level. Job notifications will generally apply to all jobs within the Plan.

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="buildCompleted.allBuilds" name="All Jobs Completed" weight="40">
  <description>Notification sent for every job that finishes building</description>
  <scope>plan</scope>
</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. There are two places where you can configure notifications. Against a plan, or globally in the Administration section in Bamboo. When configuring notifications for a plan, you can select plan level notifications or job level notifications (since Bamboo 2.7). To determine where your notificationType will be displayed use the scope option when defining the plugin.

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 and 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.
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 interface. There is an abstract class which you can extend that implements a lot of these methods for you.

com.atlassian.bamboo.notification.Notification
com.atlassian.bamboo.notification.AbstractNotification

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.
@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
        // 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 wont 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.

Building a Notification Plugin - copy for Bamboo 2.7

This page is a draft in progress and visible to atlassian-staff only.

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 and 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.
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.
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 prefer.
would like to view.

System Plugin Modules

System modules allow you to extend various internal parts of Bamboo such as agent allocation, Spring components, Lucene indexes and deletion.

Capability Type Module

The Capability Type Module is used for registering custom capabilities for your Bamboo plugin.

Component Module

Component plugin modules enable you to add components to Bamboo's internal component system (powered by Spring).

Index Reader Module

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.

Plan Deletion Interceptor Action Module

The Plan Deletion Interceptor Action module before the plan and its results have been deleted. Any plugins that implement this interface will be run inline with the methods on DeletionService. If any implementations fail to execute (throw an exception, etc) the plan will be deleted regardless of the success or failure of this module.

Post Build Index Writer Module

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 require a re-index of a particular build (result), and when you run the re-index all action under the Administration tab.

Post Chain Index Writer Module

The Post Chain Index Writer modules allows you to write your custom data for a build into the index, which allows for future retrieval in your custom Report Module. The Post Chain Index Writer modules will be invoked in three places in Bamboo: when a build completes and it indexes, operations which require a re-index of a particular chain (result), and when you run the re-index all action under the Administration tab.

Servlet Context Listener Plugin Module

Servlet Context Parameter Plugin Module

Servlet Filter Plugin Module

Servlet Plugin Module

Capability Type Module

On this page:

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example

Purpose of this Module Type

The Capability Type Module is used for registering custom capabilities for your Bamboo plugin.

You may want to use this module to:

- Provide a human readable name for your plugins capability
- Provide custom template for editing the capability
- Group multiple yet related capabilities. For example, Bamboo uses this module for its Mercurial support to define both the Mercurial and SSH executables used by Mercurial
Configuration

The root element for the Capability Type module is **capabilityType**. It allows the following attributes and child elements for configuration:

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to <strong>creating plugin module instances</strong>.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the <strong>complete module key</strong>. A module with key <strong>fred</strong> in a plugin with key <strong>com.example.modules</strong> will have a complete key of <strong>com.example.modules:fred</strong>.</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin's administrative user interface.</td>
<td></td>
</tr>
<tr>
<td>sortOrder</td>
<td>✓</td>
<td>Defines the order that the capability will be displayed on the user interface. This value must be unique.</td>
<td></td>
</tr>
</tbody>
</table>

### Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.</td>
<td></td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>Using the &quot;edit&quot; attribute it is possible to provide a custom template for editing your capability or capabilities</td>
<td></td>
</tr>
</tbody>
</table>

### Interface

Capability Type modules must implement the **AbstractCapabilityTypeModule** class.

By implementing the **CapabilityDefaultsHelper** interface on your CapabilityTypeModule the interface makes the auto-detection of capabilities possible via the **addDefaultCapabilities** method.

### Example

Here is an example **atlassian-plugin.xml** file containing a Capability Type module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>My Capability Type module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <capabilityType key="myCapabilityTypeModule" name="My Capability Type Module" class="com.atlassian.example.MyCapabilityTypeModule" sortOrder="1000">
    <description>A Custom Capability Type</description>
    <resource type="freemarker" name="edit" location="/com/atlassian/example/editMyCapability.ftl"/>
  </capabilityType>
</atlassian-plugin>
```

Here is an example of an "edit" template used in Bamboos Mercurial support:
Component Module

Available: Bamboo 2.6 and later

Purpose of this Module Type

Component plugin modules enable you to add components to Bamboo’s internal component system (powered by Spring).

Component Plugin Module

Each component module adds a single object to Bamboo’s component management system.

Other plugins and objects within Bamboo can then be autowired with your component. This is very useful for having a single component that is automatically passed to all of your other plugin modules (i.e. a Manager object).

Here is an example atlassian-plugin.xml file containing a single component module:

```xml
<atlassian-plugin name="Sample Component" key="bamboo.extra.component">
...
  <component name="Keyed Test Component"
    key="testComponent"
    alias="bogusComponent"
    class="com.atlassian.bamboo.plugin.descriptor.BogusComponent" />
...
</atlassian-plugin>
```

- the `name` attribute represents how this component will be referred to in the interface.
- the `key` attribute represents the internal, system name for your component.
- the `class` attribute represents the class of the component to be created
- the `alias` attribute represents the alias this component will be stored with. This element is optional, if not specified the module `key` will be used instead.

Accessing Your Components

Accessing your components is extremely simple.

Autowired Objects

If your object is being autowired (for example another plugin module or an XWork action), the easiest way to access a component is to add a basic Java setter method.

For example, if you use the above BogusComponent module your object would retrieve the component as follows:

```java
public void setBogusComponent(BogusComponent bogusComponent)
{
    this.bogusComponent = bogusComponent;
}
```

Non-autowired Objects

If your object is not being autowired, you may need to retrieve the component explicitly. This is done via the ContainerManager like so:

```java
BogusComponent bc = (BogusComponent) ContainerManager.getComponent("bogusComponent");
```

Notes
Some issues to be aware of when developing a component:

- One component module can depend on another component module but be careful of circular references (i.e., A requires B, B requires A).
- The component “namespace” is flat at the moment, so choose a sensible alias for your component.

### RELATED TOPICS

Installing a Plugin

### Index Reader Module

**Available:** Bamboo 1.0 and later

**On this page:**
- Description
- Interface
- Sample Module Descriptor Element

**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"
<description>Reads the clover result from an index document and populates into build result summary</description>
</indexReader>
```

### Plan Deletion Interceptor Action Module

**Available:** Bamboo 2.7 and later

**On this page:**
- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Interface
- Example

**Purpose of this Module Type**

The Plan Deletion Interceptor Action module before the plan and its results have been deleted. Any plugins that implement this interface will be run inline with the methods on DeletionService. If any implementations fail to execute (throw an exception, etc.) the plan will be deleted regardless of the success or failure of this module.

Note that all derivatives of Plan, such as Job and Chain, will be passed through this module. You may need to use the Narrow utility to cast to the correct type.

**Configuration**

The root element for the Plan Deletion Interceptor Action module is planDeletionInterceptorAction. It allows the following attributes and child elements for configuration:

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
</table>

The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.

The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred.

The human-readable name of the plugin module. Only used in the plugin's administrative user interface.

The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.

Plan Deletion Interceptor Action modules must implement the PlanDeletionInterceptorAction interface.

Here is an example atlassian-plugin.xml file containing a Plan Deletion Interceptor Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
    <plugin-info>
        <description>A Plan Deletion Interceptor Action module type test</description>
        <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
        <version>1.0</version>
    </plugin-info>

    <planDeletionInterceptorAction key="myPlanDeletionInterceptorAction"
        class="com.atlassian.example.bamboo.actions.MyPlanDeletionInterceptorAction">
        <description>Removes plugin configuration from the administration configuration before the plan is deleted</description>
        </planDeletionInterceptorAction>
</atlassian-plugin>
```

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.

### Available

Bamboo 1.0 and later
Interface
Post Build Index Writer modules must implement the `com.atlassian.bamboo.index.CustomPostBuildIndexWriter` interface.

Sample Module Descriptor Element

```
<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>
```

Post Chain Index Writer Module

**Available:** Bamboo 2.7 and later

On this page:
- Purpose of this Module Type
- Configuration
- Attributes
- Elements
- Interface
- Example

Purpose of this Module Type

The Post Chain Index Writer modules allows you to write your custom data for a build into the index, which allows for future retrieval in your custom Report Module. The Post Chain Index Writer modules will be invoked in three places in Bamboo: when a build completes and it indexes, operations which requires a re-index of a particular chain (result), and when you run the re-index all action under the Administration tab.

The Post Chain Index Writer should always be written in conjunction with a Index Reader Module which will be able to retrieve the data in the index.

Configuration

The root element for the Post Chain Index Writer module is `postChainIndexWriter`. It allows the following attributes and child elements for configuration:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. Only used in the plugin's administrative user interface.</td>
</tr>
</tbody>
</table>

Elements

The table summarises the elements. The sections below contain further information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The description of the plugin module. The ‘key’ attribute can be specified to declare a localisation key for the value instead of text in the element body. Use this element to describe the section.

### Interface

Post Chain Index Writer modules must implement the `PostChainIndexWriter` interface.

### Example

Here is an example `atlassian-plugin.xml` file containing a Post Chain Index Writer Action module:

```xml
<atlassian-plugin name="Hello World" key="example.plugin.helloworld">
  <plugin-info>
    <description>A Post Chain Index Writer module type test</description>
    <vendor name="Atlassian Software Systems" url="http://www.atlassian.com"/>
    <version>1.0</version>
  </plugin-info>
  <postChainIndexWriter key="myChainIndexWriter" name="Example Post Chain Index Writer"
    class="com.atlassian.example.bamboo.actions.MyChainIndexWriter">
    <description>Writes the My Example Plugin Results to the Lucene index</description>
  </postChainIndexWriter>
</atlassian-plugin>
```

### Servlet Context Listener Plugin Module

**Available:** Bamboo 2.5 and later

#### On this page:
- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Example
- Notes

#### Purpose of this Module Type

Servlet Context Listener plugin modules allow you to deploy Java Servlet context listeners as a part of your plugin. This helps you to integrate easily with frameworks that use context listeners for initialisation.

#### Configuration

The root element for the Servlet Context Listener plugin module is `servlet-context-listener`. It allows the following attributes and child elements for configuration:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to <a href="#">creating plugin module instances</a>. The servlet context listener Java class. Must implement <code>javax.servlet.ServletContextListener</code>.</td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
</tr>
<tr>
<td>key</td>
<td>🌐</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the <a href="#">complete module key</a>. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>. I.e. the identifier of the context listener.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>N/A</td>
</tr>
</tbody>
</table>
The human-readable name of the plugin module. I.e. the human-readable name of the listener. | The plugin module.
---|---
Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-OSGi plugins. | false

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. I.e. the description of the listener.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

Here is an example atlassian-plugin.xml file containing a single servlet context listener:

```xml
<atlassian-plugin>
  ...
</atlassian-plugin>
```

**Notes**

Some information to be aware of when developing or configuring a Servlet Context Listener plugin module:

- The servlet context you listen for will not be created on web application startup. Instead, it will be created the first time a servlet or filter in your plugin is accessed after each time it is enabled, triggering a new instance of your listener followed by the calling of the listener's `contextCreated()` method. This means that if you disable a plugin containing a listener and re-enable it again, the following will happen:
  1. The `contextDestroyed()` method will be called on your listener after the plugin was disabled.
  2. A new servlet context will be created after the plugin was re-enabled.
  3. Your listener will be instantiated.
  4. The method `contextCreated()` on your listener will be called.

**Servlet Context Parameter Plugin Module**

**Available:** Bamboo 2.5 and later

**On this page:**

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
- Example
- Notes

**Purpose of this Module Type**

Servlet Context Parameter plugin modules allow you to set parameters in the Java Servlet context shared by your plugin's servlets, filters, and listeners.

**Configuration**

The root element for the Servlet Context Parameter plugin module is `<servlet-context-param>`. It allows the following attributes and child elements for configuration:

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
</tbody>
</table>
key

The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred. i.e. The identifier of the context parameter.

name

The human-readable name of the plugin module. i.e. The human-readable name of the context parameter.

system

Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-OSGi plugins.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. i.e. the description of the listener.</td>
<td>N/A</td>
</tr>
<tr>
<td>param-name</td>
<td></td>
<td>The servlet context parameter name.</td>
<td>N/A</td>
</tr>
<tr>
<td>param-value</td>
<td></td>
<td>The servlet context parameter value.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Example

Here is an example atlassian-plugin.xml file containing a single servlet context parameter:

```

```

Notes

Some information to be aware of when developing or configuring a Servlet Context Parameter plugin module:

- This parameter will only be available to servlets, filters, and context listeners within your plugin.

Servlet Filter Plugin Module

Available: Bamboo 2.5 and later

On this page:

- Purpose of this Module Type
- Configuration
  - Attributes
  - Elements
  - Examples
- Example
- Accessing your Servlet Filter
- Notes

Purpose of this Module Type

Servlet Filter plugin modules allow you to deploy Java Servlet filters as a part of your plugin, specifying the location and ordering of your filter. This allows you to build filters that can tackle tasks like profiling and monitoring as well as content generation.

Configuration

The root element for the Servlet Filter plugin module is `servlet-filter`. It allows the following attributes and child elements for configuration:

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>The class which implements this plugin module. The class you need to provide depends on the module type. For example, Confluence theme, layout and colour-scheme modules can use classes already provided in Confluence. So you can write a theme-plugin without any Java code. But for macro and listener modules you need to write your own implementing class and include it in your plugin. See the plugin framework guide to creating plugin module instances. The servlet filter Java class must implement <code>javax.servlet.Filter</code>.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Required</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td></td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key fred in a plugin with key com.example.modules will have a complete key of com.example.modules:fred. I.e. the identifier of the servlet filter.</td>
<td></td>
</tr>
<tr>
<td>location</td>
<td></td>
<td>The position of the filter in the application's filter chain. If two plugins provide filters at the same position, the 'weight' attribute (see below) is evaluated.</td>
<td>before-dispatch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- after-encoding - Near the very top of the filter chain in the application, but after any filters which ensure the integrity of the request.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- before-login - Before the filter that logs in the user with any authentication information included in the request.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- before-decoration - Before the filter which does decoration of the response, typically with Sitemesh.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- before-dispatch - At the end of the filter chain, before any servlet or filter which handles the request by default.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. I.e. the human-readable name of the filter.</td>
<td></td>
</tr>
<tr>
<td>system</td>
<td></td>
<td>Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-OSGi plugins.</td>
<td>false</td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td>The weight of the filter, used to decide which order to place the filter in the chain for filters which have specified the same 'location' attribute (see above). The higher weight, the lower the filter's position.</td>
<td>100</td>
</tr>
</tbody>
</table>

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. I.e. the description of the filter.</td>
<td></td>
</tr>
<tr>
<td>init-param</td>
<td></td>
<td>Initialisation parameters for the filter, specified using param-name and param-value sub-elements, just as in web.xml. This element and its child elements may be repeated.</td>
<td>N/A</td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>A resource for this plugin module. This element may be repeated. A 'resource' is a non-Java file that a plugin may need in order to operate. Refer to Adding Plugin and Module Resources for details on defining a resource.</td>
<td>N/A</td>
</tr>
<tr>
<td>url-pattern</td>
<td></td>
<td>The pattern of the URL to match. This element may be repeated. The URL pattern format is used in Atlassian plugin types to map them to URLs. On the whole, the pattern rules are consistent with those defined in the Servlet 2.3 API. The following wildcards are supported:</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- * matches zero or many characters, including directory slashes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ? matches zero or one character</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Examples</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- /mydir/* matches /mydir/myfile.xml</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <em>/admin/.</em>???ml matches /mydir/otherdir/admin/myfile.html</td>
<td></td>
</tr>
<tr>
<td>dispatcher</td>
<td></td>
<td>Determines when the filter is triggered. You can include multiple dispatcher elements. If this element is present, its content must be one of the following: REQUEST, INCLUDE, FORWARD, ERROR. Note: This element is only available in Plugin Framework 2.5 and later.</td>
<td>Filter will be triggered on all conditions</td>
</tr>
</tbody>
</table>

**Example**

Here is an example atlassian-plugin.xml file containing a single servlet filter:

```
<atlassian-plugin>
  ...
  <servlet-filter>
    <description>My New Filter</description>
    <key>fred</key>
    <system>false</system>
  </servlet-filter>
  ...
</atlassian-plugin>
```

**Accessing your Servlet Filter**

Accessing your Servlet Filter
Your servlet will be accessible within the Atlassian web application via each `url-pattern` you specify, but unlike the Servlet Plugin Module, the `url-pattern` is relative to the root of the web application.

For example, if you specify a `url-pattern` of `/helloworld` as above, and your Atlassian application was deployed at `http://yourserver/jira` — then your servlet filter would be accessed at `http://yourserver/jira/helloworld`.

**Notes**

Some information to be aware of when developing or configuring a Servlet Filter plugin module:

- Your servlet filter’s `init()` method will not be called on web application startup, as for a normal filter. Instead, this method will be called the first time your filter is accessed after each time it is enabled. This means that if you disable a plugin containing a filter or a single servlet filter module, and re-enable it again, the filter will be re-created and its `init()` method will be called again.
- Because servlet filters are deployed beneath root, be careful when choosing each `url-pattern` under which your filter is deployed. If you plan to handle the request in the filter, it is recommended to use a value that will always be unique to the world!
- Some application servers, like WebSphere 6.1, won’t call servlet filters if there is no underlying servlet to match the URL. On these systems, you will only be able to create a filter to handle normal application URLs.

**Servlet Plugin Module**

**Available:** Bamboo 2.5 and later

**On this page:**

- Purpose of this Module Type
- Configuration
- Attributes
- Elements
- Example
- Accessing your Servlet
- Notes

**Purpose of this Module Type**

Servlet plugin modules enable you to deploy Java servlets as part of your plugins.

**Configuration**

The root element for the Servlet plugin module is `servlet`. It allows the following attributes and child elements for configuration:

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>✓</td>
<td>The servlet Java class. Must be a subclass of <code>javax.servlet.http.HttpServlet</code>. See the plugin framework guide to creating plugin module instances.</td>
<td></td>
</tr>
<tr>
<td>disabled</td>
<td></td>
<td>Indicate whether the plugin module should be disabled by default (value='true') or enabled by default (value='false').</td>
<td>false</td>
</tr>
<tr>
<td>i18n-name-key</td>
<td></td>
<td>The localisation key for the human-readable name of the plugin module.</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>✓</td>
<td>The identifier of the plugin module. This key must be unique within the plugin where it is defined. Sometimes, in other contexts, you may need to uniquely identify a module. Do this with the complete module key. A module with key <code>fred</code> in a plugin with key <code>com.example.modules</code> will have a complete key of <code>com.example.modules:fred</code>. i.e. the identifier of the servlet.</td>
<td>N/A</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The human-readable name of the plugin module. i.e. the human-readable name of the servlet.</td>
<td>The plugin key.</td>
</tr>
<tr>
<td>system</td>
<td></td>
<td>Indicates whether this plugin module is a system plugin module (value='true') or not (value='false'). Only available for non-O SGi plugins.</td>
<td>false</td>
</tr>
</tbody>
</table>

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
<td>The description of the plugin module. The 'key' attribute can be specified to declare a localisation key for the value instead of text in the element body. i.e. the description of the servlet.</td>
<td></td>
</tr>
<tr>
<td>init-param</td>
<td></td>
<td>Initialisation parameters for the servlet, specified using <code>param-name</code> and <code>param-value</code> sub-elements, just as in <code>web.xml</code>. This element and its child elements may be repeated.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
A resource for this plugin module. This element may be repeated. A ‘resource’ is a non-Java file that a plugin may need in order to operate. Refer to Adding Plugin and Module Resources for details on defining a resource.

The pattern of the URL to match. This element may be repeated.

The URL pattern format is used in Atlassian plugin types to map them to URLs. On the whole, the pattern rules are consistent with those defined in the Servlet 2.3 API. The following wildcards are supported:

- * matches zero or many characters, including directory slashes
- ? matches zero or one character

Examples

- /mydir/* matches /mydir/myfile.xml
- */admin/*.??ml matches /mydir/otherdir/admin/myfile.html

---

### Accessing your Servlet

Your servlet will be accessible within the Atlassian web application via each `url-pattern` you specify, beneath the `/plugins/servlet` parent path.

For example, if you specify a `url-pattern` of `/helloworld` as above, and your Atlassian application was deployed at `http://yourserver/jira`— then your servlet would be accessed at `http://yourserver/jira/plugins/servlet/helloworld`.

### Notes

Some information to be aware of when developing or configuring a Servlet plugin module:

- Your servlet's `init()` method will not be called on web application startup, as for a normal servlet. Instead, this method will be called the first time your servlet is accessed after each time it is enabled. This means that if you disable a plugin containing a servlet, or a single servlet module, and re-enable it again, the servlet is re-instantiated and its `init()` method will be called again.
- Because all servlet modules are deployed beneath a common `/plugins/servlet` root, be careful when choosing each `url-pattern` under which your servlet is deployed. It is recommended to use a value that will always be unique to the world!

---

### Bamboo Event Listeners

**Available:** Bamboo 2.2 and later

### On this page:

- Description
- Interface
- Sample Module Descriptor Element
- Available Events to listen to

#### Description

The Bamboo Event Listener module allows developers 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.

It would be wise to note that currently Bamboo does not support using Atlassian Event 2.x listeners or events from this extension type.

#### 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
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>AgentOfflineEvent</td>
<td>Occurs when a specific agent goes offline</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 the Bamboo system. 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>BuildCanceledEvent</td>
<td>Occurs when a Job or Build is canceled via the PlanExecutionManager</td>
</tr>
<tr>
<td>BuildCommentedEvent</td>
<td>Occurs when a result has been commented on by the User</td>
</tr>
<tr>
<td>BuildCompletedEvent</td>
<td>Occurs after a Job or Build has completed and results saved to the database but not necessarily before or after any CustomBuildCompleteActions are performed</td>
</tr>
<tr>
<td>BuildConfigurationUpdatedEvent</td>
<td>Occurs when the build configuration has been updated</td>
</tr>
<tr>
<td>BuildCreatedEvent</td>
<td>Occurs when a Job or Build has been created</td>
</tr>
<tr>
<td>BuildDeletedEvent</td>
<td>Occurs when a Job or Build has been deleted</td>
</tr>
<tr>
<td>BuildFinishedEvent</td>
<td>Occurs when BuildExecutionManager.finishBuild() has been called on a Job or Build. This is guaranteed to be fired even if the system has encountered an irrecoverable error and as such a corresponding BuildResultSummary may not be available.</td>
</tr>
<tr>
<td>BuildQueuedEvent</td>
<td>Occurs when a Job or Build has been queued</td>
</tr>
<tr>
<td>BuildQueueTimeoutEvent</td>
<td>Occurs when a Job or Build has timed out whilst being on the queue.</td>
</tr>
<tr>
<td>BuildHungEvent</td>
<td>Occurs when Bamboo’s build monitoring detects the build has met the configured build hanging criteria</td>
</tr>
<tr>
<td>BuildRequirementUpdatedEvent</td>
<td>Occurs when a Requirement has changed on a Job or Build</td>
</tr>
<tr>
<td>BuildResultDeletedEvent</td>
<td>Occurs when a result for a Job or Build has been deleted</td>
</tr>
<tr>
<td>BuildResultsSummaryUpdatedEvent</td>
<td>Occurs when a result for a Job or Build has been updated. Currently the only way to edit a build summary is editing the JIRA issues associated with the build.</td>
</tr>
<tr>
<td>BuildsMovedEvent</td>
<td>Occurs when moving plans between projects.</td>
</tr>
<tr>
<td>BuildTriggeredEvent</td>
<td>In Bamboo 2.6 and earlier, this event occurs after a bamboo has detected that the Build needs building, before the build is placed on the queue. In Bamboo 2.7 and later, this event occurs when a job has been executed.</td>
</tr>
<tr>
<td>ChangeDetectionRequiredEvent</td>
<td>Event which signifies that Bamboo should check repositories for changes</td>
</tr>
<tr>
<td>ChainCompletedEvent</td>
<td>Occurs when a Plan finishes executing</td>
</tr>
<tr>
<td>ChainCreatedEvent</td>
<td>Occurs when a Plan is created</td>
</tr>
<tr>
<td>ChainDeletedEvent</td>
<td>Occurs when a Plan is deleted</td>
</tr>
<tr>
<td>ChainResultDeletedEvent</td>
<td>Occurs when a Plan Result has been deleted</td>
</tr>
<tr>
<td>DeletionFinishedEvent</td>
<td>Occurs after the deletion action has finished</td>
</tr>
<tr>
<td>ElasticConfigUpdatedEvent</td>
<td>Occurs when an Elastic Configuration has been updated</td>
</tr>
<tr>
<td>EmailEvent</td>
<td>Occurs when an email is to be sent</td>
</tr>
<tr>
<td>IMEvent</td>
<td>Occurs when an instant message is to be sent</td>
</tr>
<tr>
<td>InitialBuildRequiredEvent</td>
<td>Occurs when bamboo detects that an initial clean build is required for a plan (usually after plan creation or importing data)</td>
</tr>
<tr>
<td>Event Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JobCompletedEvent</td>
<td>Occurs when a [Job] has finished executing</td>
</tr>
<tr>
<td>PostBuildCompletedEvent</td>
<td>Thrown after the build is saved AND the build process has been completely cleaned up</td>
</tr>
</tbody>
</table>
| ServerStartedEvent             | This event is fired when the Bamboo server has completed its initialisation process — that is, after Bamboo has completed the following tasks:  
  - connected to its database  
  - initialised all plugins  
  - local agents have started |
| StageCompletedEvent            | Occurs when a stage has completed executing                                 |

Most of these events either extend a BuildEvent (build key available) or a BuildResultEvent (build key and build number available).

**Bamboo's Build Process**

A Chain (what users see as a Plan in the UI) consists of one or more Stages and each Stage consists of one or more Jobs. Before progressing to the next Stage, every Job in the previous Stage must have completed successfully. There is no order to Jobs within a Stage so when a Stage is executed, all Jobs in that Stage are placed in the queue immediately.

All triggering strategies now work at the Chain level. Chains do not take Job information into account when deciding whether to trigger. If repository polling or repository trigger strategies are selected, only changes detected against the Chain's are used to determine if the build will be triggered.

Below is a diagram which shows Bamboo's build process flow, as well as the available plugin modules you can build.

**Build Process For Chain**
Build Process Per Job

Further Notes
1. **Build is Triggered** — Changes are detected via polling (against the Chain's repository configuration only), manual builds, etc.
2. **Preparation for Build** — The Build Number is determined then ResultsSummary objects and BuildContexts are created for the Chain and every Job within the Chain. The build number information, latest revision keys and ResultSummaries are persisted to the database.
3. **Job Queued** — The server decides which agents can execute the job and queues the job.
4. **Agent Picks Up Job** — The capability context is set. The job is removed from queue. The agent begins looping through the build tasks.
5. **Retrieve Source** — 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.
6. **Prepare Job** — The agent runs the PrepareBuildTask. This begins streaming the logs back to the server. The agent also runs the CustomPreBuildAction plugin point.
7. **Executes the Job** — Timer begins. The agent runs the Builder#executeBuild. The CommandDecorator plugin will be run against the command line just before execution. After the Builder has been run, all the CustomBuildProcessors are run. Timer is stopped.
8. **Process Job Result** — The server runs CustomBuildProcessorsServer, checks if the job has passed and saves and indexes the result. The server also fires off the BuildCompletedEvent and PostBuildCompletedEvent events, and executes any CustomPostBuildCompletedActions.
9. **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>getDefaultJiraServer()</td>
</tr>
</tbody>
</table>
### BuildManager

**Build Plans**

Server

- `getBuildByKey(String key)`, `getAllBuildsForRead()`, `getBuildResults(Build build, Integer buildNumber)`

### BuildResultsSummaryManager

**Build Results**

Server

- `getBuildResultsSummary(Build build, int buildNumber)`, `getLastBuildSummary(String planKey)`

### BuildExecutionUpdateManager

Build Loggers and other running details to be passed back to the server

Server and Agent

- `getBuildLogger(@NotNull String buildPlanKey)`

### InstantMessagingServerManager

Connection Details for Instant Messaging Server

Server

- `getAllInstantMessagingServers()`, `getInstantMessagingServer(long instantMessagingServerId)`

### LocalAgentManager

Manages build agents both remote and local from the server

Server

- `getAllLocalAgents()`, `getAllRemoteAgents()`, `createLocalAgent(PipelineDefinition pipelineDefinition)`, `getAgent(long agentId)`

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 your 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
privateBuildContext buildContext;

public void init(@NotNullBuildContext buildContext)
{
    this.buildContext = buildContext;
}
```

The build context provide you with:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TriggerReason</td>
<td>Interface that encapsulates the reason a build was triggered</td>
</tr>
<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 a vcs revision key, then run the builder command and then collect all the various artifacts.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>Used to perform source code related activities</td>
</tr>
</tbody>
</table>
Builder

The builder class knows how to execute your build and determine whether or not the build has passed.

Custom Configuration

Any other configuration you want to add to the builds.

Artifact Definitions

Which file patterns to copy as artifacts.

```java
BuildContext.getBuildPlanDefinition().getArtifactDefinitions().add(artifact)
```

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><code>BuildContext.getRepository().getSourceCodeDirectory(planKey);</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Repository urls</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

```
((SvnRepository)
BuildContext.getBuildPlanDefinition().getRepository()).getRepositoryUrl()
```

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.

```java
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

```java
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/`.

BuildResultSummary

- Commit Information
- Trigger Reason
- Build State
- Test Counts
- The agent this build was built on
Bamboo 3.1 Documentation

- Vcs Revision Key
- Related Jira Issues
- Custom Build Data
- Labels

**BuildResult**

- Test results
- Artifact information

**AdministrationConfiguration**

The AdministrationConfiguration stores Bamboo’s system wide properties such as the default url, instance name, global variables (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
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");
        // Return value should cascade up to the global context
        assertEquals(1 , returnValue.size());
        assertEquals(200 , returnValue.get("testvalue"));
        assertEquals(returnValue, bandanaManager.getValue(PlanAwareBandanaContext.GLOBAL_CONTEXT,
                                               "system.bamboo.repository.Hello:mykey"));
        assertNull(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");
        // Now that the return value has a specific value for the plan, it should just display
        that info
        assertEquals(1 , returnValue2.size());
        assertEquals(100, returnValue2.get("testvalue"));
        assertEquals(1 , returnValue2.size());
        assertEquals(100 , returnValue2.get("testvalue"));
        EasyMock.verify(mockPlan);
    }
}
```

### Downloadable Plugin Resources

**Available:** Bamboo 2.3 and later

**On this page:**
- **Description**
- **Defining a Single Downloadable Resource**
- **Defining a Directory of Downloadable Resources**
- **Referring to Downloadable Resources**

**Description**

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.

```xml
<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.

```xml
<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

<table>
<thead>
<tr>
<th>Available: Bamboo 2.3 and later</th>
</tr>
</thead>
</table>

On this page:

- Description
- Defining a Single Web Resource
- Referring to Web Resources

Description

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.

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.

```xml
<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 Freemarker 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}/{plugin version}/{system counter}/c/download/resources/{plugin key}:{module key}/{resource name}
```

For the above example:

```html
<link type="text/css" rel="stylesheet" href="http://bamboo.example.com/s/170/1.0/1/_/download/resources/com.atlassian.bamboo.plugin.autofavourite:autofavourite-resources/autofavourite.css">
```

will be inserted in the header of the page.

Differences between Plugins1 and Plugins2

On this page:

- Introduction
  - Plugins Framework
  - Plugin versions
    - Plugins1
    - Plugins2
- Development and Installation
  - Configuration
  - Installation
- Java packages

Introduction

Plugins Framework

Bamboo uses a library called the Atlassian Plugin Framework to manage its plugins. This library is developed separately to Bamboo (it is a shared library used by all the Atlassian products) and therefore has its own versioning.

For example, Bamboo v2.4 uses Plugin Framework v2.3, Bamboo v2.3 uses Plugin Framework v2.2.

Plugin versions

Before Plugins Framework v2 (Bamboo 2.2 and earlier), plugins were installed in Bamboo by adding the plugin JAR file to Bamboo’s classpath (WEB-INF/lib). This style of plugin is referred to as "Plugins1".

Plugins Framework v2 (Bamboo 2.3 and higher) introduced a new way to install plugins. Plugins can be installed in Bamboo’s "installed-plugins" directory (under the Bamboo home directory) this sub-folder doesn’t exist? from where they will be installed and managed by an OSGi Container. This type of plugin is referred to as "Plugins2".

It is important to note that Plugins2 is not considered a replacement for Plugins1. Each provides some advantages and disadvantages. Plugin developers should consider their particular plugin, and choose which plugin type to use accordingly.

Plugins1

Plugins1 was the original way to install and manage plugins. In Bamboo, these are installed by placing the plugin JAR in your WEB-INF/lib/ directory. (This is a "static plugin". The framework also has another form called a "dynamic Plugins1 plugin", but these are not supported in Bamboo). This means the Java classes in your plugin live in the core application classpath, and are loaded just the same as the core Bamboo classes. If you install two Plugins1 plugins (A and B) in JIRA, then plugin B will be able to use the classes from plugin A as all the classes live in the same ClassLoader. However, Plugin B has no way to declare that it relies on Plugin A. If Plugin A is not installed, then this will cause ClassNotFound exceptions to occur at runtime.

Plugins2

Plugins2 plugins are not installed in the core ClassLoader. They are installed and managed by an OSGi container. This means each Plugin
Bamboo 3.1 Documentation

has its own child ClassLoader that loads the classes of that plugin. By default, plugins cannot use the classes of another plugin. However,
one plugin can explicitly export some of its packages, and then other plugins can import these dependencies. In this way, the
interdependencies can be much better managed.
In addition, a Plugins2 plugin can create its own custom extension points. Effectively, you can allow plugin points for your plugin.

Development and Installation
Configuration

The JAR file for a Plugins1 plugin looks exactly the same as one for Plugins2 with one difference in the configuration file.
The atlassian-plugin.xml for a Plugins2 plugin must declare it is Plugins2 in the <atlassian-plugin> tag.

<atlassian-plugin key="${atlassian.plugin.key}" name="${project.name}" plugins-version="2">

Installation

A Plugins1 plugin must be on the application classpath, and therefore is installed in WEB-INF/lib.
On the other hand, a Plugins2 plugin must not be on the standard classpath. They are installed in a special subfolder of the Bamboo Home
directory - <bamboo-home>/plugins/installed-plugins/.

Java packages
OSGi exports and imports dependencies based on Java packages. Only one ClassLoader can export classes from any given package.
Under Plugins2 this means it is even more important to not duplicate package names of core Bamboo classes, or other plugins.
Is there an equivalent Bamboo example
Of particular interest is the Webwork plugin module.
In JIRA v3.13 and earlier, most Plugin developers probably followed the example in the documentation that showed an Action class in the
com.atlassian.jira.web.action package.
This meant they could declare a Webwork module with a "simple" class name. This is anyway a bad idea as it allows for possible
name-space clashes. Furthermore, it will simply not work under Plugins2.
A plugin developer must create an action that lives in a unique package, and include the fully qualified class name of the action in the
configuration file.
eg:
<webwork1 key="qquserissue" name="Quick Create User Issue" class="java.lang.Object">
<actions>
<action name="com.atlassian.jira.toolkit.action.QuickCreateUserIssueAction"
alias="QuickCreateUserIssue">
<view name="createuserissue">/templates/quickcreateuser.vm</view>
</action>
</actions>
</webwork1>

Plugin Internationalisation
Do not publish this document
We do not allow users to choose locale, so there is no practical use for this feature yet.

From Bamboo 2.3 onwards, internationalisation is supported for Bamboo plugins, just as it is supported in plugins for Atlassian's JIRA and
Confluence. This means that you can allow labels and messages in your plugin to be customised via properties files. You can implement
internationalisation support for your plugin by adding the appropriate code and resources to your plugin.
For further instructions on how to internationalise your Bamboo plugin, please see this guide to plugin internationalisation. The instructions
are written for Confluence plugin development, but also apply to Bamboo plugin development.

Using Version 1 Plugins
Building Version 1 plugin using Atlassian Plugins SDK
How do I ensure that my plugin has the right version?
Ensuring valid scope of dependencies
Mark Bamboo (core) dependencies as provided
Exclude transitive dependencies that are provided by Bamboo
Configure the project to bundle dependencies with the plugin jar
Precautions
By default the Atlassian Plugin SDK generates Version 2 (OSGi) plugins. Unfortunately Bamboo currently doesn't support Version 2 (v2)
plugins for Remote Agents, meaning that plugin modules running on Agents require the Plugin to be a of Version 1 (v1). The most notable

710


example is the Source Repository Module - which most commonly is executed on Remote Agents to access the source repository.

Ensure to always test your plugin with Remote Agents to verify that there are no NoClassDefFound exceptions.

Building Version 1 plugin using Atlassian Plugins SDK

A v1 plugin is a plain old jar that is available on the Bamboo classpath. It should not contain any OSGi descriptors in its Manifest.mf. The tricky part is to make sure all dependencies get bundled with your plugin - either as separate jar or bundled with your plugin.

How do I ensure that my plugin has the right version?

See the atlassian-plugin section in Creating your Plugin Descriptor where the plugin version is declared.

Ensuring valid scope of dependencies

This also applies to v2 plugins, but is especially important for v1 plugins as they can pollute Bamboo's classpath with wrong versions of classes.

Mark Bamboo (core) dependencies as provided

It's important not to include any jars that already ship with your Bamboo server.

```xml
<dependency>
  <groupId>com.atlassian.bamboo</groupId>
  <artifactId>atlassian-bamboo-api</artifactId>
  <version>${bamboo.version}</version>
  <!-- Don't bundle API classes -->
  <scope>provided</scope>
</dependency>
```

Exclude transitive dependencies that are provided by Bamboo

atlas-mvn dependency:tree is your friend here.

```xml
<dependency>
  <groupId>org.eclipse.jgit</groupId>
  <artifactId>org.eclipse.jgit</artifactId>
  <version>0.9.3</version>
  <exclusions>
    <exclusion>
      <!-- jsch is provided by Bamboo -->
      <groupId>com.jcraft</groupId>
      <artifactId>jsch</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

Configure the project to bundle dependencies with the plugin jar

By default atlas-package from Atlassian Plugins SDK bundles dependencies for a v2 plugin layout. The jars are copied to META-INF/lib in the plugin jar which is suitable for OSGi deployment.

To make plugin v1 work the jars need to be repackaged with the plugin jar so they are visible to non-OSGi classloader.

```xml
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-bamboo-plugin</artifactId>
  <version>3.2.3</version>
  <extensions>true</extensions>
  <configuration>
    <productVersion>${bamboo.version}</productVersion>
    <productDataVersion>${bamboo.data.version}</productDataVersion>
    <!-- Version 1 compatible layout -->
    <extractDependencies>true</extractDependencies>
  </configuration>
</plugin>
```

Precautions

V1 plugins are plain old jars on the Bamboo classpath.
The main reasons to use them instead of v2 plugins are to:

- make them available to remote agents,
- provide classes to be used by the system or other v1 plugins.

There are a few issues that should be taken into account:

- Duplicate classes. It's especially difficult when multiple 3rd party plugins bundle the same classes. Duplicate classes are reported during Bamboo startup in the `atlassian-bamboo.log`. This becomes a problem when different plugins provide different versions of the jars. When adding a potentially popular dependency consider jarjar instead of unpack.
- As a special case of above, be careful not to override other plugin's (or core) classes.
- Full server restart is needed to upgrade the plugin - no `atlas-cli` possible.

**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 plugin module.
- 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.
On this page:

- General Information
- Project Service
- Plan Service
- Build Service -s 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 contain the size of whole collection (size attribute) as well as the start-index and max-result.

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><strong>Description</strong></td>
<td>Lists all the projects set up on the Bamboo server.</td>
</tr>
</tbody>
</table>
**URI Parameters**

- ?expand=projects or no parameter — List projects with projects list expanded and links to projects available.
- ?expand=projects.project — List projects with projects list expanded and project details visible.
- ?expand=projects.project.plans — List of projects and plans belonging to the project with plans expanded and links to plans available.
- ?expand=projects.project.plans.plan — List of projects and plans belonging to the project with plans expanded and plan details visible.
- ?expand=projects.project.plans.plan.actions — List of projects and plans belonging to the project with plans expanded, and plan details and actions visible.
- ?favourite&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.
- ?favourite&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.
- ?favourite&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.
- ?cloverEnabled&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.
- ?cloverEnabled&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.
- ?cloverEnabled&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.

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>
</table>
| projects/project | • name — Name of the project  
|                  | • key — Project key   |
| projects/project/link | • rel — Relationship between link and element (defaults to "self")  
|                  | • href — URL for the project |

Example of response JSON:

```json
{"expand":"projects","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/project"}}
```

**Plan Service**

<table>
<thead>
<tr>
<th>URI</th>
<th>/plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Lists all the plans set up on the Bamboo server.</td>
</tr>
<tr>
<td>URI Parameters</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>?expand=plans or no parameter — List all plans for all projects with list expanded (but no plan details).</td>
<td></td>
</tr>
<tr>
<td>?expand=plans.plan — List all plans for all projects with list expanded and plan details visible.</td>
<td></td>
</tr>
<tr>
<td>?expand=plans.plan.actions — List all plans for all projects with list expanded and plan details and plan actions visible.</td>
<td></td>
</tr>
<tr>
<td>?favourite&amp;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.</td>
<td></td>
</tr>
<tr>
<td>?favourite&amp;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.</td>
<td></td>
</tr>
<tr>
<td>?cloverEnabled&amp;expand=plans.plan — List all Clover-enabled plans ¹ plans for all projects with list expanded and plan details visible. Plans which are not favourites will not be returned.</td>
<td></td>
</tr>
<tr>
<td>?cloverEnabled&amp;expand=plans.plan.actions — List all Clover-enabled plans ¹ plans for all projects with list expanded and plan details and plan actions visible. Plans which are not favourites will not be returned.</td>
<td></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).

<table>
<thead>
<tr>
<th>HTTP Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET — Returns a list of plans.</td>
<td></td>
</tr>
</tbody>
</table>

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"/>
  <plans expand="plan" size="3" max-result="3" start-index="0">
    <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>
</plans>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>plans/plan</td>
<td></td>
</tr>
<tr>
<td>• name — Name of the plan</td>
<td></td>
</tr>
<tr>
<td>• key — Plan key</td>
<td></td>
</tr>
<tr>
<td>plans/plan/link</td>
<td></td>
</tr>
<tr>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
<td></td>
</tr>
<tr>
<td>• href — URL for the plan</td>
<td></td>
</tr>
</tbody>
</table>

Example of response JSON:

```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"}}],
  "max-result": 2,"start-index": 0,"size": 2,"expand": "plans"
}]
```

Build Service — All Builds

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/result</td>
<td>Lists the latest builds for all plans on the Bamboo server.</td>
</tr>
</tbody>
</table>
URI Parameters

- `?expand=results` or no parameter — Lists the latest build for each of the plans on the Bamboo server, in descending order (by completion time).
- `?expand=results.result` — Lists the latest build 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 builds returned).
- `?expand=results[0:N].result` — Lists the last N builds for the plans on the Bamboo server 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 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 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=results.result.artifacts` — Lists the artifacts produced by each of the latest builds for all plans on the Bamboo server.
- `?expand=results.result.labels` — Lists the labels associated with each of the latest builds for all plans on the Bamboo server.
- `?expand=results.result.comments` — Lists the comments associated with each of the latest builds for all plans on the Bamboo server.
- `?expand=results.result.jiraIssues` — Lists the JIRA issues associated with each of the latest builds for all plans on the Bamboo server.

Example of response XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <results expand="results">
- <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result"/>
- <results expand="result" size="5" max-result="5" start-index="0">
- <result number="2" state="Successful" key="MYPLAN1-2">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN1-2"/>
</result>
- <result number="5" state="Successful" key="MYPLAN2-5">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN2-5"/>
</result>
- <result number="4" state="Successful" key="MYPLAN3-4">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN3-4"/>
</result>
- <result number="5" state="Successful" key="MYPLAN4-5">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ2-MYPLAN4-5"/>
</result>
- <result number="5" state="Successful" key="MYPLAN5-5">
  <link rel="self" href="http://localhost:8085/bamboo/rest/api/latest/result/MYPROJ3-MYPLAN5-5"/>
</result>
</results>
</builds>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>results/result</td>
<td>number — Build result 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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>results/result/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>result/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>result/buildCompletedTime</td>
<td>Description: Finish time of build</td>
</tr>
<tr>
<td>result/buildDurationInSeconds</td>
<td>Description: Build duration in seconds</td>
</tr>
<tr>
<td>result/buildDurationDescription</td>
<td>Description: Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>result/buildRelativeTime</td>
<td>Description: Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>result/queueStartedTime</td>
<td>Description: Time when build was queued</td>
</tr>
<tr>
<td>result/queueTimeInSeconds</td>
<td>Description: How long build was queued</td>
</tr>
<tr>
<td>result/vcsUpdateStartedTime</td>
<td>Description: Time when VCS update occurred</td>
</tr>
<tr>
<td>result/vcsUpdateInSeconds</td>
<td>Description: How long VCS took to complete</td>
</tr>
<tr>
<td>result/buildTestSummary</td>
<td>Description: Summary of test results</td>
</tr>
<tr>
<td>result/successfulTestCount</td>
<td>Description: Number of successful tests in build</td>
</tr>
<tr>
<td>result/failedTestCount</td>
<td>Description: Number of failed tests in build</td>
</tr>
<tr>
<td>result/buildReason</td>
<td>Description: Reason for build</td>
</tr>
<tr>
<td>results/result/artifacts/artifact/name</td>
<td>Description: Name of the build artifact</td>
</tr>
<tr>
<td>results/result/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>results/result/labels/label/name</td>
<td>Description: Name of the label</td>
</tr>
<tr>
<td>results/result/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/content</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/creationDate</td>
<td>Description: Date that comment was created</td>
</tr>
<tr>
<td>results/result/comments/comment/modificationDate</td>
<td>Description: Date that comment was last updated</td>
</tr>
<tr>
<td>results/result/jiraIssues/issue</td>
<td>• key — Key of the JIRA issue</td>
</tr>
</tbody>
</table>

Example of response JSON:

```json
{"expand":"results","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/result"},
"builds":null}
```

**Build Service — Specific Project**

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/result/PROJECTKEY</td>
<td>Lists the latest builds for all plans for a specific project on the Bamboo server.</td>
</tr>
</tbody>
</table>

(where PROJECTKEY is the key of the desired project, e.g. MYPROJ1)
**URI Parameters**

- **?expand=results** 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=results.result** — 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=results[0:N].result** — 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 5 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).

**Example of response XML:**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<results expand="results">
  - <results expand="result" size="3" max-result="3" start-index="0">
    - <result number="2" state="Successful" key="MYPROJ1-MYPLAN1-2">
      <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN1-2"/>
    </result>
    - <result number="5" state="Successful" key="MYPROJ1-MYPLAN2-5">
      <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN2-5"/>
    </result>
    - <result number="4" state="Successful" key="MYPROJ1-MYPLAN3-4">
      <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN3-4"/>
    </result>
  </results>
</results>
```

**Elements in the response:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| results/result           | - number — Build number
|                          | - state — State of the build (e.g. Successful, Failed)
|                          | - key — Key of the build result |
| results/result/link      | - rel — Relationship between link and element (defaults to "self")
<p>|                          | - href — URL for the build result |
| result/buildStartedTime  | Description: Start time of build |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result/buildCompletedTime</td>
<td>Finish time of build</td>
</tr>
<tr>
<td>result/buildDurationInSeconds</td>
<td>Build duration in seconds</td>
</tr>
<tr>
<td>result/buildDurationDescription</td>
<td>Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>result/buildRelativeTime</td>
<td>Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>result/queueStartedTime</td>
<td>Time when build was queued</td>
</tr>
<tr>
<td>result/queueTimeInSeconds</td>
<td>How long build was queued</td>
</tr>
<tr>
<td>result/vcsUpdateStartedTime</td>
<td>Time when VCS update occurred</td>
</tr>
<tr>
<td>result/vcsUpdateInSeconds</td>
<td>How long VCS took to complete</td>
</tr>
<tr>
<td>result/buildTestSummary</td>
<td>Summary of test results</td>
</tr>
<tr>
<td>result/successfulTestCount</td>
<td>Number of successful tests in build</td>
</tr>
<tr>
<td>result/buildReason</td>
<td>Reason for build</td>
</tr>
<tr>
<td>results/build/artifacts/artifact/name</td>
<td>Name of the build artifact</td>
</tr>
<tr>
<td>results/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>results/result/labels/label/name</td>
<td>Name of the label</td>
</tr>
<tr>
<td>results/result/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/content</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/creationDate</td>
<td>Description: Date that comment was created</td>
</tr>
<tr>
<td>results/result/comments/comment/modificationDate</td>
<td>Description: Date that comment was last updated</td>
</tr>
<tr>
<td>results/result/jiraIssues/issue</td>
<td>• key — Key of the JIRA issue</td>
</tr>
<tr>
<td>results/result/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":"results","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/result"},
"results":{...}}
```

**Build Service — Specific Plan**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>result/PROJECTKEY-PLANKEY (where PROJECTKEY-PLANKEY is the key of the desired build plan of a project, e.g. MYPROJ1-MYPLAN1)</td>
</tr>
<tr>
<td>Description</td>
<td>Provides information about builds executed for a build plan from latest to the first build</td>
</tr>
</tbody>
</table>
**URI Parameters**

- ?expand=results or no parameter — List all build results for the build plan (build details not visible).
- ?expand=results.result — 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).*
- ?expand=results[0:N].result — List the last N build results for the build plan with build details expanded, in descending order.

**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.

- ?expand=results.result.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).*
- ?expand=plan — Displays the build plan information (expanded), including a summary of build results information.
- ?expand=plan.actions — Displays the build plan information (expanded), including a summary of build results information and available actions.
- ?expand=results.result.artifacts — Lists the artifacts produced by each of the latest builds for the specified plan.
- ?expand=results.result.labels — Lists the labels associated with each of the latest builds for the specified plan.
- ?expand=results.result.comments — Lists the comments associated with each of the latest builds for the specified plan.
- ?expand=results.result.jiraIssues — Lists the JIRA issues associated with each of the latest builds for the specified plan.

**HTTP Methods**

- GET — Returns a list of plans.

Example of response XML:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<results expand="results">
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result"/>
  <results expand="result" size="2" max-result="2" start-index="0">
    <result number="1" state="Successful" key="MYPROJ1-MYPLAN6-1">
      <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN6-1"/>
    </result>
    <result number="2" state="Successful" key="MYPROJ1-MYPLAN6-2">
      <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/result/MYPROJ1-MYPLAN6-2"/>
    </result>
  </results>
</results>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>results/result</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>results/result/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>result/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>result/buildCompletedTime</td>
<td>Description: Finish time of build</td>
</tr>
<tr>
<td>result/buildDurationInSeconds</td>
<td>Description: Build duration in seconds</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>result/buildDurationDescription</td>
<td>Description: Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>result/buildRelativeTime</td>
<td>Description: Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>result/queueStartedTime</td>
<td>Description: Time when build was queued</td>
</tr>
<tr>
<td>result/queueTimeInSeconds</td>
<td>Description: How long build was queued</td>
</tr>
<tr>
<td>result/vcsUpdateStartedTime</td>
<td>Description: Time when VCS update occurred</td>
</tr>
<tr>
<td>result/vcsUpdateInSeconds</td>
<td>Description: How long VCS took to complete</td>
</tr>
<tr>
<td>result/buildTestSummary</td>
<td>Description: Summary of test results</td>
</tr>
<tr>
<td>result/successfulTestCount</td>
<td>Description: Number of successful tests in build</td>
</tr>
<tr>
<td>result/failedTestCount</td>
<td>Description: Number of failed tests in build</td>
</tr>
<tr>
<td>result/buildReason</td>
<td>Description: Reason for build</td>
</tr>
<tr>
<td>results/result/artifacts/artifact/name</td>
<td>Description: Name of the build artifact</td>
</tr>
<tr>
<td>results/result/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>results/result/labels/label/name</td>
<td>Description: Name of the label</td>
</tr>
<tr>
<td>results/result/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/content</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/creationDate</td>
<td>Description: Date that comment was created</td>
</tr>
<tr>
<td>results/result/comments/comment/modificationDate</td>
<td>Description: Date that comment was last updated</td>
</tr>
<tr>
<td>results/result/jiraIssues/issue</td>
<td>• key — Key of the JIRA issue</td>
</tr>
<tr>
<td>results/result/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":"results","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/result"},
"results":{
    "expand":...}}
```

**Build Service — Specific Build Result**

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/result/PROJECTKEY-PLANKEY/BUILDNUMBER</td>
<td>Provides information about a single build executed for a build plan (build result information only)</td>
</tr>
</tbody>
</table>

(where PROJECTKEY-PLANKEY is the key of the desired build plan of a project, e.g. MYPROJ1-MYPLAN1, and BUILDNUMBER is the number of the build, e.g. 1, latest, etc)
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=changes — Displays the latest build result for the plan and list of changes for build.
- ?expand=results.result.artifacts — Lists the artifacts for the specified build result.
- ?expand=results.result.labels — Lists the labels for the specified build result.
- ?expand=results.result.comments — Lists the comments for the specified build result.
- ?expand=results.result.jiraIssues — Lists the JIRA issues for the specified build result.
- ?expand=changes — Lists the changsets for the specified build result.
- ?expand=changes.change — Lists the changsets (changelog details expanded) for the specified build result.
- ?expand=changes.change.files — Lists the changsets (changelog 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 tests (test result details expanded) for the specified build result.
- ?expand=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" ?>
- <result 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>
  <queueDurationInSeconds>1.0</queueDurationInSeconds>
  <vcsUpdateStartedTime>2009-06-15T14:53:55.546+1000</vcsUpdateStartedTime>
  <vcsUpdateDurationInSeconds>131.0</vcsUpdateDurationInSeconds>
  <artifacts size="1" max-result="1" start-index="0" />
  <link rel="self" href="http://myhost:8085/bamboo/rest/api/latest/build/MYPLAN1-5157" />
</result>
```

Elements in the response:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</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>result/buildStartedTime</td>
<td>Description: Start time of build</td>
</tr>
<tr>
<td>result/buildCompletedTime</td>
<td>Description: Finish time of build</td>
</tr>
<tr>
<td>result/buildDurationInSeconds</td>
<td>Description: Build duration in seconds</td>
</tr>
<tr>
<td>result/buildDurationDescription</td>
<td>Description: Description of build duration (conversion of seconds to more usable units of time)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>result/buildRelativeTime</td>
<td>Time when build was run (relative to current time)</td>
</tr>
<tr>
<td>result/queueStartedTime</td>
<td>Time when build was queued</td>
</tr>
<tr>
<td>result/queueTimeInSeconds</td>
<td>How long build was queued</td>
</tr>
<tr>
<td>result/vcsUpdateStartedTime</td>
<td>Time when VCS update occurred</td>
</tr>
<tr>
<td>result/vcsUpdateInSeconds</td>
<td>How long VCS took to complete</td>
</tr>
<tr>
<td>result/buildTestSummary</td>
<td>Summary of test results</td>
</tr>
<tr>
<td>result/successfulTestCount</td>
<td>Number of successful tests in build</td>
</tr>
<tr>
<td>result/failedTestCount</td>
<td>Number of failed tests in build</td>
</tr>
<tr>
<td>result/buildReason</td>
<td>Reason for build</td>
</tr>
<tr>
<td>results/result/artifacts/artifact/name</td>
<td>Name of the build artifact</td>
</tr>
<tr>
<td>results/result/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>results/result/labels/label/name</td>
<td>Name of the label</td>
</tr>
<tr>
<td>results/result/comments/comment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/content</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- author — Author of the comment</td>
</tr>
<tr>
<td>results/result/comments/comment/creationDate</td>
<td>Date that comment was created</td>
</tr>
<tr>
<td>results/result/comments/comment/modificationDate</td>
<td>Date that comment was last updated</td>
</tr>
<tr>
<td>results/result/jiraIssues/issue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- key — Key of the JIRA issue</td>
</tr>
<tr>
<td>results/result/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>results/result/changes/change</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- changeset.id — Id of the changeset</td>
</tr>
<tr>
<td></td>
<td>- author — Author of the changeset</td>
</tr>
<tr>
<td>results/result/changes/change/comment</td>
<td>Description: Comment associated with the changeset</td>
</tr>
<tr>
<td>results/result/changes/change/date</td>
<td>Description: Date of the changeset</td>
</tr>
<tr>
<td>results/result/changes/change/files/file/name</td>
<td>Description: Name of the file associated with the changeset</td>
</tr>
<tr>
<td>results/result/changes/change/files/file/revision</td>
<td>Description: Revision of the file associated with the changeset</td>
</tr>
<tr>
<td>results/result/testResults/successful</td>
<td>- size — Number of successful tests</td>
</tr>
<tr>
<td>results/result/testResults/failed</td>
<td>- size — Number of failed tests</td>
</tr>
<tr>
<td>key</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| results/result/testResults/failed/testResult | status — status of test, i.e. “failed”  
 methodName — Method associated with failed test  
 className — Class associated with failed test |
| results/result/testResults/failed/testResult/duration | Duration of test                                                                          |
| results/result/testResults/failed/testResult/durationInSeconds | Duration of test in seconds                                                                |
| results/result/testResults/failed/testResult/errors | size — Number of errors in test                                                            |
| results/result/testResults/failed/testResult/errors/error/message | Error message for error in failed test.                                                     |

Example of response JSON:

```json
```

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>

<table>
<thead>
<tr>
<th>URI Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>reportKey — report class as returned by /chart/reports resource in key tag</td>
<td></td>
</tr>
<tr>
<td>buildKeys — comma-separated list of build keys (PROJECTKEY-PLANKEY) as returned by /chart/reports resource</td>
<td></td>
</tr>
<tr>
<td>groupByPeriod — period to group results by in the report chart. Valid values = AUTO, DAY, WEEK, MONTH</td>
<td></td>
</tr>
<tr>
<td>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)</td>
<td></td>
</tr>
<tr>
<td>dateFrom (optional) — start date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE</td>
<td></td>
</tr>
<tr>
<td>dateTo (optional) — start date for report chart as dd/MM/yyyy, (e.g. 15/6/2009), mandatory if dateFilter=RANGE</td>
<td></td>
</tr>
<tr>
<td>width (optional) — width of the chart in pixels</td>
<td></td>
</tr>
<tr>
<td>height (optional) — height of the chart in pixels</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GET — Returns a list of reports.</td>
<td></td>
</tr>
</tbody>
</table>

Example of response XML (list all available reports):
URI | /chart/reports
---|---
Description | 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>
<tbody>
<tr>
<td>report</td>
<td></td>
</tr>
</tbody>
</table>
  - key — key of the Bamboo report  
  - name — name of the Bamboo report |

Example of response JSON (list all available reports):

```json
```
Build Queue Service

<table>
<thead>
<tr>
<th>URI</th>
<th>/queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</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>URI Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>no parameter — Displays the number of builds waiting in the build queue.</td>
</tr>
<tr>
<td>?expand=queuedBuilds — Lists all the builds waiting in the build queue.</td>
</tr>
<tr>
<td>?expand=queuedBuilds.queuedBuild.changes — Lists all the builds waiting in the build queue with associated changes expanded.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET — Returns a list of projects.</td>
</tr>
<tr>
<td>POST — Adds a build to the build queue (e.g. /queue/MYPROJ1-MYPLAN1). No POST data required. Response will be empty.</td>
</tr>
<tr>
<td>DELETE — Removes a build from the build queue (e.g. /queue/MYPROJ1-MYPLAN1-BUILDNUMBER). Response will be empty.</td>
</tr>
</tbody>
</table>

Example command to trigger a build or delete it from the queue:

curl -X POST --user admin:admin http://host:8085/bamboo/rest/api/latest/queue/PLAN-KEY?os_authType=basic
curl -X DELETE --user admin:admin http://host:8085/bamboo/rest/api/latest/queue/PLAN-KEY?os_authType=basic

Example of response XML (retrieving builds in the build queue):

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <queue 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</triggerReason>
      - <changes expand="change" size="1" max-result="1" start-index="0">
        <change changesetId="98662" author="apowers"/>
      </changes>
    </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>
</tbody>
</table>
queue/queuedBuilds/queuedBuild

- **buildResultKey** — Key for the build result
- **buildNumber** — Build number
- **planKey** — Key for the build plan

queue/queuedBuilds/queuedBuild/triggerReason

Description: Build trigger reason

queue/queuedBuilds/queuedBuild/changes

- **size** — Number of changes associated with build

queue/queuedBuilds/queuedBuild/changes/change

- **changesetId** — Id of the changeset
- **author** — Author of the changeset

Example of response JSON:

```json
{"expand":"queuedBuilds","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/queue"},
"queuedBuilds":{ ...

"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

**URI**

/build/PROJECTKEY-PLANKEY-BUILDNUMBER/comment

**Description**

Retrieves the comments for a build result, adds a comment to a build result.

**URI Parameters**

- no parameter — Retrieves the comments for a build result.
- ?expand=comments.comment — Retrieves the comments for a build result with expanded comment details.

**HTTP Methods**

- GET — Retrieves the comments for a build result.
- POST — Adds a comment to the build result. Response will be empty.

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>
      <creationDate>2009-09-11T22:35:34.436+0200</creationDate>
      <modificationDate>2009-09-11T22:35:34.436+0200</modificationDate>
    </comment>
  </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")
<p>|                | • href — URL for the comment    |
| comments/comments | • size — Number of comments for the build result |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>comments/comments/comment</td>
<td>• author — Author of the comment</td>
</tr>
<tr>
<td>comments/comments/comment/content</td>
<td>Description: Content of the comment</td>
</tr>
<tr>
<td>comments/comments/comment/creationDate</td>
<td>Description: Creation date of the comment</td>
</tr>
<tr>
<td>comments/comments/comment/modificationDate</td>
<td>Description: Last modified date of the comment</td>
</tr>
</tbody>
</table>

**Example of response JSON:**

```json
{"expand":"comments","link":{"rel":"self","href":"http://myhost:8085/bamboo/rest/api/latest/build/MYP
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):**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <comment author="apowers">
  <content>A build result comment.</content>
</comment>
```

**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**

<table>
<thead>
<tr>
<th>URI</th>
<th>/build/PROJECTKEY-PLANKEY-BUILDNUMBER/label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Retrieves the labels for a build result, adds or removes a label to a build result.</td>
</tr>
<tr>
<td>URI Parameters</td>
<td>• no parameter — Retrieves the labels for a build result.</td>
</tr>
<tr>
<td>HTTP Methods</td>
<td>• GET — Retrieves the labels for a build result.</td>
</tr>
<tr>
<td></td>
<td>• POST — Adds a label to the build result. Response will be empty.</td>
</tr>
<tr>
<td></td>
<td>• DELETE — Removes a label from the build result, build/PROJECTKEY-PLANKEY-BUILDNUMBER/label/labelname e.g. Response will be empty.</td>
</tr>
</tbody>
</table>

**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>
<tbody>
<tr>
<td>labels/link</td>
<td>• rel — Relationship between link and element (defaults to &quot;self&quot;)</td>
</tr>
<tr>
<td></td>
<td>• href — URL for the label</td>
</tr>
</tbody>
</table>
Example of response 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 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

**URI**
/`info`

**Description**
Retrieves information about a Bamboo server.

**URI Parameters**
- no parameter — Retrieves information about a Bamboo server.

**HTTP Methods**
- GET — Retrieves information about a Bamboo server.

Example of response XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <info>
  <version>2.4</version>
  <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:

```
{"version":"2.4","edition":null,"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.
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`. The URL must also contain the `os_authType=basic` query parameter.

**Example:**

```
wget --http-user=admin --http-password=admin http://localhost:8085/rest/api/latest/plan?os_authType=basic
```
```
curl --user admin:admin http://localhost:8085/rest/api/latest/plan?os_authType=basic
```

Or in a browser use `http://host:8085/rest/api/latest?os_authType=basic&os_username=<user>&os_password=<pw>`

REST Resources and URI Structure

URIs 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:

```
http://myhost.com:8085/bamboo/rest/api/1/
```

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/latest/...
* 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>
</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.

Authentication
Bamboo requires authentication to access the remote API. All methods accept an authentication token parameter.

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
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**

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:

```xml
<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:

```xml
<response>
  <build>
    <name>Confluence Stable - LDAP User Management</name>
    <key>CONFSTABFUNC-LDAP</key>
    <state>success</state>
    <buildNumber>146</buildNumber>
    <failedTestCount>0</failedTestCount>
    <successfulTestCount>4</successfulTestCount>
    <buildTime>2008-02-12 23:21:41</buildTime>
  </build>
  <build>
    <name>Confluence Stable - Main Build</name>
    <key>CONFSTAB-MAIN</key>
    <state>failed</state>
    <buildNumber>10</buildNumber>
    <failedTestCount>3</failedTestCount>
    <successfulTestCount>10</successfulTestCount>
    <buildTime>2008-02-12 12:02:14</buildTime>
  </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>buildDurationInSeconds</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 resultys</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>
<buildCompletedDate>2009-02-24 16:39:46</buildCompletedDate>
<buildDurationInSeconds>1136</buildDurationInSeconds>
<buildDurationDescription>18 minutes</buildDurationDescription>
<buildRelativeBuildDate>42 minutes ago</buildRelativeBuildDate>
<buildTestSummary>1497 passed</buildTestSummary>
<buildReason>Code has changed</buildReason>
<commit>
  <commit author="bob"/>
</commit>
```

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:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
</table>
**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>
</table>

URL:

```
/api/rest/getBambooBuildNumber.action
```

**Successful Response:**

```
<response>
  <bambooBuildNumber>607</bambooBuildNumber>
</response>
```
auth | authentication token for the session to log out from

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-ee5c287</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>
<tr>
<td>state</td>
<td>The state of the elastic instance</td>
<td>STARTING (instance pending) / IDENTIFIED (instance pending with identification details available) / RUNNING / SHUTTING_DOWN / TERMINATED / FAILED_TO_START (error state) / UNKNOWN (error state)</td>
</tr>
<tr>
<td>uptime</td>
<td>The length of time the instances has been running (according to AWS)</td>
<td>Duration in milliseconds</td>
</tr>
</tbody>
</table>

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><em>(optional)</em> 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:**

<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:**

---
Successful Response:

```xml
<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

```xml
<response/>
```

Add Label To Build Result

Adds 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&content=my%20comment
```

Successful Response

```xml
<response/>
```
Example:

/api/rest/addLabelToBuildResults.action?auth=YOwcnQHh5q&buildKey=MYPROJECT-TRUNK&buildNumber=109&label=myLabel

Successful Response

```xml
<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

```xml
<response vcsRevisionKey="75149">
  <commits>
    <commit author="bob" date="2009-02-18T17:22:38+1100">
      <comment>
        Fixing annoying bug with Command Executer
      </comment>
      <files>
        <file name="/trunk/components/bamboo-core/src/main/java/com/atlassian/bamboo/command/Command.java" revision="75149"/>
      </files>
    </commit>
  </commits>
  <failedTests>
    <testResult testClass="com.atlassian.bamboo.builder.coverage.CloverReportParserTest" testMethod="testExtractsProjectCoveragePercentage" duration="0.195">
      <errors>
        <error>
          org.dom4j.DocumentException: null Nested exception: null
          at org.dom4j.io.SAXReader.read(SAXReader.java:358)
          at org.dom4j.io.SAXReader.read(SAXReader.java:261)
        </error>
      </errors>
    </testResult>
  </failedTests>
  <successfulTests>
    <testResult testClass="com.atlassian.bamboo.rest.GetRecentlyCompletedBuildResultsForBuildTest" testMethod="testExecutesNormally" duration="0.002"/>
    <testResult testClass="com.atlassian.bamboo.util.HtmlUtilsTest" testMethod="testPreformattedTextMultiplespaces" duration="0.002"/>
  </successfulTests>
</response>
```
<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 build</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>
<tr>
<td>errors</td>
<td>List of errors generated by the test</td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>Error output of an individual error</td>
<td>Freeform content.</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 CVS/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 plan you want to trigger</td>
</tr>
<tr>
<td></td>
<td>example BAM-TRUNK, where BAM = project key, TRUNK = plan key</td>
</tr>
</tbody>
</table>

**URL:**

/api/rest/updateAndBuild.action

**Successful Response:**

---

744
 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:

```xml
<response>
  <string>OK a build has been triggered.</string>
</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:

```xml
<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=m3OaZ14Ck2&projectKey=MY_PROJECT

Successful Response:

```
<response>
  <project>
    <name>${project.name}</name>
    <key>${project.key}</key>
    <currentStatus>${project.currentStatus}</currentStatus>
  </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>
<tr>
<td>currentStatus</td>
<td>Provides an overall status of the project</td>
<td>&quot;current&quot; if any builds are building &quot;fail&quot; if any builds are failing, otherwise &quot;success&quot;</td>
</tr>
</tbody>
</table>

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>

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>Represents an individual build plan</td>
<td></td>
</tr>
<tr>
<td>enabled</td>
<td>Whether or not the build plan is enabled (as opposed to disabled)</td>
<td>true/false</td>
</tr>
<tr>
<td>name</td>
<td>Full plan name</td>
<td>User friendly name that appears in Bamboo (includes project name)</td>
</tr>
<tr>
<td>key</td>
<td>Full plan Key</td>
<td>Build plan identifier in the form &lt;projectkey&gt;-&lt;planKey&gt;</td>
</tr>
</tbody>
</table>

**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](#).

If you have a question, please ask it as a comment and someone from Atlassian will reply. Comment threads will gradually be merged back into this FAQ as needed. Please try to be as specific as possible with your questions.

Questions

Page: [How do I search for previous build result?](#)

Page: [Note to Atlassian staff on the Bamboo Developer FAQ](#)

Page: [How do I inject managers into my plugin?](#)

**How do I inject managers into my plugin?**

*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?**

*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 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):

```java
buildExecutionManager.tryToDetectAndBuild(buildKey, new
BuildExecutionManager.BuildDetectionAction()
{
    publicBuildContextprocess()
    {
        try
        {
            BuildLoggerbuildLogger = buildPlan.getBuildLogger();
            BuildDefinitionbuildDefinition = buildPlan.getBuildDefinition();
            Repositoryrepository = buildDefinition.getRepository();

            // Some feedback to the UI for what's going down
            buildLogger.addBuildLogEntry("Manual build triggered by " + user);

            // This block only needed if you care about changes
            String lastVcsRevisionKey = buildPlan.getLastVcsRevisionKey();
            BuildChangesbuildChanges;
            if (lastVcsRevisionKey != null)
            {
                buildChanges =
                changeDetectionManager.collectChangesSinceLastBuild(buildPlan.getKey(), repository,
                lastVcsRevisionKey);
            }
            else
            {
                buildChanges = new BuildChangesImpl();
            }

            // A trigger reason
            TriggerReasontriggerReason =
            triggerReasonManager.getTriggerReason(ManualBuildTriggerReason.KEY, buildChanges,
            EasyMap.build(ManualBuildTriggerReason.TRIGGER_MANUAL_USER, user.getName()));

            BuildContextBuildContext = new BuildContextImpl(buildPlan,
            buildManager.retrieveNextBuildNumber(buildPlan.getKey()), // needed to get the correct build
            result number
            triggerReason,
            buildDefinition,
            buildChanges);

            return buildContext;
        }
        catch (Exceptione)
        {
            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);
```

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)
```

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)

```java
buildExecutionManager.tryToDetectAndBuild(buildKey, new
BuildExecutionManager.BuildDetectionAction()
{
    public BuildContext process()
    {
        try
        {
            BuildLogger buildLogger = buildPlan.getBuildLogger();
            BuildDefinition buildDefinition = buildPlan.getBuildDefinition();
            Repository repository = buildDefinition.getRepository();

            buildLogger.addBuildLogEntry("Manual build triggered by " + user); // logging to the
            live activity log

            // This block is only required if you care about the changes
            String lastVcsRevisionKey = buildPlan.getLastVcsRevisionKey();
            BuildChanges buildChanges;
            if (lastVcsRevisionKey != null)
            {
                buildChanges = changeDetectionManager.collectChangesSinceLastBuild(buildPlan.getKey(), repository,
                lastVcsRevisionKey);
            }
            else
            {
                buildChanges = new BuildChangesImpl();
            }

            // Generate a trigger reason
            TriggerReason triggerReason =
            triggerReasonManager.getTriggerReason(ManualBuildTriggerReason.KEY, buildChanges,
            EasyMap.build(ManualBuildTriggerReason.TRIGGER_MANUAL_USER, user.getName()));

           BuildContext buildContext = new BuildContextImpl(buildPlan,
            buildManager.retrieveNextBuildNumber(buildPlan.getKey()), // you need this to ensure build numbers
            are unique
            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);
```

Note to Atlassian staff on the Bamboo Developer FAQ

⚠️ This page is only visible to atlassian-staff.

Just add your FAQ as a child page to the Bamboo Developer FAQ page with a 'faq_bamboo_dev' label and it will display in the list of questions.

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
    - User Interface Plugin Modules
      - Additional Build Configuration Module
      - Report Module
      - Web Item Module
      - Web Panel Module
      - Web Repository Viewer Module
      - Web Section Module
      - XWork Plugin Module
    - Build Lifecycle Plugin Modules
      - Build Agent Requirement Filter Module
      - Build Complete Action Module
      - Builder Plugin Module
      - Build Processor Module
      - Build Processor Server Module
      - Build Trigger Condition Module
      - Command Decorator Module
      - Post-Build Completed Action Module
      - Post-Chain Action Module
      - Post-Job Action Module
      - Post-Stage Action Module
      - Pre-Build Action Module
      - Pre-Build Queued Action
      - Pre-Chain Action Module
      - Pre-Job Action Module
      - Pre-Stage Action Module
      - Source Repository Module
      - Task Plugin Module
        - LegacyBuilderToTaskConverter Plugin Module
        - Tasks Overview
        - Trigger Reason Module
    - Notification Plugin Modules
      - Notification Condition Module
      - Notification Recipient Module
      - Notification Type Module
        - Building a Notification Plugin
    - System Plugin Modules
      - Capability Type Module
      - Component Module
      - Index Reader Module
      - Plan Deletion Interceptor Action Module
      - Post Build Index Writer Module
      - Post Chain Index Writer Module
      - Servlet Context Listener Plugin Module
      - Servlet Context Parameter Plugin Module
      - Servlet Filter Plugin Module
      - Servlet Plugin Module
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](#)

**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:
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](#).

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.
public class BuildLabeller implements CustomBuildCompleteAction {
    private static final Logger log = Logger.getLogger(BuildLabeller.class);
    
    /**
    * Dependency on labelManager. Bamboo's Spring IOC will automatically inject manager
    * into this class via the setter.
    */
    private LabelManager labelManager;

    /***
    * This action will run after a build has completed.
    * The build will be labelled with "out_of_memory" if the "OutOfMemoryError" was detected in the
    * logs.
    *
    * @param build
    * @param buildResults
    */
    public void run(Build build, BuildResults buildResults) {
        List logs = build.getBuildLogger().getBuildLog();
        for (Iterator iterator = logs.iterator(); iterator.hasNext();)
            {SimpleLogEntry log = (SimpleLogEntry) iterator.next();
                if(log.getLog().indexOf("OutOfMemoryError") != -1)
                    {getLabelManager().addLabel("out_of_memory", buildResults, null);
                        break;
                    }
            }
    }

    /***
    * This method is used to validate a build configuration for a build plan
    * This is used if the CustomBuildCompleteAction needs to have configuration stored
    * against the build plan.
    *
    * @param buildConfiguration
    * @return
    */
    public ErrorCollection validate(BuildConfiguration buildConfiguration) {
        return null;
    }
}

getters & setters
public LabelManager getLabelManager() {
    return labelManager;
}
public void setLabelManager(LabelManager labelManager) {
    this.labelManager = labelManager;
}

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`).

```xml
<buildCompleteAction key="labeller" name="Build Labeller"
    class="com.atlassian.bamboo.plugins.labeller.BuildLabeller">
    <description>An automatic labelling plugin.</description>
</buildCompleteAction>
```

**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 `bamboo-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 Build Result](image)

**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
  - User Interface Plugin Modules
    - Additional Build Configuration Module
    - Report Module
    - Web Item Module
    - Web Panel Module
    - Web Repository Viewer Module
    - Web Section Module
    - XWork Plugin Module
  - Build Lifecycle Plugin Modules
    - Build Agent Requirement Filter Module
    - Build Complete Action Module
    - Builder Plugin Module
    - Build Processor Module
    - Build Processor Server Module
    - Build Trigger Condition Module
    - Command Decorator Module
    - Post-Build Completed Action Module
    - Post-Chain Action Module
    - Post-Job Action Module
    - Post-Stage Action Module
    - Pre-Build Action Module
    - Pre-Build Queued Action
    - Pre-Chain Action Module
    - Pre-Job Action Module
    - Pre-Stage Action Module
    - Source Repository Module
    - Task Plugin Module
      - LegacyBuilderToTaskConverter Plugin Module
      - Trigger Reason Module
    - Notification Plugin Modules
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
**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
[#if build.buildDefinition.customConfiguration.get('custom.bamboo.labeller.regex')?has_content ]
[@ui.bambooInfoDisplay titleKey='Pattern Matching Labelling' float=false height='80px']
  [@ww.label label='Regex Pattern' ]
  [@ww.param name='value']${build.buildDefinition.customConfiguration.get('custom.bamboo.labeller.regex')?if_exists}[/@ww.param]
[/@ww.label]
  [@ww.label label='Labels' ]
  [@ww.param name='value']${build.buildDefinition.customConfiguration.get('custom.bamboo.labeller.label')?if_exists}[/@ww.param]
[/@ww.label]
[/@ui.bambooInfoDisplay]
[/#if]
```

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:
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.

```java
/**
 * This method is used to validate a build configuration for a build plan
 * @param buildConfiguration
 * @return
 */
public ErrorCollection validate(BuildConfiguration buildConfiguration) {
    // Check the label values to see if they have any invalid characters
    ErrorCollection errors = new SimpleErrorCollection();
    String labelInput = buildConfiguration.getString("custom.bamboo.labeller.label");
    List labels = LabelParser.split(labelInput);
    for (Iterator iterator = labels.iterator(); iterator.hasNext();)
    {
        String label = (String) iterator.next();
        boolean validLabel = LabelParser.isValidLabelName(label);
        if (!validLabel)
        {
            errors.addError("custom.bamboo.labeller.label", label + " contains invalid characters *
        }
    }
    // See if the regex is a valid one by trying to compile it
    String regex = buildConfiguration.getString("custom.bamboo.labeller.regex");
    try
    {
        Pattern.compile(regex);
    } catch(PatternSyntaxException e)
    {
        errors.addError("custom.bamboo.labeller.regex", regex + " is not a valid regex pattern.");
    }
    return errors;
}
```

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);
                    }
                    break;
                }
            }
        }
    }
}

So that’s it! We have now completed a Bamboo plugin containing one BuildCompleteAction module which will match the output logs against a regular expression, and tag it with a set of label(s).

RELATED TOPICS
- Bamboo Plugin Guide
  - Bamboo Plugin Module Types
    - User Interface Plugin Modules
      - Additional Build Configuration Module
      - Report Module
      - Web Item Module
      - Web Panel Module
      - Web Repository Viewer Module
      - Web Section Module
      - XWork Plugin Module
    - Build Lifecycle Plugin Modules
      - Build Agent Requirement Filter Module
      - Build Complete Action Module
      - Builder Plugin Module
- Build Processor Module
- Build Processor Server Module
- Build Trigger Condition Module
- Command Decorator Module
- Post-Build Completed Action Module
- Post-Chain Action Module
- Post-Job Action Module
- Post-Stage Action Module
- Pre-Build Action Module
- Pre-Build Queued Action
- Pre-Chain Action Module
- Pre-Job Action Module
- Pre-Stage Action Module
- Source Repository Module
- Task Plugin Module
  - LegacyBuilderToTaskConverter Plugin Module
  - Tasks Overview
- Trigger Reason Module
- Notification Plugin Modules
  - Notification Condition Module
  - Notification Recipient Module
  - Notification Type Module
    - Building a Notification Plugin
  - Building a Notification Plugin - copy for Bamboo 2.7
- System Plugin Modules
  - Capability Type Module
  - Component Module
  - Index Reader Module
  - Plan Deletion Interceptor Action Module
  - Post Build Index Writer Module
  - Post Chain Index Writer Module
  - Servlet Context Listener Plugin Module
  - Servlet Context Parameter Plugin Module
  - Servlet Filter Plugin Module
  - Servlet Plugin Module
- Bamboo Event Listeners
  - Bamboo's Build Process
  - Accessing Bamboo Components From Plugin Modules
  - Common Bamboo Classes
  - Bamboo Persistence using Bandana
  - Downloadable Plugin Resources
  - Web Resources
  - Differences between Plugins1 and Plugins2
  - Plugin Internationalisation
  - Using Version 1 Plugins
- 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 programmatically?
  - How do I trigger off a build from my action?
  - Note to Atlassian staff on the Bamboo Developer FAQ
- Bamboo Plugin Tutorial
  - Tutorial 1 - Getting Started with a Simple Post Build Labeller
  - Tutorial 2 - Configurable Regex Labeller
- Changes by Version
  - Changes for Bamboo 3.1
  - Changes for Bamboo 3.0
  - Changes for Bamboo 2.7
  - Changes for Bamboo 2.6
  - Changes for Bamboo 2.3
  - Changes for Bamboo 2.2
  - Changes for Bamboo 2.1
  - Changes for Bamboo 2.0
    - Changes for Bamboo 2.1.5
  - Changes for 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
    - Post-Build Processing in Bamboo 2.0
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 3.1
- Changes for Bamboo 3.0
- Changes for Bamboo 2.7
- Changes for Bamboo 2.6
- Changes for Bamboo 2.3
- Changes for Bamboo 2.2
- Changes for Bamboo 2.1
- Changes for Bamboo 2.0
- Changes for Bamboo 2.4
- Changes for Bamboo 2.5

Changes for Bamboo 3.1

- General Notes
- Deprecated Classes
- Deprecated Methods
- New Integration Points
  - Tasks
  - Label Events
  - Managing caches
  - Log interceptors
  - Gravatar Support

General Notes

In 3.1 we have introduced Tasks. Each Job can now have multiple Tasks which run in order on a single Agent. If a Task fails the following tasks will not execute, with the exception of Final Tasks which will always execute. Tasks are pluginable and have replaced Bamboo’s Builders. We have included a Compatability Task which will run old Builder configurations however, we encourage you to update Builders as soon as possible.

We've made writing new Task plugins incredibly easy. You can find all the information on creating Tasks on our Tasks Overview page. Also on that page you can find information regarding the new way we handle Requirements and Capabilities.

Another feature in 3.1 is Parameterised Builds, providing the ability to override variables at the run time of a build. With this we have moved Global Variable storage into the database and introduced the concept of Plan Variables.

Deprecated Classes

<table>
<thead>
<tr>
<th>Deprecated</th>
<th>Replacement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builder &amp; AbstractBuilder</td>
<td>TaskType</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>Command</td>
<td>TaskType, ProcessService &amp; ExternalProcessBuilder</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>CommandDecorator</td>
<td>TaskType, TaskProcessCommandDecorator</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>EnvironmentVariablesAwareBuilder &amp;</td>
<td>TaskType, EnvironmentVariableAccessor and ExternalProcessBuilder</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>MutableEnvironmentVariablesAwareBuilder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PostConfigurableBuilder</td>
<td>TaskConfigurator, TaskRequirementSupport</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>BuilderModuleDescriptor</td>
<td>TaskModuleDescriptor</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>BuilderManager</td>
<td>TasksManager</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>BuilderConfigHelper</td>
<td>No Direct Replacement, however, TaskConfiguratorHelper may be useful</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>LabelPathMap</td>
<td>TaskExecutableType &amp; CapabilityDefaultsHelper</td>
<td>For more details on Tasks see our Tasks Overview page</td>
</tr>
<tr>
<td>Maven2LogHelper</td>
<td>MavenLogHelper</td>
<td></td>
</tr>
<tr>
<td>VariableSubstitutionBean</td>
<td>CustomVariableContext</td>
<td></td>
</tr>
<tr>
<td>MutableQuietPeriodAwareRepository</td>
<td>No Replacement</td>
<td>QuietPeriodAwareRepository is still valid</td>
</tr>
<tr>
<td>com.atlassian.bamboo.builder.Matcher</td>
<td>Logic included in MavenLogHelper</td>
<td>You may also be interested in the adding a LogInterceptor to the LogInterceptorStack in the BuildLogger</td>
</tr>
</tbody>
</table>

### Deprecated Methods

#### Deprecated

- Bamboo.getBuilderManager()
- BuildDefinition.getBuilder()
- BuildDefinition.setBuilder()
- BuildPlanDefinition.getBuilderV2()

- UIConfigBean.getAllBuilders()

- UIConfigBean.getBuilderTypes()
- BambooPredicates.planHasBuilderOfKey(String builderKey)
- RequirementSet.getRequirement(@NotNull String key)

- RequirementSet.removeRequirement(@NotNull String key)

- AdministrationConfiguration.setGlobalVariable()
  AdministrationConfiguration.getGlobalVariable()
  AdministrationConfiguration.removeGlobalVariable()
  http://docs.atlassian.com/atlassian-bamboo/3.1/com/atlassian/bamboo/configuration/AdministrationConfiguration.html#setGlobalVariable()

- RepositoryV2.getSourceCodeDirectory(@NotNull String planKey)
- BuildLogger.startStreamingBuildLogs(int buildNum, String planKey)
- BuildsMovedEvent.getMovedBuilds()
- ElasticInstanceManager.registerElasticAgentStarted()
- FileCopier.copyFile()
- ResultsSummaryManager.getAllResultSummariesForPlan(Plan plan)

### New Integration Points
Tasks

As mentioned above we have introduced Tasks. This is a new plugin point allowing you to easily create multiple execution steps in your Build.
We've made writing new Task plugins incredibly easy. You can find all the information on creating Tasks on our Tasks Overview page.

Label Events

We now have ResultLabelAdded, ResultLabelRemoved, FavouriteAdded and FavouriteRemoved Events. You can listen for these events using an Event Listener Module.

Managing caches

Repository (or actually any other module) can implement CacheHandler to have it's caches managed via Administration->Manage Caches page

Log interceptors

We've added infrastructure for analysing build logs on the fly. For documentation, see LogInterceptor. You can add these interceptors to the LogInterceptorStack in the BuildLogger. For examples of usage, see com.atlassian.bamboo.build.logger.interceptors.StringMatchingInterceptor and com.atlassian.bamboo.plugins.ant.task.AntBuildTask.

Gravatar Support

Want to use peoples avatars in your plugin. Just inject the GravatarService into your server side plugin, or use the freemarker macro:

```markdown
[@ui.displayUserGravatar username="msmith" size="26"/]
```

Changes for Bamboo 3.0

- General Notes
- Changed Classes/Methods
  - ConfigurablePlugin
- Deprecated Classes
- Deprecated Methods
- Removed Classes/Methods
  - Build Working Directory API in SystemDirectory.java
  - BuildManager
  - Methods on BuildPlanDefinition and BuildDefinition
- REST Changes
- New Plugin Points
- Ability to add Tabs to the Bamboo Dashboard
- Support for Web Panels
- Build Trigger Condition Module
- Miscellaneous
  - Context Aware Navigation
  - Changes to Bamboo's Page Decorators
  - Changes to Form Layout (AUI Forms)

General Notes

There have been two major features in Bamboo 3.0, Artifact Passing and the new plan/result UI. Both of these have required some core code changes which may affect your plugins.

- As part of the Artifact Passing work, we have moved artifact definitions out of the buildDefinition xml, and they have become an entity in their own right. This has resulted in many API changes around artifacts. Storage of artifacts has also been moved (to a different directory structure).
- The UI work we did may affect the rendering of your pages inside Bamboo, particularly on the Plan, Job and Result views.
- We have moved to AUI Forms which will alter the layout of any form fields your plugin provides
- We have also continued down the path started in 2.7 in removing from Build specific interfaces in favour of the more generic Plan interfaces.
- Several methods/classes deprecated in 2.7 have been removed in 3.0.

Below you can find the details of all the changes we have made, and suggested workaround/replacements. If there is anything that we have missed please let us know.

Changed Classes/Methods

ConfigurablePlugin

Methods on the ConfigurablePlugin interface have been changed from
/**
 * Extension point for adding/customizing requirements when editing a build's builder configuration.
 * @param buildConfiguration source of information for customizing build requirements
 * @param requirementSet requirements to be customized
 */
void customizeBuildRequirements(@NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet);

/**
 * Extension point for removing requirements when given plugin is excluded from build's builder configuration.
 * @param buildConfiguration source of information for customizing build requirements
 * @param requirementSet requirements to be customized
 */
void removeBuildRequirements(@NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet);

This should be a painless change if your plugin extends BaseConfigurablePlugin, where default implementation calls old methods (see below) however, these old methods are deprecated so you should not rely on them in the future.
public abstract class BaseConfigurablePlugin extends BaseBuildConfigurationAwarePlugin implements ConfigurablePlugin {
    public void customizeBuildRequirements(@NotNull PlanKey planKey, @NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet) {
        customizeBuildRequirements(buildConfiguration, requirementSet);
    }

    /**
     * @deprecated since 3.0 Use #customizeBuildRequirements(Plan, BuildConfiguration, RequirementSet)
     */
    @Deprecated
    public void customizeBuildRequirements(@NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet) {
    }

    public void removeBuildRequirements(@NotNull PlanKey planKey, @NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet) {
        removeBuildRequirements(buildConfiguration, requirementSet);
    }

    /**
     * @deprecated since 3.0 Use #removeBuildRequirements(Plan, BuildConfiguration, RequirementSet)
     */
    @Deprecated
    public void removeBuildRequirements(@NotNull BuildConfiguration buildConfiguration, @NotNull RequirementSet requirementSet) {
    }
}

### Deprecated Classes

<table>
<thead>
<tr>
<th>Deprecated</th>
<th>Replacement</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Artifact, Default Artifact        | ArtifactDefinitionBase, ArtifactDefinition and ArtifactDefinitionContext | • ArtifactDefinition is an interface for persistent class. Provides association to Buildable (producerJob). Should only be used server-side.  
• ArtifactDefinitionContext is an interface for a 'lightweight' representation of an artifact definition and is accessible through BuildContext. |
| DefaultArtifactLinkBuilder        | Use the constructor on the ArtifactLink class directly    |                                                                      |
| BuildResults                      | Use ExtraBuildResultsData interface instead              | We are slowly moving away from using the BuildResults class. We have introduced a new cut down interface, ExtraBuildResultsData to be used instead (which will provide us with more flexibility to remove the BuildResults class moving forward). |

### Deprecated Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Replacement</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArtifactLinkManager.addArtifactLink(@NotNull BuildResultsSummary buildResultsSummary, @NotNull String artifactLabel)</td>
<td>ArtifactLinkManager.addArtifactLink(@NotNull BuildResultsSummary producerResult, @NotNull ArtifactDefinitionBase artifactDefinitionBase) or ArtifactLinkManager.addArtifactLink(BuildResultsSummary producerResult, ArtifactDefinitionContext artifactDefinitionContext)</td>
<td></td>
</tr>
<tr>
<td>ErrorHandler.getErrors(String)</td>
<td>ErrorAccessor.getErrors(PlanKey planKey) or ErrorAccessor.getErrors(PlanResultKey planResultKey)</td>
<td>ErrorAcco-type-safe accessing PlanKey</td>
</tr>
</tbody>
</table>
Bamboo 3.1 Documentation

**BambooCollectionUtils.newArrayList(...)**
User Ordering.sortedCopy instead

**BuildDirectoryManager.getBuildWorkingDirectory(String planKey)**
BuildDirectoryManager.getBuildWorkingDirectory(PlanKey planKey)

**BuildDirectoryManager.getBuildWorkingDirectory()**
BuildDirectoryManager.getWorkingDirectoryOfCurrentAgent()  
getBuildD does not concurren so will ac in most c

**DashboardCachingManager.getPlan(@NotNull String planKey)**
DashboardCachingManager.getPlan(@NotNull PlanKey planKey)

**DashboardCachingManager.getPlan(@NotNull String chainKey)**
DashboardCachingManager.getPlan(@NotNull PlanKey planKey)

**DashboardCachingManager.getAllChains()**
DashboardCachingManager.getAllTopLevelPlans()  
The Dasboard has been work with

**DashboardCachingManager.updatePlanCache(@NotNull String planKey)**
DashboardCachingManager.updatePlanCache(@NotNull PlanKey planKey)

**DashboardCachingManager.removePlanFromCache(@NotNull String planKey)**
DashboardCachingManager.removePlanFromCache(@NotNull PlanKey planKey)

**BambooContainer.getErrorMessages(String buildKey)**
BambooContainer.getErrorMessage(PlanKey planKey)

**Comparators.get<X>Comparator()**
Comparators.get<X>Ordering()  
We have our comp orderings see any i several n being up clearer

### Removed Classes/Methods

**Build Working Directory API in SystemDirectory.java**

Starting with version 2.6, Bamboo build working directory structure has changed. Unfortunately old API to access that directory has not been removed - both old style access via SystemDirectory class and new style access via BuildDirectoryManager were still available. Only the new API guaranteed proper behaviour in all cases. With Bamboo 3.0, we’ve dropped the methods for build directory retrieval from SystemDirectory. Please use BuildDirectoryManager methods instead.

**BuildManager**

BuildManager was deprecated in 2.7 and dropped in 3.0.

In order to retrieve Plan information use the PlanManager.
In order to retrieve result information use the ResultsSummaryManager for database cached BuildResultsSummaries and BuildResultsSummary.getExtraBuildResultsData() to retrieve the ExtraBuildResultsData (aka BuildResults) from XML file. If the methods you require are not yet on the ExtraBuildResultsData class, check whether or not you can get the information from the BuildResultsSummary if not, you can call Narrow.to(x, BuildResults.class) to retrieve the deprecated BuildResults object (If you need to do this, please raise an issue to let us know).

**Methods on BuildPlanDefinition and BuildDefinition**

The following methods have been removed from the BuildPlanDefinition and the BuildDefinition interfaces:

**BuildPlanDefinition.java**

```java
@Nullable
Collection<Artifact> getArtifactDefinitions();
```
Bamboo 3.1 Documentation

### BuildDefinition.java

```java
/**
 * Get the custom {@link Artifact}s.
 *
 * @return The collection of artifacts, null, if the build does not have any artifacts
 */
Map<String, Artifact> getArtifacts();

/**
 * Add a new artifact to the existing build artifacts
 *
 * @param artifact
 */
void addArtifact(Artifact artifact);

/**
 * Replace the build's artifacts with these ones.
 *
 * @param artifacts
 */
void setArtifacts(Map<String, Artifact> artifacts);
```

Please use the `ArtifactDefinitionManager` to access a Plan's Artifacts.

### REST Changes

/`build` REST endpoint has been removed, please use `/result` instead. Expand parameters have also been changed from `builds.build` to `results.result`

### New Plugin Points

**Ability to add Tabs to the Bamboo Dashboard**

A new web-section has been made available allowing you to add a Web Item Module (tab) on to the Bamboo dashboard.

### Support for Web Panels

Bamboo now provides support for Web Panels. Web Panels allow you to provide arbitrary snippets of HTML to be inserted into a page. There are currently only a few locations available to place these panels, however, we hope to increase this list soon.

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan.navigator</td>
<td>Renders below the plan navigator on all Plan, Job, Plan result and Job result screens</td>
<td>3.0</td>
</tr>
<tr>
<td>job.configuration.artifact.definitions</td>
<td>Renders below the Artifact definitions table in the Job configuration</td>
<td>3.0</td>
</tr>
<tr>
<td>job.configuration.artifact.subscriptions</td>
<td>Renders below the Shared Artifacts table in the Job configuration</td>
<td>3.0</td>
</tr>
<tr>
<td>plan.result.artifacts</td>
<td>Renders below the default content of the Plan Result Artifacts tab</td>
<td>3.0</td>
</tr>
<tr>
<td>job.result.artifacts</td>
<td>Renders below the default content of the Job Result Artifacts tab</td>
<td>3.0</td>
</tr>
</tbody>
</table>

You can find information on how to use these on the Web Panel Module page.

### Build Trigger Condition Module

A new plugin point has been added to provides the ability to gate whether a build should be triggered or not. It uses a preference system so multiple trigger condition plugins can be used to determine the desired result. A build which fails the trigger condition will not be executed, there will be no trace of the existence of this build.

For more information on how to create a Trigger Condition Plugin see: Build Trigger Condition Module

### Miscellaneous

**Context Aware Navigation**

In Bamboo 3.0 we have added the Plan Navigator. When moving between Jobs and the Plan (as well as Job Results and Plan Results), the navigator remembers our context i.e. which tab you are on and will attempt to keep you on the same tab. If your plugin UI exists in a location which has the Plan Navigator, you may want to manually define where the Plan Navigator takes you. You can do this via extra properties in you XWork Plugin Module definition. You can find more information on the XWork Plugin Module page.
Changes to Bamboo’s Page Decorators

Bamboo will decorate any actions that are part of your plugin. Previously, this decoration usually only provided, for example, the header and footer. In Bamboo 3.0 some decorators are providing a lot more, which will hopefully make the life of the plugin developer easier. Because of these changes you might find some of your pages are displaying strangely. This will mainly affect pages which display a tab menu (Plan, Job, Plan Result, Job Result). Now everything outside the tab content will be provided by the decorator, meaning your plugin only needs to supply the content to go inside the tab.

You will need to ensure your plugins are using the correct decorator, remove any excess html from your plugin which is no longer needed (because the decorator is supplying it) and define which tab will be selected on your page.

You can find more details on using the decorators on the XWork Plugin Module page.

Changes to Form Layout (AUI Forms)

We have changed Bamboo's forms from those old blue background panels, to the Atlassian Standard. The will affect the layout of any forms you might have in your plugins. If you were already using the webwork templates for form elements (e.g. [@ww.textField .../]) there should be no code changes required. However you might want to assess whether your forms still look good and re-address any custom styling you may have put in place.

If you are not using the built in templates for form elements you will need to use AUI styling to get your forms to match. You can find information and examples on using AUI Forms in our AUI Documentation.

One thing which is not clear in that documentation is form field lengths. The standard field length is now much shorter than before and if you wish to alter the field lengths you can do so using the cssClass param. The available options are: long-field, medium-field (default), short-field e.g:

```java
[ww.textField labelKey='setup.install.build.directory' name='buildDir' required='true' cssClass="long-field" ]
```

Changes for Bamboo 2.7

Structure Changes

Bamboo 2.7 involves the introduction of Staged Builds. Each Plan contains one or many stages, and each stage contains one or many Jobs. This has involved significant back-end changes to accommodate this feature. A Build used to be the core object of Bamboo. This class is now deprecated and in its place are the Chain and the Job. The functionality of the Build class has been split between these two classes and what the users now see as a "Plan" in the UI is represented by the Chain class.

Logical Break Up Of Responsibilities During The Build:

<table>
<thead>
<tr>
<th>Chain</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Repository For Change Detection</td>
<td>Source Repository For Checkout</td>
</tr>
<tr>
<td>Build Strategy and Triggering</td>
<td>Builder</td>
</tr>
<tr>
<td>Notifications</td>
<td>Requirements/Agent Matching</td>
</tr>
<tr>
<td>Permissions</td>
<td>Artifacts</td>
</tr>
</tbody>
</table>

You will notice in the above table that there is a repository configured for the Chain level and for the Job. The Chain's repository is used for build triggering and reporting included changes for the Chain result. The Job's repository is used for checking out the source code and reporting included changes for the job result. To make maintenance of your plans easier, you can set up your jobs to inherit the repository of the chain, however, this should be transparent to your plugins.

Both the Chain and the Job extend the Plan interface and share the one database table. There have also been two common interfaces introduced to help deal with commonalities between the implementations (TopLevelPlan and Buildable). As the Build class has now been deprecated most of Bamboo's internals have been updated, this includes managers and our plugin interfaces. Each case is different, but you will find some methods now deal with Plan, TopLevelPlan or Buildable, others will deal directly with the Chain or Job, whilst others will happily deal with all 5, you just have to tell it which one your after. When developing your plugin, if you are passed a common interface, you will need to ensure that the operation you wish to perform is actually applicable to the specific implementation.

Jobs are grouped in a Chain by ChainStages. ChainStages have a specific order inside the Chain, whilst Jobs inside a ChainStages do not have any order. The diagrams below shows the relationships and hierarchy of the new classes (The Build object has been included in the diagram to help explanation)
As jobs are a child of the chain, the identification key of the job also includes the full key of the chain. For example, a chain is represented by "BAMBOO-TRUNK" whilst the job is "BAMBOO-TRUNK-UNITTESTS".

The structure of result objects follows the same pattern as the plans. `ChainResultSummary` and `BuildResultsSummary` (Job) share the common interface `ResultsSummary`. Job results are grouped by `ChainStageResult`.

The New Build Process

A Chain (what users see as a Plan in the UI) consists of one or more Stages and each Stage consists of one or more Jobs. Before progressing to the next Stage, every Job in the previous Stage must have completed successfully. There is no order to Jobs within a Stage so when a Stage is executed, all Jobs in that Stage are placed in the queue immediately.

All triggering strategies now work at the Chain level. Chains do not take Job information into account when deciding whether to trigger. If repository polling or repository trigger strategies are selected, only changes detected against the Chain's are used to determine if the build will be triggered.

Below is a diagram which shows Bamboo's build process flow, as well as the available plugin modules you can build.
Build Process Per Job

Further Notes
1. **Build is Triggered** — Changes are detected via polling (against the Chain’s repository configuration only), manual builds, etc.

2. **Preparation for Build** — The Build Number is determined then ResultsSummary objects and BuildContexts are created for the Chain and every Job within the Chain. The build number information, latest revision keys and ResultSummaries are persisted to the database.

3. **Job Queued** — The server decides which agents can execute the job and queues the job.

4. **Agent Picks Up Job** — The capability context is set. The job is removed from queue. The agent begins looping through the build tasks.

5. **Retrieve Source** — 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.

6. **Prepare Job** — The agent runs the PrepareBuildTask. This begins streaming the logs back to the server. The agent also runs the CustomPreBuildAction plugin point.

7. **Executes the Job** — Timer begins. The agent runs the `Builder#executeBuild`. The CommandDecorator plugin will be run against the command line just before execution. After the Builder has been run, all the CustomBuildProcessors are run. Timer is stopped.

8. **Process Job Result** — The server runs CustomBuildProcessorServer, checks if the job has passed and saves and indexes the result. The server also fires off the `BuildCompletedEvent` and `PostBuildCompletedEvent` events, and executes any CustomPostBuildCompletedActions.

9. **On BuildCompletedEvent** *(performed by the server)* — Notifications are sent in this phase.

### Deprecated Classes

If any required functionality can not be found on the replacement classes please let us know.

<table>
<thead>
<tr>
<th>Deprecated Class</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildManager</td>
<td>PlanManager</td>
</tr>
<tr>
<td>BuildResultSummaryManager</td>
<td>ResultsSummaryManager</td>
</tr>
<tr>
<td>BuildResults</td>
<td></td>
</tr>
<tr>
<td>BuildIdentifierHelper</td>
<td></td>
</tr>
<tr>
<td>BuildPlanConstructorFactory, BuildPlanConstructor</td>
<td>There is no replacement for this, however the various implementations of the PlanCreationService may provide the required functionality</td>
</tr>
<tr>
<td>ExtendedBuildResultsSummary</td>
<td>BuildResultsSummary</td>
</tr>
</tbody>
</table>

### Deprecated Methods

<table>
<thead>
<tr>
<th>Deprecated Method</th>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildContext.getBuildPlanDefinition()</td>
<td>BuildContext.getBuildDefinition()</td>
</tr>
<tr>
<td>CurrentlyBuilding.getBuildTime()</td>
<td>CurrentlyBuilding.getElapsedTime()</td>
</tr>
<tr>
<td>DashboardCachingManager.getAllBuilds()</td>
<td>DashboardCachingManager.getAllTopLevelPlans()</td>
</tr>
</tbody>
</table>

### Removed Classes

<table>
<thead>
<tr>
<th>Removed Class</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotificationCondition</td>
<td>Deprecated since 2.2, The NotificationType module should be used instead. See Building a Notification Plugin.</td>
</tr>
<tr>
<td>BuildPlanCreationService</td>
<td>There is no replacement for this, however the various implementations of the PlanCreationService or the PlanValidationService may provide the required functionality. <a href="http://docs.atlassian.com/atlassian-bamboo/latest/com/atlassian/bamboo/caching/DashboardCachingManager.html">http://docs.atlassian.com/atlassian-bamboo/latest/com/atlassian/bamboo/caching/DashboardCachingManager.html</a></td>
</tr>
</tbody>
</table>

### REST Changes

**Build resource is replaced by Result resource.**

The build resource has been deprecated, however is still operational. All new queries should use `/result` resource instead. Syntax remains the same.
Currently running builds are available by REST

Runtime information for running builds is available via REST now. Not all result information is available while the build is running (for example, testResults information) so cannot be returned in the REST response. You will however be able to retrieve other valuable information such as progress of the build, commits and build logs (for Jobs only)

Chain result example:

```
<result id="2326529" number="6" lifeCycleState="InProgress" state="Unknown" key="CONF-CONFUNIT-6" expand="changes,metadata, stages, labels, jiraIssues, comments">
  <link rel="self" href="http://localhost:8085/bamboo/rest/api/latest/result/CONF-CONFUNIT-6"/>
  <buildStartedTime>2010-11-08T15:32:03.953+01:00</buildStartedTime>
  <buildDuration>0</buildDuration>
  <buildDurationDescription>Unknown</buildDurationDescription>
  <vcsRevisionKey>132473</vcsRevisionKey>
  <buildReason>Manual build</buildReason>
  <stages size="1" max-result="1" start-index="0"/>
  <labels size="0" max-result="0" start-index="0"/>
  <jiraIssues size="1368" max-result="1368" start-index="0"/>
  <comments size="0" max-result="0" start-index="0"/>
  <changes size="3171" max-result="3171" start-index="0"/>
  <metadata size="2" max-result="2" start-index="0"/>
  <progress>
    <isValid>true</isValid>
    <isUnderAverageTime>true</isUnderAverageTime>
    <percentageCompleted>0.2066399933736914</percentageCompleted>
    <percentageCompletedPretty>20%</percentageCompletedPretty>
    <prettyTimeRemaining>3 mins remaining</prettyTimeRemaining>
    <prettyTimeRemainingLong>Approximately 3 minutes remaining</prettyTimeRemainingLong>
    <averageBuildDuration>277681</averageBuildDuration>
    <prettyAverageBuildDuration>4 mins</prettyAverageBuildDuration>
    <buildTime>57380</buildTime>
    <prettyBuildTime>57 secs</prettyBuildTime>
  </progress>
</result>
```

Job result example

```
```
REST reflects internal structure of chains and jobs.

New expand points have been added for querying chain structure.

For example:

http://localhost:8085/bamboo/rest/api/latest/project/CONF?expand=plans.plan.stages.stage.plans.plan
Job key support

REST honours new key scheme, so Chain keys and Job keys may be used to address proper entity.

_isInBuildQueue_ attribute of RestPlan has been replaced with _isActive_

Can now pass in plan type to many different queries.

```
type=chain|job|build
```

May be applied to all queries that list plans and results, so /project /plan /result

Test results

Test results information is available only for Job level results. Chain results contain only statistical test results information, so there is no testResults expand point.

```
expand=testResults.all|testResults.successful|testResults.failed|testResults.newFailed|testResults.existingFailed
```

May be applied to any Job result or chain result if properly expanded (for Chains it should be prefixed by stages.stage.results.result.)

Only one category of test results can be expanded at a time.

Other Changes

Plan Configuration Changes

In 2.7 Bamboo has lost its Plan Creation Wizard. It now has a shortened one page form to create a new Plan. Plugin configuration (except for Builders and Repositories) will not be available at creation time. This should hopefully make writing these plugins easier as you no longer have to deal with both ways of creating configuration and there will always be a BuildDefinition and Plan object available to you.

Also, the view of the plan configuration has also been reformatted to a simplified one-page view. We encourage plugins, to limit the amount of space their view configuration takes up. And, if possible do not show anything if the plugin is not enabled (as opposed to showing "feature
X is disabled*)

There is also now a new tab on both the Plan and Job Configuration available for plugin developers to put their plugin configuration - the "Miscellaneous" tab. To make use of this tab, use the Additional Build Configuration Plugin point and/or ensure you plugin class implements MiscellaneousBuildConfigurationPlugin interface. You will notice that this interface actually provides you with the method isApplicableTo(plan). This method allows you to decide whether you plugin applies to Plans or Jobs or both.

To allow for the condensed creation page there is a new type of template (optional) on the Repository interface. This allows the UI to show the minimum required configuration on the creation page whilst allowing more advanced configuration after creating the initial plan. If you are extending AbstractRepository, you can just provide the new template type in your plugin definition (see the example below). If you don't supply the minimal template, the normal edit template will be used. If implementing the Repository interface directly you will need to implement the getMinimalEditHtml() method.

```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"/>
  <resource type="freemarker" name="minimalEdit" location="templates/plugins/repository/svnRepositoryMinimalEdit.ftl"/>
</repository>
```

Method Signature Changes

Base Configurable Plugin, Build Configuration Aware Plugin

Most plugin points which allow you to display content in Bamboo's UI, provide you with a getEditHtml() method to implement. Many have base implementations inside Bamboo, so it may not effect you. But, if you have specifically implemented/overridden this method in your plugin you will need to ensure that it properly handles a null plan argument

```java
/**
 * Returns HTML that represents the form fragment for editing the plugin. No opening and
 * closing form tags are
 * required.
 * @param buildConfiguration BuildConfiguration for which edit HTML has to be rendered.
 * @param plan  Plan for which edit HTML has to be rendered.  May be null during creation
 * phase of plan.
 * @return HTML string. May be null.
 */
@Nullable
String getEditHtml(@NotNull BuildConfiguration buildConfiguration, @Nullable Plan plan);
```

Build Complete Action Module

The CustomBuildCompleteAction interface has been updated to take in a Buildable instead of a Build inline with the new plan class hierarchy. However we actually suggest you move away from using this plugin point (it can be prone to thread safety problems) and instead use the http://docs.atlassian.com/atlassian-bamboo/latest/com/atlassian/bamboo/build/CustomPostBuildCompletedAction.htmlPost-Build Completed Action Module or the Build Processor Server Module.

```java
/**
 * This action will run when a build completes.
 * @param buildable - the Buildable plan object
 * @param buildResult - the BuildResults object that holds information
 */
public void run(@NotNull Buildable buildable, @NotNull BuildResults buildResult);
```

Trigger Reasons

The TriggerReason interface (Trigger Reason Module) has been updated to take in a ResultsSummary instead of a BuildResultsSummary. You will need to ensure any implementations of this will work with both Chains and Jobs. Also, it should be noted that Jobs do not have their own Trigger Reason, they will inherit the trigger reason of their parent.

Web Repository Module

The WebRepositoryViewer methods have also been updated to accept a ResultsSummary instead of a BuildResultsSummary and therefore also needs to be updated to support Chains as well as Jobs.

Report Module
The `ReportCollector` interface has been updated to accept a `ResultsSummary` instead of a `BuildResultsSummary` and therefore also needs to be updated to support Chains as well as Jobs.

**Notification Changes**

**Notification Type** can now be scoped to determine where it can be configured. There are three options (though poorly named):

- **Chain** - For Plan level notifications
- **Plan** - for Job level notifications
- **System** - for global notifications (configured in the administration section of Bamboo)

If you don’t provide a scope, it will default to a Job level notification. Also, note that whilst there are Job and Plan level notifications, they are both configured at the plan level. Job notifications will generally apply to all jobs within the Plan.

**Changes for Bamboo 2.6**

On this page:

- Developing for Bamboo 2.6
  - The ConfigurablePlugin Interface (and BaseConfigurablePlugin)
  - `com.atlassian.bamboo.results.BuildResults` has been deprecated
  - Notification Dispatcher
  - Some methods in `com.atlassian.bamboo.event.BuildCompletedEvent` have been removed
  - `TestResultError`
  - `ProjectManager/BuildManager split`
  - `Build/Plan split`
  - `BuildExecutionManager.BuildDetectionAction interface`
  - Rest changes
  - System property `DISABLE_SUCCESSFUL_TEST_INDEX`

**Developing for Bamboo 2.6**

If you are a Bamboo plugin developer, this page outlines changes in Bamboo 2.6 that may affect Bamboo plugins compiled for Bamboo version 2.5.x or earlier.

**The ConfigurablePlugin Interface (and BaseConfigurablePlugin)**

The ConfigurablePlugin interface is used by any plugins which take in configuration information via the plan wizard. The BaseConfigurablePlugin is an Abstract class which helps to implement the methods of the ConfigurablePlugin interface.

The following interface methods have been changed in the ConfigurablePlugin

```java
String getViewHtml(@NotNull Build build);
```

is now

```java
String getViewHtml(@NotNull Plan plan);
```

```java
String getEditHtml(@NotNull BuildConfiguration buildConfiguration, @NotNull Build build);
```

is now

```java
String getEditHtml(@NotNull BuildConfiguration buildConfiguration, @NotNull Plan plan);
```

And on the abstract class (you might have overridden these methods)

```java
protected void populateContextForView(@NotNull Map<String, Object> context, @NotNull Build build)
```

is now

```java
protected void populateContextForView(@NotNull Map<String, Object> context, @NotNull Plan plan)
```

```java
protected void populateContextForEdit(@NotNull Map<String, Object> context, @NotNull BuildConfiguration buildConfiguration, @NotNull Build build)
```

is now

```java
protected void populateContextForEdit(@NotNull Map<String, Object> context, @NotNull BuildConfiguration buildConfiguration, @NotNull Plan plan)
```

**The implications**

A Build is a subclass of a Plan so you will just need to update the method signatures of your implementing/overriding classes. All usages of
these methods should be able to remain the same.

**com.atlassian.bamboo.results.BuildResults has been deprecated**

The `com.atlassian.bamboo.results.BuildResults` class has been deprecated and will be removed in Bamboo 3.0. Instead, please use the `BuildResultSummary` class where possible. This is particularly important for freemarker templates, where the build results objects are no longer available for builds that have never been built.

**Notification Dispatcher**

The `notificationDispatcher` has moved modules. The implications of this is that if your plugin uses the `notificationDispatcher`, you might need to recompile/rebuild your plugin. No code changes should be required. However, if your plugin has a maven dependency on `atlassian-bamboo-core` purely so you can use the `notificationDispatcher`, you should be able to remove this dependency, which could speed up your builds slightly.

**Some methods in com.atlassian.bamboo.event.BuildCompletedEvent have been removed**

The `getBuildResults()`, `getBuild()` and `getBuildResultSummary()` methods of `com.atlassian.bamboo.event.BuildCompletedEvent` have been removed but can be queried by using the `BuildManager` and `BuildResultsSummaryManager`.

**TestResultError**

The `com.atlassian.bamboo.results.tests.TestResultError` class has been removed. Test result errors now implement the `TestCaseResultError` interface (method names inside have changed too).

Please use the `com.atlassian.bamboo.resultsummary.tests.TestCaseResultErrorImpl` class in place of `TestResultError`.

**ProjectManager/BuildManager split**

Methods related to the management of projects (e.g. `getAllProjects`) are no longer available in the `BuildManager` interface. Instead, please use `ProjectManager` to access them.

**Build/Plan split**

Some of the methods that used to return objects implementing `Build` interface now return `Plan` interface.

**BuildExecutionManager.BuildDetectionAction interface**

Interface `BuildDetectionAction` is now standalone (extracted out of `BuildExecutionManager`).

**Rest changes**

Content returned from `/rest/api/latest/build/XXX-XX-X?expand=testResults.failed.testResult` may have changed slightly in Bamboo 2.6, particularly the format of the output in the `duration` fields.

Calls to `/api/rest/getBuildResultsDetails.action` will no longer return any test results. Please use the above url instead.

**System property DISABLE_SUCCESSFUL_TEST_INDEX**

Bamboo no longer indexes test results. Hence, this system property has been removed.

### 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](warning)

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/](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

```java
public Commandline getCommandLine(ReadOnlyCapabilitySet capabilitySet)
```

This will affect plugins who extend this class directly or extends the AbstractBuilder. Instead you should use the following methods

```java
/**
 * 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

```java
@Nullable
RepositoryV2 getRepositoryV2();
BuilderV2 getBuilderV2();
```

### 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:

```java
CapabilitySet capabilitySet = CapabilitySetProvider.getAgentCapabilitySet(buildAgent);
```

or

```java
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
They have been replaced with the following options

```java
public String substituteBambooVariables(@Nullable final String commandLine)
public String substituteBambooVariables(@Nullable final String commandline, @Nullable finalBuildContext buildContext, final BuildLogger buildLogger)
public String[] substituteBambooVariables(final String[] commandLine, @Nullable final BuildContext buildContext, final BuildLogger buildLogger)
public String substituteBambooVariables(@Nullable final String commandLine, @Nullable final BuildResults buildResults, final BuildLogger buildLogger)
```

New Plugin Points

- Event Listener Plugin Point
- Build Agent Requirement Filter Plugin Point

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 2.1

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

```java
com/atlassian/bamboo/jiraissues
com/atlassian/bamboo/jiraserver
```

It now has the following structure

```java
com/atlassian/bamboo/jira
com/atlassian/bamboo/jira/jiraissues
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 is available to you instead to retrieve issues from the jira server.

The JiraIssueUtils 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
public NotificationRule createNotificationRule(NotificationSet notificationSet,
String conditionKey, String conditionData,
String emailRecipients, String imRecipients,
String users, String groups, Set<String> roles);

public NotificationRule updateNotificationRule(NotificationRule oldRule,
String conditionKey, String conditionData,
String notificationEmailString, String notificationIMString,
String userArray, String groupArray, Set<String> roles);

public List<NotificationCondition> getAllConditions();
```

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:

- 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
- Post-Build Processing in Bamboo 2.0

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 a BuildContext. 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 method**

   ```java
   BuildChanges collectChangesSinceLastBuild(@NotNull String planKey, @NotNull String lastVcsRevisionKey) throws RepositoryException;
   ```

   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**

   ```java
   String retrieveSourceCode(@NotNull String planKey, @Nullable String vcsRevisionKey) throws RepositoryException;
   ```

   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.
public interface BuildTask extends Callable<BuildContext> {
  void init(@NotNull BuildContext buildContext);
  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 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.

Changes for Bamboo 2.4

Developing for Bamboo 2.4

Please note, due to some once-off versioning changes for Bamboo 2.4 (the major release only), Bamboo 2.4 should be considered to be Bamboo 2.4.0 for development purposes. Hence, if you developing plugins for Bamboo 2.4 (the major release only), you will need to develop against version 2.4.0 of the Bamboo artifacts.
Changes for Bamboo 2.5

Developing for Bamboo 2.5

New Method for Repository Interface

Please note, a new method has been added to the repository interface in line with web repository changes for 2.5. If you have extended the AbstractRepository class, this change will not affect you. However, if you have not extended the AbstractRepository class you will need to implement the new method before your code will compile.

Web Repository Functionality Removed from the Repository Plugin Point

Prior to Bamboo 2.5, the repository plugins themselves implemented the functionality that provided web repository links to the user. This web repository functionality has now been pulled out into its own extendible plugin point. This change provides more focused support for the different web repository viewers available.

The information (e.g. web repository url and module name) is no longer stored within the repository configuration. It is now an entity in its own right.

Bamboo now provides Fisheye URLs for all repository types (out of the box), including any external plugins. If you would like to make use of these, you can simply remove the existing web repository functionality from your repository plugin. However, if the repository plugin was providing specialised web repository support, you will need to move the old functionality over to a new Web Repository Viewer Module.

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.

Maven has also provided documentation for both 3rd party jars in general and Sun jars in particular.

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 2.4

1. Install Maven 2.1
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.

5. In the atlassian-bamboo directory run mvn idea:clean: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=126m</td>
</tr>
<tr>
<td>Program Params</td>
<td>8085 pathtobamboo/bamboo-web-app/src/main/webapp /</td>
</tr>
<tr>
<td>Working Dir</td>
<td>D:/dev/src/atlassian/bamboo-home</td>
</tr>
<tr>
<td>Class Path of Module</td>
<td>atlassian-bamboo-web-server</td>
</tr>
</tbody>
</table>

Building Bamboo 2.2 or Bamboo 2.3

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.

5. In the atlassian-bamboo directory run mvn idea:clean: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=126m</td>
</tr>
<tr>
<td>Program Params</td>
<td>8085 pathtobamboo/bamboo-web-app/src/main/webapp /</td>
</tr>
<tr>
<td>Working Dir</td>
<td>D:/dev/src/atlassian/bamboo-home</td>
</tr>
<tr>
<td>Class Path of Module</td>
<td>atlassian-bamboo-web-server</td>
</tr>
</tbody>
</table>

Building Bamboo 2.0.x or Bamboo 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.
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. 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=126m</td>
</tr>
<tr>
<td>Program Params</td>
<td>8085 path tobamboo/bamboo-web-app/src/main/webapp /</td>
</tr>
<tr>
<td>Working Dir</td>
<td>D:\dev\src\atlassian\bamboo-home</td>
</tr>
<tr>
<td>Class Path of Module</td>
<td>atlassian-bamboo-web-server</td>
</tr>
</tbody>
</table>

If you have an external instance of Bamboo running and you wish to debug it you can set up a remote debugger following these instructions. 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

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. Run mvn idea:clean idea:idea -DdownloadSources=true
6. This should generate atlassian-bamboo.* files in your source directory. Open the project with Idea.
7. 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
8. Do the same for the atlassian-bamboo-web-server but make it depended only on the atlassian-bamboo module.
9. Add the application. Mine looks like:

<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=126m</td>
</tr>
<tr>
<td>Program Params</td>
<td>8085 path tobamboo/bamboo-web-app/src/main/webapp /</td>
</tr>
<tr>
<td>Working Dir</td>
<td>D:\dev\src\atlassian\bamboo-home</td>
</tr>
<tr>
<td>Class Path of Module</td>
<td>atlassian-bamboo-web-server</td>
</tr>
</tbody>
</table>

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 is usually `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:

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[ERROR] FATAL ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Failed to resolve artifact.
GroupId: com.atlassian.pom
ArtifactId: atlassian-closedsource-pom
Version: 12
Reason: Unable to download the artifact from any repository
com.atlassian.pom:atlassian-closedsource-pom:pom:12
from the specified remote repositories:
central (http://repo1.maven.org/maven2)
```

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.Name$Table.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

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 -Dpackaging=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 -Dpackaging=jar -Dfile=/path/to/file 
-Dir=\[url\] -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.

Error:
ERROR BUILD ERROR
INFO ------------------------------------------------------------------------
INFO Failed to resolve artifact.
Missing: 
----------
1) org.ddsteps:ddsteps-httpserver-mock:jar:1.0-m1
Try downloading the file manually from the project website.
Then, install it using the command:
mvn install:install-file -DgroupId=org.ddsteps -DartifactId=ddsteps-httpserver-mock 
-Dversion=1.0-m1 -Dpackaging=jar -Dfile=/path/to/file
Alternatively, if you host your own repository you can deploy the file there:
mvn deploy:deploy-file -DgroupId=org.ddsteps -DartifactId=ddsteps-httpserver-mock -Dversion=1.0-m1 
-Dpackaging=jar -Dfile=/path/to/file -Durl=url -DrepositoryId=id
Path to dependency:
1) com.atlassian.bamboo:atlassian-bamboo-core:jar:2.2.3
2) org.ddsteps:ddsteps-httpserver-mock:jar:1.0-m1
Solution:
Grab http://repo.ddsteps.org/maven/release/org/ddsteps/ddsteps-httpserver-mock/1.0-m1/ddsteps-httpserver-mock-1.0-m1.jar
Then
mvn install:install-file -DgroupId=org.ddsteps -DartifactId=ddsteps-httpserver-mock -Dversion=1.0-m1 -Dpackaging=jar -Dfile=/path/to/file
Then retry the build.

In IDEA

Error:

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.

Contributing to the Bamboo Documentation

Would you like to share your Bamboo hints, tips and techniques with us and with other Bamboo users? We welcome your contributions.

On this page:

- Blogging your Technical Tips and Guides - Tips of the Trade
- Contributing Documentation in Other Languages
- Updating the Documentation Itself
Bamboo 3.1 Documentation

- Getting Permission to Update the Documentation
- Following our Style Guide
- How we Manage Community Updates

Blogging your Technical Tips and Guides – Tips of the Trade

Have you written a blog post describing a specific configuration of Bamboo or a neat trick that you have discovered? Let us know, and we will link to your blog from our documentation. More....

Contributing Documentation in Other Languages

Have you written a guide to Bamboo in a language other than English, or translated one of our guides? Let us know, and we will link to your guide from our documentation. More....

Updating the Documentation Itself

Have you found a mistake in the documentation, or do you have a small addition that would be so easy to add yourself rather than asking us to do it? You can update the documentation page directly.

Getting Permission to Update the Documentation

Our documentation wiki contains developer-focused documentation (such as API guides, plugin and gadget development guides and guides to other frameworks) as well as product documentation (user's guides, administrator's guides and installation guides). The wiki permissions are different for each type of documentation.

- If you want to update the Developer Network or other developer-focused wiki spaces, just sign up for a wiki username then log in and make the change.
- If you want to update the Bamboo product documentation, we ask you to sign the Atlassian Contributor License Agreement (ACLA) before we grant you wiki permissions to update the documentation space. Please read the ACLA to see the terms of the agreement and the documentation it covers. Then sign and submit the agreement as described on the form attached to that page.

Following our Style Guide

Please read our short guidelines for authors.

How we Manage Community Updates

Here is a quick guide to how we manage community contributions to our documentation and the copyright that applies to the documentation:

- Monitoring by technical writers. The Atlassian technical writers monitor the updates to the documentation spaces, using RSS feeds and watching the spaces. If someone makes an update that needs some attention from us, we will make the necessary changes.
- Wiki permissions. We use wiki permissions to determine who can edit the various types of documentation spaces.
  - Developer documentation (API guides, plugin development and gadget development): Anyone can edit these spaces, provided they have signed up for a wiki username and logged in to the wiki.
  - Product documentation (user's guides, administrator's guides, installation guides): We ask people to sign the Atlassian Contributor License Agreement (ACLA) and submit it to us. That allows us to verify that the applicant is a real person. Then we give them permission to update the documentation.
- Copyright. The Atlassian documentation is published under a Creative Commons 'cc-by' license. Specifically, we use a Creative Commons Attribution 2.5 Australia License. This means that anyone can copy, distribute and adapt our documentation provided they acknowledge the source of the documentation. The cc-by license is shown in the footer of every page, so that anyone who contributes to our documentation knows that their contribution falls under the same copyright.

RELATED TOPICS

Tips of the Trade
Author Guidelines
Atlassian Contributor License Agreement

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 them 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
- Setting up phpUnit on Elastic Bamboo
- Bamboo JMeter Aggregator - Getting the most from performance builds
- Bamboo plugins for Git and GitHub
- Secure Installation of Bamboo

### 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: Working with 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?

**Setting up phpUnit on Elastic Bamboo**

- By: Michael White
- About: Getting automated unit testing working for your PHP code. (Note, this tutorial is written for JIRA Studio, however the instructions can be applied to non-JIRA Studio installations of Bamboo).
- Date: 20 Oct 2010
- 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 the Source Repository for a Plan

Installation

Secure Installation of Bamboo
- By: Stéphane Bagnier, on the ‘Antelink Blog’
- About: Part of a series about the complete installation of the Atlassian suite behind a proxy with SSL everywhere
- Date: 14 December 2010
- Related documentation: Bamboo Installation Guide

Have you written a technical tip for Bamboo?
Add a comment to this page, linking to your blog post or article. We will include it if the content fits the requirements of this page.

Feedback?
Your first port of call should be the author of the linked blog post. If you want to let us know how useful (or otherwise) a linked post is, please add a comment to this page.

Other Sources of Information

Bamboo documentation
Evaluator resources
Atlassian website
Atlassian forums
Atlassian blog
Bamboo plugins

Bamboo Documentation in Other Languages

Below are some links to Bamboo documentation written in other languages. In some cases, the documentation may be a translation of the English documentation. In other cases, the documentation is an alternative guide written from scratch in another language. This page presents an opportunity for customers and community authors to share documentation that they have written in other languages.

Please be aware that these are external guides.
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. The information in the linked guides has not been tested or reviewed by Atlassian.

On this page:
- No guides yet
Adding Your Own Guide to this Page

Have you written a guide for Bamboo in another language? Add a comment to this page, linking to your guide. We will include it if the content fits the requirements of this page.

Giving Feedback about One of the Guides

If you have feedback on one of the guides listed above, please give the feedback to the author of the linked guide.

If you want to let us know how useful (or otherwise) one of these guides is, please add a comment to this page.

Other Sources of Information

Bamboo documentation
Atlassian website
Atlassian blog
Bamboo plugins

TreeNavigation

Index

TreeNavigationVersions

Click for all versions

- Bamboo 3.1.x
- Bamboo 3.0.x
- Bamboo 2.7.x
- Bamboo 2.6.x
- Bamboo 2.5.x
- 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