

# Spira Importer Plugin

This section outlines how to use SpiraTest, SpiraPlan or SpiraTeam (hereafter referred to as SpiraPlan) in conjunction with the Jenkins continuous integration build server.

When you use the SpiraPlan plugin for Jenkins, it will allow you to associate each Jenkins project with a corresponding project and release in SpiraPlan. Then, each time Jenkins creates a new build, a new build artifact will be created in SpiraPlan. Each build in SpiraPlan will be automatically linked to the incidents fixed, source code revisions committed, and any SpiraPlan tokens in the Jenkins changelog will be parsed and turned into SpiraPlan artifact hyperlinks.

Now that the plugin has been installed, you need to go back to the Jenkins homepage and click on the "[Manage Jenkins](#)" hyperlink followed by the "[Configure System](#)" hyperlink. This will bring up the main Jenkins configuration page. Scroll down to find the "**Spira Integeration**" section:



Spira Integration	
Spira URL	<input type="text" value="http://localhost/SpiraTeam"/>
Username	<input type="text" value="fredbloggs"/>
Password	<input type="password" value="*****"/>

Enter in the **URL** you use to access your instance of SpiraPlan, together with a valid **username** and **password**. Once you have entered the values, click on the [Test Connection] button to verify that Jenkins can connect to SpiraPlan successfully.

Once it has connected successfully, click the [Save] button at the bottom of the screen to save your connection settings.

## Configuring a Jenkins Job

Now that you have setup the global SpiraPlan settings in Jenkins, next you need to associate each of your Jenkins Jobs with their corresponding SpiraPlan Project and Release/Iteration. To do this, click on the name of the Jenkins Job and then click on the "Configure" hyperlink for that Job:

Project name: Build JUnit  
 Description:

[Preview](#)

Discard Old Builds  
 This build is parameterized  
 Disable Build (No new builds will be executed until the project is re-enabled.)  
 Execute concurrent builds if necessary

**Advanced Project Options** [Advanced...](#)

---

**Source Code Management**

CVS  
 None  
 Subversion

Modules: Repository URL:   
 Local module directory (optional):

[Add more locations...](#)

Check-out Strategy:   
Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser:

[Advanced...](#)

Build periodically  
 Poll SCM

**Build Environment**

Enable Spira Integration

**Build**

**Post-build Actions**

Aggregate downstream test results  
 Archive the artifacts  
 Build other projects  
 Publish JUnit test result report  
 Publish Javadoc  
 Record fingerprints of files to track usage  
 E-mail Notification

Under the section **"Build Environment"** select the checkbox marked **"Enable Spira Integration"**. That will display the SpiraPlan configuration panel for this Job:

**Build Environment**

Enable Spira Integration

**Spira Configuration**

Project ID:   
 Release Version Number:

[Verify Release](#)

Now you need to enter the following values:

- **Project ID** – The numeric ID of the SpiraPlan Project that the Build belongs to. (e.g. for Project PR00001 just enter 1)
- **Release Version Number** – The version number of the SpiraPlan Release/Iteration that the Build belongs to. (e.g. for Release RL0004 with version number 1.0.0.0 you'd enter just 1.0.0.0)

Once you have entered in the Project ID and Release version number, click the [Verify Release] button and the plugin will connect to SpiraPlan and verify that the project exists, that the current user can connect to that project, and that the specified release/iteration exists in the project.

Once it has verified successfully, click the [Save] button at the bottom of the screen to save your Job configuration settings. You are now ready to use Jenkins with SpiraPlan.

## Viewing the Build Results in SpiraPlan

Now that you have associated your Jenkins job with a specific SpiraPlan project and release/iteration, you can now use Jenkins to manage your software builds and have the results of the build be reported back into SpiraPlan. For example when the 'Build JUnit' job illustrated in the previous section is executed, it will report back the following result in Jenkins:

add description

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">Build JUnit</a>	1 day 18 hr (#97)	1 day 18 hr (#95)	0.87 sec

Icon: [S](#) [M](#) [L](#)

Legend [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

The corresponding build entry will also be created in SpiraPlan under the specified project and release/iteration:

Build Name ▲▼	Creation Date ▲▼	Status ▲▼	Last Updated ▲▼	ID ▲▼
<a href="#">Build JUnit #97</a>	14-Mar-2012	Succeeded	14-Mar-2012	BL000045
<a href="#">Build JUnit #96</a>	14-Mar-2012	Succeeded	14-Mar-2012	BL000044
<a href="#">Build JUnit #97 - Succeeded</a>	14-Mar-2012	Succeeded	14-Mar-2012	BL000043
Started by user anonymous Building in workspace D:\Program Files\Jenkins\jobs\Build JUnit\workspace Updating svn://doctor/Common U test.txt At revision 17861				
<a href="#">Build JUnit #93</a>	14-Mar-2012	Failed	14-Mar-2012	BL000041
<a href="#">Build JUnit #92</a>	8-Mar-2012	Succeeded	8-Mar-2012	BL000040
<a href="#">Build JUnit #89</a>	8-Mar-2012	Succeeded	8-Mar-2012	BL000039
<a href="#">Build JUnit #88</a>	8-Mar-2012	Succeeded	8-Mar-2012	BL000038

If you have configured your Project Home to include the list of recent builds, the build information will also be displayed on the Project Home dashboard:

### Recent Builds

Name	Status	Creation Date
<a href="#">Build JUnit #97</a>	Succeeded	3/14/2012 3:34:02 PM
<a href="#">Build JUnit #96</a>	Succeeded	3/14/2012 3:33:15 PM
<a href="#">Build JUnit #95</a>	Failed	3/14/2012 3:31:33 PM
<b>Build JUnit #96 - Succeeded</b> Started by user anonymous Building in workspace D:\Program Files\Jenkins\jobs\Build JUnit\workspace Updating svn://doctor/Common At revision 17860 no change for svn://doctor/Common since the previous build		

Clicking on either of the hyperlinks will allow you to navigate to the Build details page inside SpiraPlan:

Welcome, System Administrator | Library Information System | My Profile | Administration | Log Out | Search

My Page | Project Home | **Planning** | Testing | Tracking | Reporting

Requirements | Releases > Build Details | Iterations | Planning Board

<< Back To Build List

Library System Release 1

- [Build JUnit #97](#)
- [Build JUnit #96](#)
- [Build JUnit #95](#)
- [Build JUnit #94](#)
- [Build JUnit #93](#)
- [Build JUnit #92](#)
- [Build JUnit #89](#)
- [Build JUnit #88](#)
- [Build JUnit #87](#)
- [Build JUnit #86](#)
- [Build JUnit #85](#)
- [Build JUnit #84](#)
- [Build JUnit #83](#)
- [Build JUnit #82](#)
- [Build JUnit #81](#)

**Build:** [Build JUnit #97](#) [BL:000045]

Name: Build JUnit #97

Description: Started by user anonymous Building in workspace D:\Program Files\Jenkins\jobs\Build JUnit\workspace Updating svn://doctor/Common U test.txt At revision 17861

Status: **Succeeded**

Creation Date: 3/14/2012 3:34:02 PM

Last Updated: 3/14/2012 3:34:04 PM

Incidents \* | Revisions \* | Test Runs

> Refresh | Apply Filter | Clear Filter | -- Show/Hide columns --

Incident Name ▲▼	Type ▲▼	Status ▲▼	Priority ▲▼	Detected By ▲▼	Creation
<a href="#">Cannot add a new book to the system</a>	Bug	Assigned	1 - Critical	Joe P Smith	4-Nov-200

Show 15 rows per page

This page will display the status (success / failure) and details of the build (from the Jenkins Console Output) together with a list of the associated incidents, test runs and source code revisions. The following section will explain how to use your Source Code Management (SCM) system to take advantage of the SpiraPlan plugin and automatically link incidents and source code revisions to the build information.

## Working with Source Code Changesets

When your developers commit changes to your application's source into the SCM repository, they should make sure to link the commit to the appropriate artifacts in SpiraPlan. For example they may want to record that the revision fixes a specific incident or implements a specific feature (requirement).

Linking an artifact is very simple. All the developer needs to do is enter the artifact token in the following format:

### [PREFIX:ID]

The first half, the Artifact Identifier, is a two-letter code that is used throughout SpiraPlan, and is visible on almost every page in the application. For example, a requirement's identifier is "RQ". Incidents are "IN", and tasks are "TK". The artifact ID is the number of the artifact. So by creating a commit message that reads:

*Due to requirement [RQ:12], the code for .toString in class XMLparser was modified. This also fixed Incident [IN:1034].*

SpiraPlan will automatically detect the tokens and will include links to them under the Associations tab for each revision detail in SpiraPlan.

In addition, when Jenkins creates the next build (that includes this revision), the plugin will automatically parse the revision message and convert the tokens into hyperlinks to the corresponding SpiraPlan artifact. That way, when developers view the build changelog in Jenkins, it will automatically include links to the SpiraPlan items:

The screenshot shows the Jenkins interface for Build #97 (Mar 14, 2012 3:34:02 PM). On the left, there are navigation links: Back to Project, Status, Changes, Console Output, Edit Build Information, Tag this build, and Previous Build. The main content area shows the build status and a list of changes. The first change is: "1. Fixed [IN:7] and [RQ:4] (detail)". Below this, it says "Started by anonymous user".

Meanwhile, inside SpiraPlan, the system will use the same information to automatically link the list of associated revisions to the build record:

Revision	Author	Summary	Commit Date	Content	Properties
17861	[REDACTED]	Fixed [IN:7] and [RQ:4]	14-Mar-2012	Yes	No

Showing 15 rows per page. Displaying page 1 of 1.

If the commit message contains Incident tokens, the plugin will also automatically link those incidents to the appropriate build:

Incident Name	Type	Status	Priority	Detected By	Creation Date	Owner	Build
Cannot add a new book to the system	Bug	Assigned	1 - Critical	Joe P Smith	4-Nov-2003	Joe P Smith	Build JUnit #97

Showing 15 rows per page. Displaying page 1 of 1.

Similarly when you view the list of incidents inside SpiraPlan you will now be able to sort and filter the list by the associated build:

Incident Name	Type	Status	Priority	Detected By	Creation Date	Owner	Build	ID	Edit
Cannot log into the application	Incident	New	1 - Critical	Fred Bloggs	1-Nov-2003		Build JUnit #92	IN000001	Edit
Not able to add new author	Incident	New		Joe P Smith	1-Nov-2003		Build JUnit #92	IN000002	Edit
Clicking on link throws fatal error	Incident	New		Fred Bloggs	1-Nov-2003		Build 0001	IN000003	Edit
Database not backing up correctly	Bug	Open		Joe P Smith	2-Nov-2003		Build 0001	IN000004	Edit
Cannot install system on Oracle 9i	Bug	Open	1 - Critical	Fred Bloggs	2-Nov-2003		Build 0002	IN000005	Edit
The book listing screen doesn't sort	Bug	Open	3 - Medium	Joe P Smith	2-Nov-2003		Build 0002	IN000006	Edit
Cannot add a new book to the system	Bug	Assigned	1 - Critical	Joe P Smith	4-Nov-2003	Joe P Smith	Build JUnit #97	IN000007	Edit
Editing the date on a book is clunky	Bug	Assigned	2 - High	Joe P Smith	4-Nov-2003	Fred Bloggs		IN000008	Edit
Editing the date on an author is clunky	Bug	Assigned	3 - Medium	Joe P Smith	4-Nov-2003	Joe P Smith	Build 0004	IN000009	Edit
Doesn't let me add a new category	Bug	Resolved	4 - Low	Fred Bloggs	4-Nov-2003	Fred Bloggs	Build 0004	IN000010	Edit
Validation on the edit book page	Bug	Resolved	1 - Critical	Fred Bloggs	15-Nov-2003	Joe P Smith	Build 0005	IN000011	Edit
Quote handling issues throughout	Bug	Resolved	2 - High	Fred Bloggs	15-Nov-2003	Fred Bloggs	Build 0005	IN000012	Edit
The tables get cutoff on low-res modes	Bug	Closed	3 - Medium	Joe P Smith	15-Nov-2003	Joe P Smith	Build 0005	IN000013	Edit
Permissions not updating when changed	Bug	Closed	4 - Low	Joe P Smith	15-Nov-2003	Fred Bloggs	Build 0005	IN000014	Edit
Session handling	Bug	Closed	1 - Critical	Joe P Smith	15-Nov-2003	Joe P Smith		IN000015	Edit

Congratulations! You are now able to use SpiraPlan and Jenkins to be able to manage your builds and have the build status integrated into your SpiraPlan project dashboard.

## Scheduling Test Sets Upon Successful Builds

One additional feature of the integration with SpiraTest and SpiraPlan (hereafter just SpiraTest) is the ability to have SpiraTest automatically schedule the execution of a test set whenever a build passes.

To do that, make sure the Test Set is associated with the SpiraTest release or iteration that is being built and then set the **Schedule on Build** field to "Yes" and optionally enter in the delay (after the build succeeds) that you want the test set to be scheduled for:

### Dates and Times

Creation Date: 1/1/2007 7:00:00 PM

Last Executed: 12/1/2003 5:45:20 AM

Last Updated: 1/1/2007 7:00:00 PM

Planned Date: 02/04/2007 6:00 ...

Recurrence: -- One Time --

Schedule on Build:  Yes

Post-Build Delay (s): 20

This means that you don't need to separately manage your build schedule in Jenkins and your test automation schedule in SpiraTest.