JiraTestResultReporter-plugin

Plugin Information
View JiraTestResultReporter on the plugin site for more information.

Creates issues in Jira for failed unit tests.

2.x Versions

WARNING: 2.x versions of this plugin are not compatible with the previous 1.x versions. What this means is that your configurations related to this plugin will not be imported from previous versions when you do an upgrade.

For details about previous versions usage and configuration, please see the **1.x Versions** section.

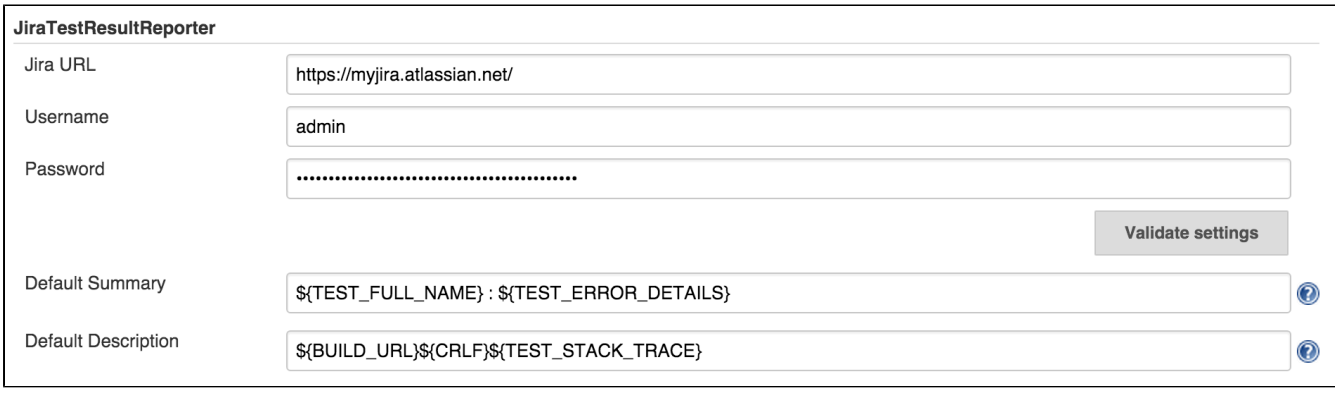
What it does

This plugin allows you to create and/or link issues from Jira to failed tests in Jenkins. The creation/linking is done directly in the Jenkins interface. For the creation of the issues you can supply a template for what is going to be added in most of the issue's fields.

Global Configuration

Before doing anything else, the global configurations must be done. In order to do these go to **Manage Jenkins -> Configure System -> JiraTestResultReporter** and enter here the JIRA server url the username and password. It is highly recommended that you click the Validate Settings button every time you make any changes here. Also from here you can configure the global templates for Summary and Description, by clicking on the Advanced button. These templates will be used to create issues if they are not overridden in the job configuration.

> **Use a dedicated Jira user**
>
> From a security and usage perspective, it's recommended to create a dedicated Jira user for the reporting.



Job Configuration

The first thing we need to do here is enabling the plugin:

- **Freestyle projects** and **Multi-configuration projects**
  First, JUnit test reports need to be enabled by going to **Add post-build action -> Publish JUnit test report**. Then check the box next to **JiraTestResultReporter** in the **Additional test report features section**.
- **Maven Project**
  Add post-build action -> Additional test report features -> check the box next to **JiraTestResultReporter**.

Configuration:

Insert the **Project key** in the respective field. Again, highly recommended to push the Validate Settings.

After setting the project key the **Issue type** select will be populated with the available issue types for that specific project.

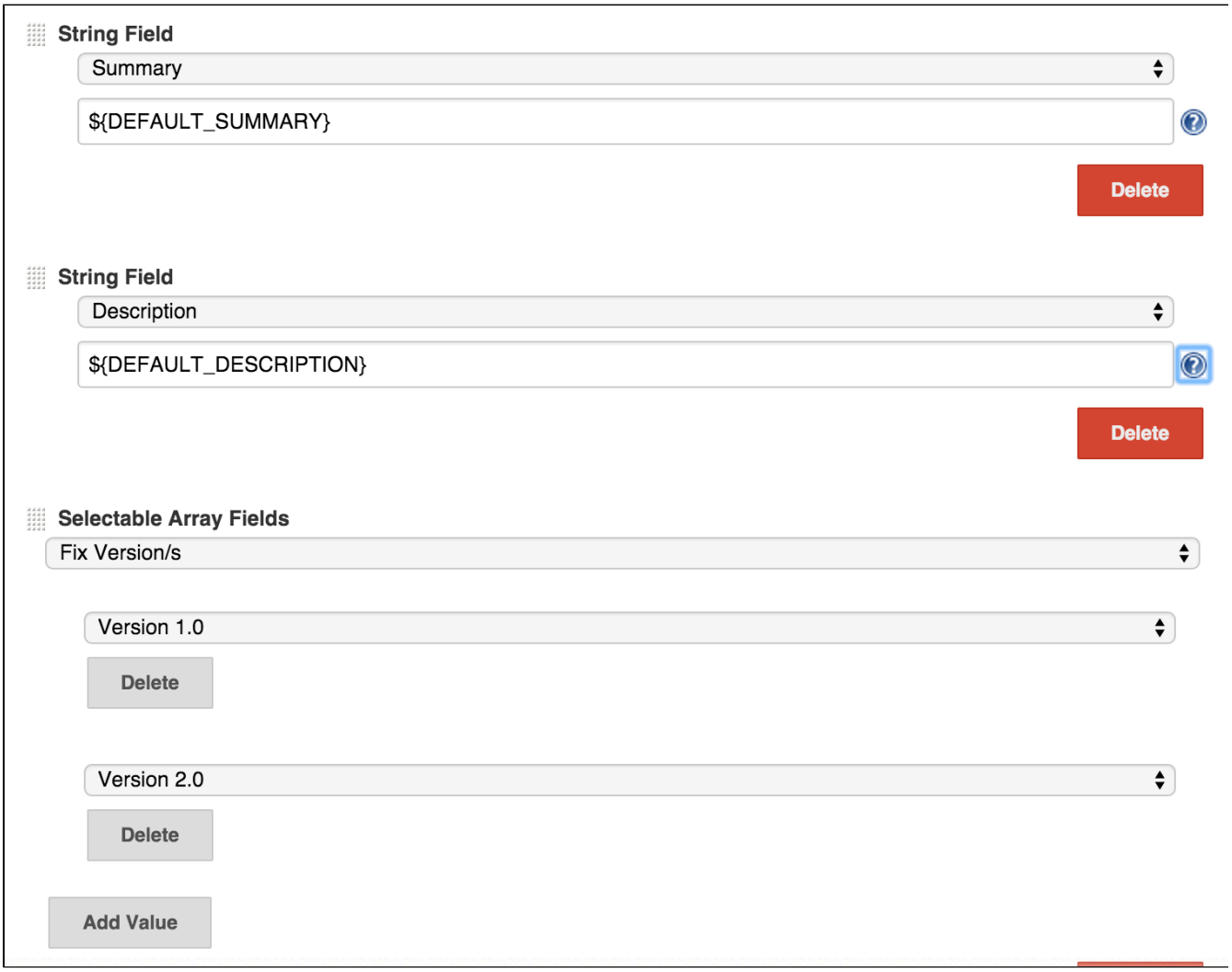
If you check the **Auto raise issue** check box, this plugin will create and link issues for all the failing tests in new builds that don't already have linked issues.

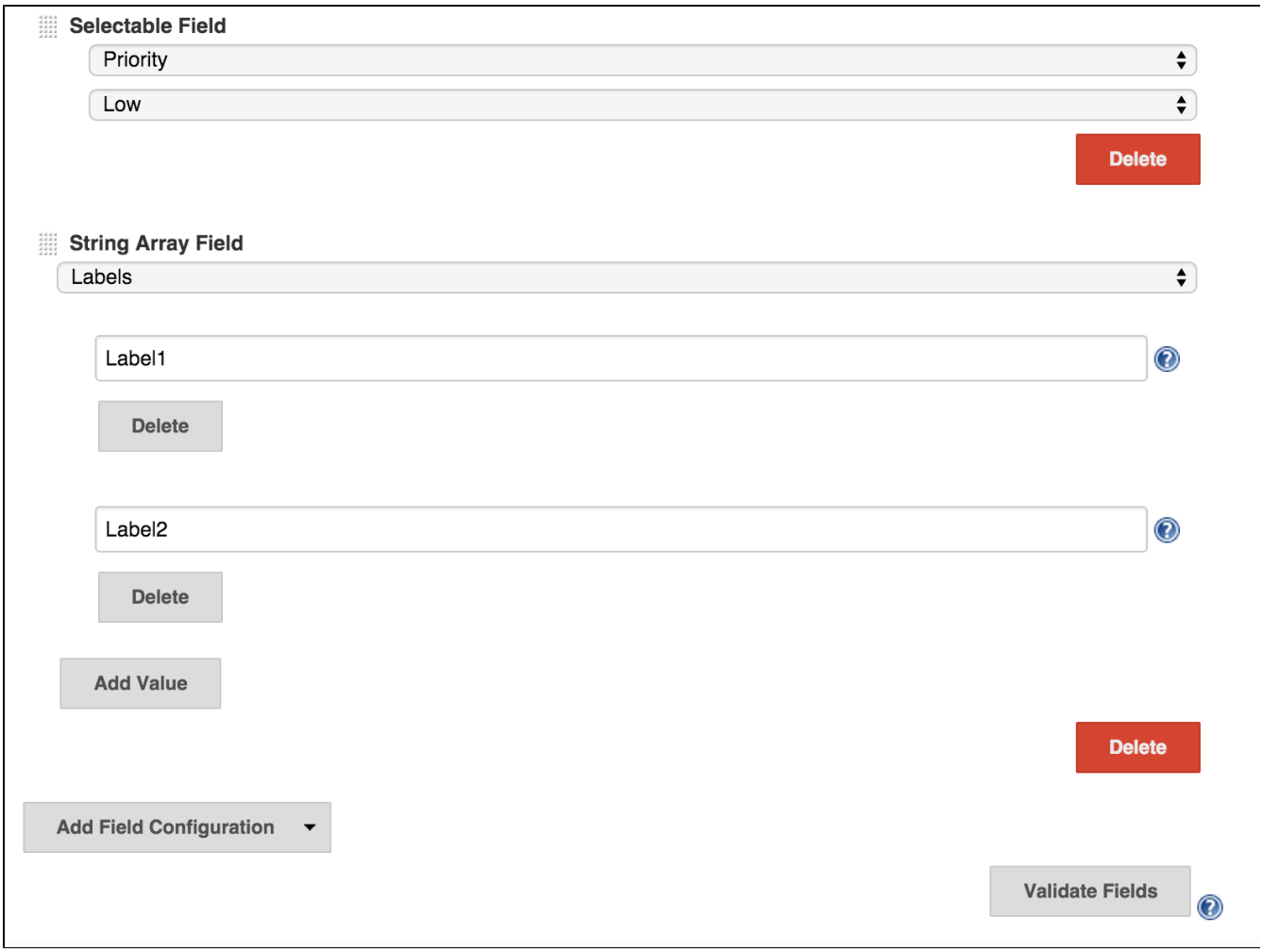
Checking **Auto resolve issue** check box will enable an experimental feature. Basically, if you had a test that was failing and you had a linked Jira issue to it, the plugin will try to resolve your issue. What this means is that it will look for available transitions from the current state and it will try to find one that in the name has the word "resolve". If such a transition is found it will apply it, otherwise it will just log a message. In future releases this will be configurable.

Only after configuring the fields above, if you want you can override the **Summary** and **Description** values by clicking the**Advanced** button. If you want, here you can configure all available fields for that specific issue type. Due to Jenkins interface development limitations, you have to search for the desired field from the four available types of fields, after clicking the Add Field Configuration.

Important: Do not leave empty values for fields, if you didn't find the desired field in the current chosen option, delete it before trying the next one.

Finally, I cannot say that this is recommended (although it is (smile)), **read the help tag for the Validate Fields** and if the warning there is not a problem for you click the button.

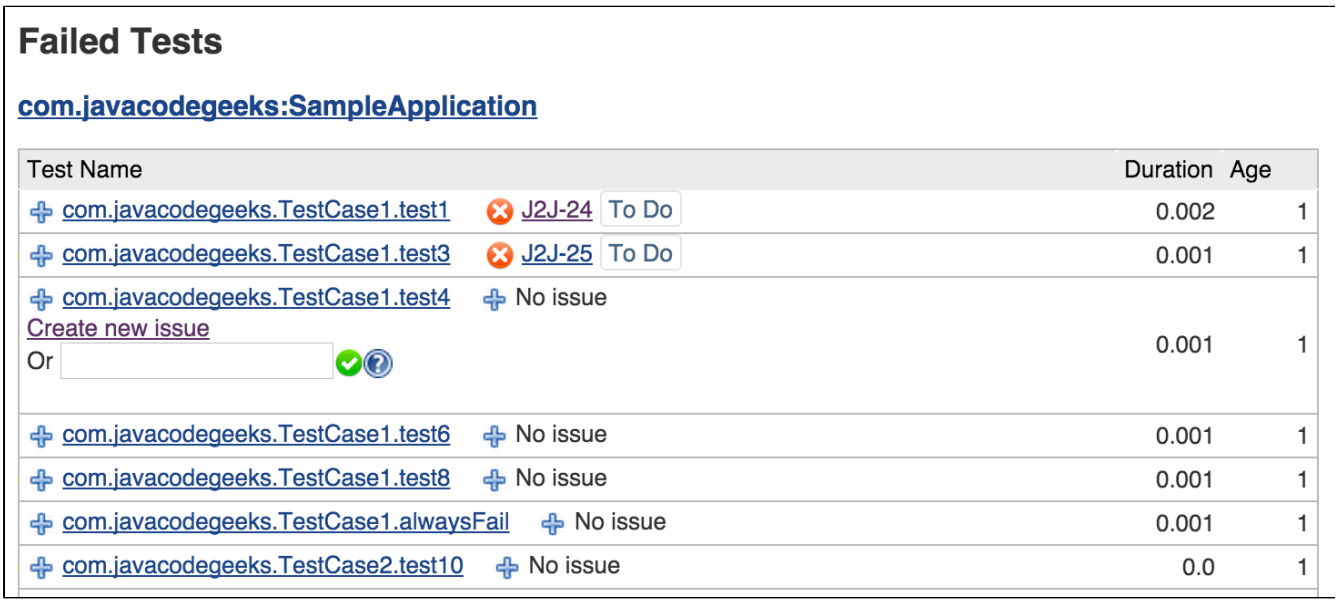




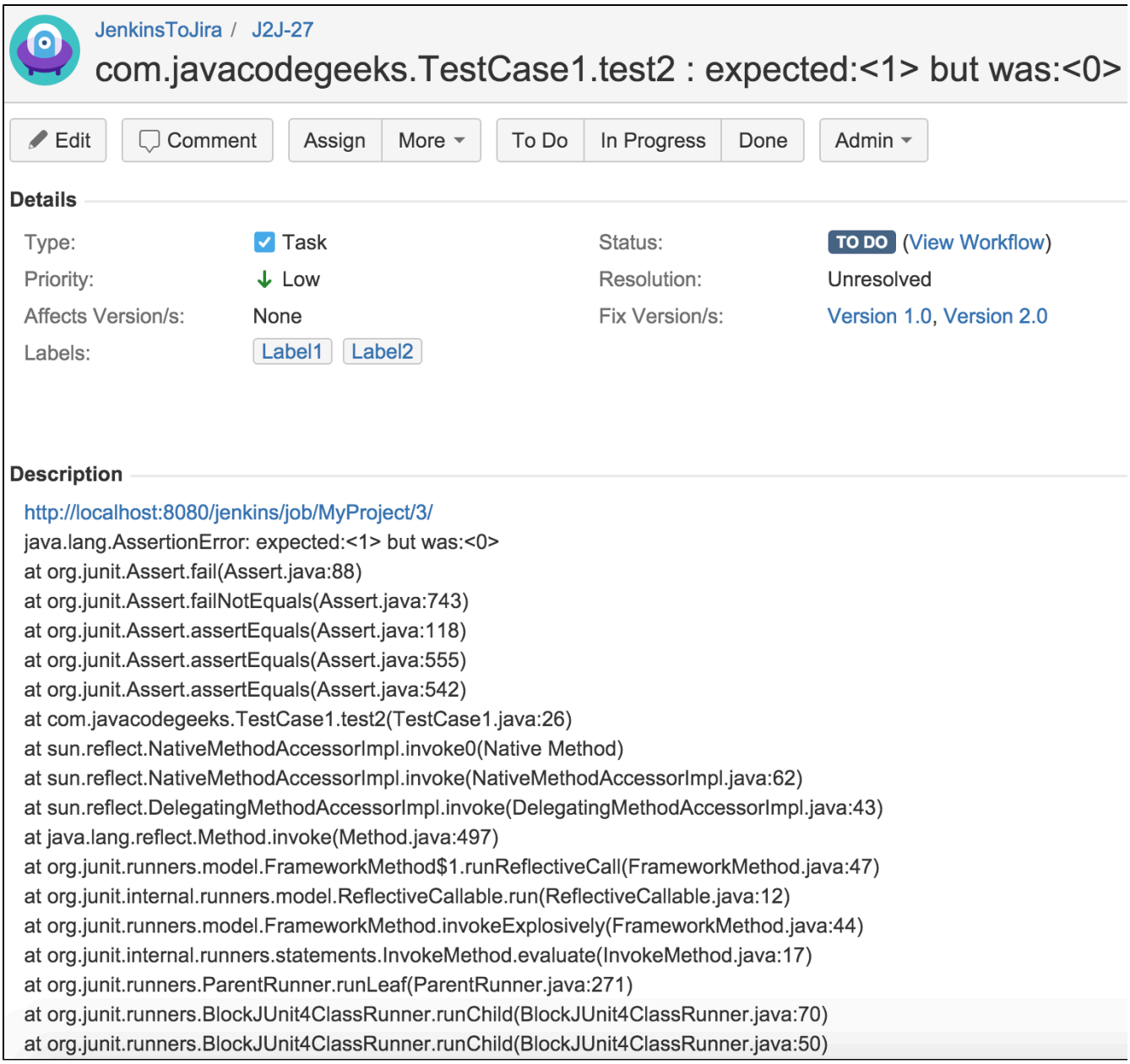
Usage

After building the project, go to the test results page. Next to the test cases you will see a **blue plus button**, next to a **No issue** message. If you want to **create an issue**, or **link an existing one**, click the blue plus button and choose the desired option. For **unlinking** an issue, click the **red x button**.

When creating, linking and unlinking issues, you it is recommended that wait for the page to reload, before doing something else for another test. Errors will be shown inline, if any.



Finally, your issues are created and you can see them by clicking the links directly from the Jenkins interface.



Variables

For text fields in the Job Configuration and Global Confinguration (Summary and Description only) you can use variables that will expand to the appropriate value when the issue is created in JIRA. You can use all the environment variables defined by Jenkins (see link). Additionaly, this plugin can expand a set of predefined variables that expose information about the test.

Insert a string value.

You can include Jenkins Environment variables (see link), or the following variables defined by this plugin:

Variable usage: **${VAR_NAME}**

CRFL - new line

DEFAULT_SUMMARY - configured in the global configuration page

DEFAULT_DESCRIPTION - configured in the global configuration page

TEST_RESULT

TEST_NAME

TEST_FULL_NAME

TEST_STACK_TRACE

TEST_ERROR_DETAILS

TEST_DURATION

TEST_PACKAGE_NAME

TEST_STDERR

TEST_STDOUT

TEST_OVERVIEW

TEST_AGE

TEST_PASS_COUNT

TEST_SKIPPED_COUNT

TEST_FAIL_SINCE

TEST_IS_REGRESSION - expands to true/false

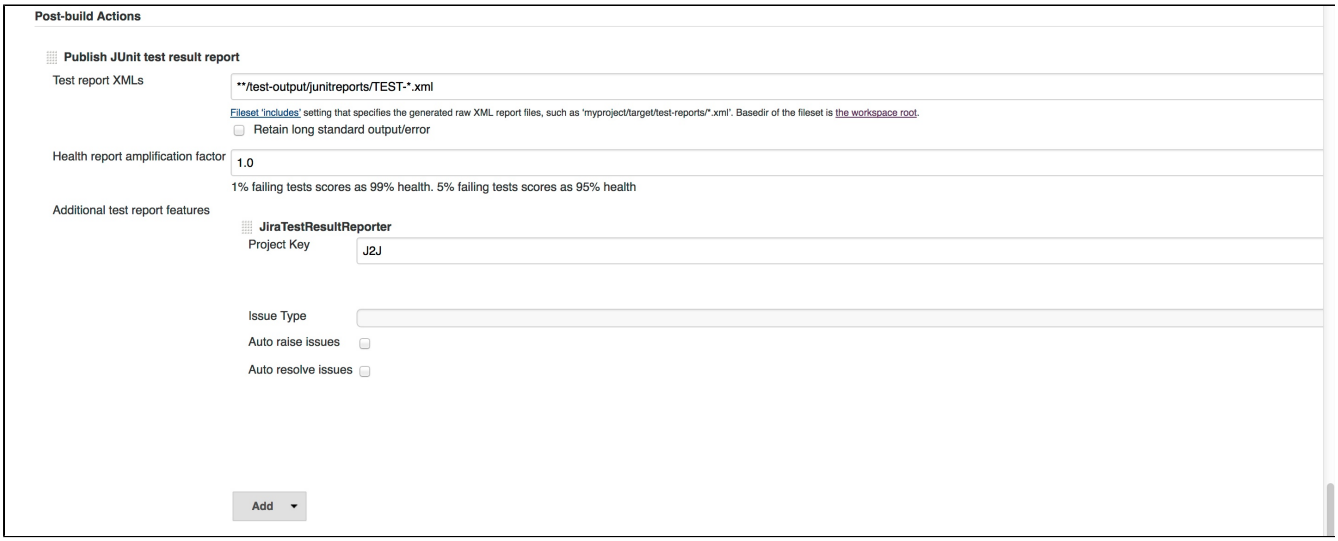
BUILD_RESULT

WARNING: Your input will not be validated against the server's metadata. Check Jira to make sure you insert a valid value for this field and use the Validate Fields button bellow, otherwise the plugin will fail to create your issue.

Integrations

TestNG

TestNG will automatically generate JUnit test reports that can be found by default in *<project-folder>/test-output/junitreports.* Unfortunately, you cannot use the published results by the Jenkins TestNG plugin with our plugin. In order to use the link functionality you must configure a JUnit publisher (**Post Build Actions -> Publish JUnit test result report**) as follows:



xUnit

You can use the functionality of the JiraTestResultReporter with all the testing standards that are supported by the xUnit plugin (see their wiki page for supported formats). The workflow would be as follows:

1. Run your testing tool that creates result files in a testing standard
2. Configure the xUnit plugin to convert said testing standard into JUnit
3. Configure the JUnit publisher (**Post Build Actions -> Publish JUnit test result report**)
4. Enable **Additional test report features** and configure **JiraTestResultRepoter** as explained in the **Job Configuration** section above

JSON API endpoint

You can view the test to issue mappings in JSON format by sending a request to the following url: http://my.jenkins.com/plugin/JiraTestResultReporter/testToIssueMapping/api/json?job=$JobName. For Freestyle and Maven jobs you simply query the url: http://my.jenkins.com/plugin/JiraTestResultReporter/testToIssueMapping/api/json?job=MyJob. For Matrix Projects sending a request as before will return you a JSON with mappings for all axis and if you want a specific axis you do as following: http://my.jenkins.com/plugin/JiraTestResultReporter/testToIssueMapping/api/json?job=MyJob/MyAxisName=MyAxisValue. It is recommended that you use url encoded characters.

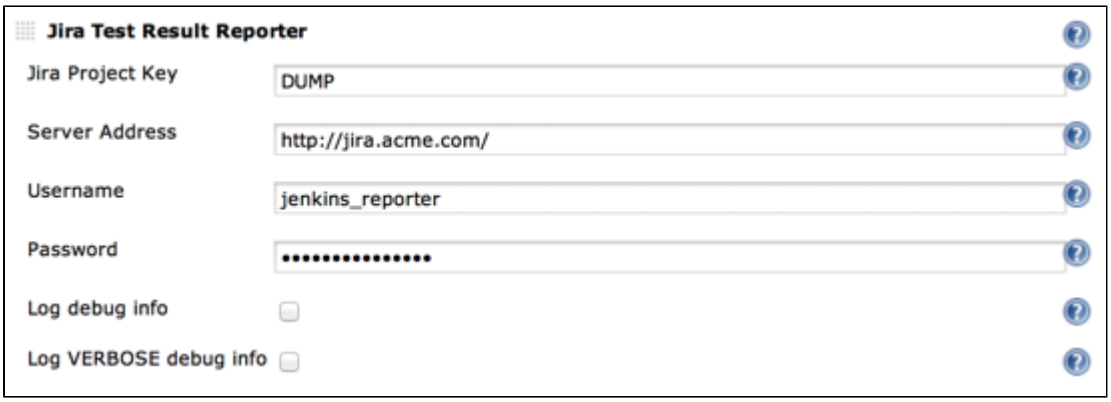
1.x Versions

What it does

This plugin examines the build job for failed unit tests. It work by using the Jenkins internal test result management for detecting failed tests. Just let Jenkins run and report your unit tests e.g. by adding the “Publish xUnit test results report” to your build job.

If JiraTestResultReporter detects new failed tests, it will create an issue for every test case in Jira:

Usage



- In the build job add JiraTestResultReporter as a post-build action.
- Configure the plugin for this job. See the help boxes for details.
- Build your job. If there are failed tests, the plugin will create issues for them. This will (should!) happen only once for every new failed tests; new in this case means tests that have an age of exactly 1.

Use a dedicated Jira user

From a security and usage perspective, it's recommended to create a dedicated Jira user for the reporting. This helps to identify (e.g. filter) issues created by the plugin.

Version history

Version	Changes
2.0.5	- JENKINS-47645 - java.lang.NoSuchMethodError on "Configure System" page / unable to edit configuration - JENKINS-44691 - Correlates a new test failure with a resolved jira issue
2.0.4	- JENKINS-40520 - Breaks Save button on Manage Jenkins/Configure System - JENKINS-39813 - [Doc] Add steps to integrate TestNG and xUnit - JENKINS-22405 - Error if the job has no testresults
2.0.3	- enable view of the linked issue even if the test passes - added API endpoint to view the test to issue mapping - added job configuration field that allows jira user fields
2.0.2	- fix configuration issue (JENKINS-34904) - added more logging - issue summary is now shown as tooltip when hovering over issue link
2.0.1	- fix packaging issue (JENKINS-34806) - fix UI bugs due to Jenkins changes
2.0.0	- the plugin was rewritten entirely
1.0.4	- new: validate config settings - new: option to create issues for old issues (to catch up on a project with unreported issues)
1.0.3	- fixed an issue that prevented storing of the configuration (Sorry!)
1.0.2	- first version