

Integration with Jenkins



Jenkins

- [Overview](#)
- [Release Notes](#)
- [Installation](#)
 - [Manual Installation](#)
- [Configuration](#)
 - [Jira servers](#)
- [Creating a new Project](#)
- [Build Steps](#)
 - [Xray: Cucumber Features Export Task](#)
 - [Configuration](#)
 - [Xray: Results Import Task](#)
 - [Configuration](#)
 - [Additional fields](#)
- [Examples](#)
 - [Cucumber](#)
 - [Exporting Cucumber features](#)
 - [Importing the execution results](#)
 - [Importing the execution results with user-defined field values](#)
 - [JUnit](#)
 - [Importing the execution results](#)
- [Troubleshooting](#)
 - [The build process is failing with status code 403](#)

Overview

Xray enables easy integration with Jenkins through the "Xray for JIRA Jenkins Plugin", providing the means for successful Continuous Integration by allowing users to report automated testing results.

Release Notes

- [Xray for JIRA Jenkins Plugin 1.2.1 Release Notes](#)
- [Xray for JIRA Jenkins Plugin 1.2.0 Release Notes](#)
- [Xray for JIRA Jenkins Plugin 1.1.0 Release Notes](#)
- [Xray for JIRA Jenkins Plugin 1.0.0 Release Notes](#)

Installation

The installation is made manually. For more information on how to install add-ons, please refer to [how to install add-ons](#).



Requirements

This app was tested against Jenkins v2.32.x and it may not work properly with previous versions.

Manual Installation

 **Download the latest version of the Jenkins Plugin**
You may download the latest version of the Jenkins plugin from the latest [Release Notes](#).

If you have the actual `xray-for-jira-connector.hpi` file,

- 1. Go to the Update Center of Jenkins in Manage Jenkins > Manage Plugins.
- 2. Select the advanced tab
- 3. In the Upload Plugin section, click upload and select the file `xray-for-jira-connector.hpi` file.

Configuration

Xray for Jenkins is configured in the global settings configuration page **Manage Jenkins > Configure System > Xray for Jira configuration**.

Jira servers

The Jira servers configuration defines connections with Jira instances.

To add a new Jira instance connection, you need to specify some properties:

- 1. **Configuration alias**
- 2. **Server Address:** The address of the Jira Server where Xray is running
- 3. Authentication:
 - a. **User:** username
 - b. **Password.**

Xray for JIRA configuration

JIRA servers

Configuration alias

Xray local

Server address

http://localhost:8080

Username

admin

Password

....

Test Connection

Delete instance

Configuration alias

Xray local V2

Server address

http://abc.xyz

Username

admin

Password

....

Test Connection

Delete instance

Save

Apply


Creating a new Project


The project is where the work that should be performed by Jenkins is configured.


For this app, you should configure a "Freestyle project". In the home page, clicking New Item > Freestyle project, provide a name, and then click OK.


Enter an item name


Xray project


**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.


**Pipeline**
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.


**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

if you want to create a new item from other existing, you can use this option:

Copy from

OK

Build Steps

Build steps are the building blocks of the build process. These need to be defined in the project configuration.

The app provides one build step for exporting Cucumber Scenario/Scenario Outlines from Jira as .feature files, and one post-build action which publishes the execution results back to Jira, regardless of the build process status.



Please note

The fields of the tasks may take advantage of the Jenkins Environment variables, which can be used to populate fields such as the "Revision" for specifying the source code's revision. For more information, please see [Jenkins set environment variables](#).

Xray: Cucumber Features Export Task

This build step will export the Cucumber Tests (i.e., Scenario/Scenario Outlines) in .feature or bundled in a .zip file. The rules for exporting are defined [here](#).

It invokes Xray's Export Cucumber Tests REST API endpoint (see more information [here](#)).

Configuration

Some fields need to be configured in order to export the Cucumber Tests. As input, you can either specify issue keys (see the endpoint documentation [here](#)) or the ID of the saved filter in Jira.

| field | description |
|---------------|---|
| Jira instance | The Jira instance where Xray is running |
| Issue keys | Set of issue keys separated by ";" |
| Filter ID | A number that indicates the filter ID |

| | |
|-----------|---|
| File path | The relative path of the directory where the features should be exported to; normally, this corresponds to the "features" folder of the Cucumber project that has the implementation steps. Note: The directory will be created if it does not exist. |
|-----------|---|

Xray: Results Import Task

The app provides easy access to Xray's Import Execution Results REST API endpoints (see more information [here](#)). Therefore, it mimics the endpoints input parameters.


It supports importing results in Xray's own JSON format, Cucumber, Behave, JUnit, and NUnit, among others.

Configuration

| field | description |
|-----------------------|---|
| Jira instance | The Jira instance where Xray is running |
| Format | A list of test result formats and its specific endpoint |
| Execution Report File | The results relative file path Note: regex is not supported. |

Additional fields

Depending on the chose test result format and endpoint, some additional fields may need to be configured.

| format and specific endpoint | field | description |
|---|-----------------------|--|
| Behave JSON multipart Cucumber JSON multipart NUnit XML multipart JUnit XML multipart Robot XML multipart | Test execution fields | <p>An object (JSON) specifying the fields for the issue. You may specify the object either directly in the field or in the file path.</p> <div>  Learn more The custom field IDs can be obtained using the Jira REST API Browser tool included in Jira. Each ID is of the form "customfield_ID". Another option, which does not require Jira administration rights, is to invoke the "Get edit issue meta" in an existing issue (e.g., in a Test issue) as mentioned here. Example: GET http://yourserver/rest/api/2/issue/CALC-1/editmeta </div> |
| NUnit XML JUnit XML Robot XML | Project key | Key of the project where the Test Execution (if the Test Execution Key field wasn't provided) and the Tests (if they aren't created yet) are going to be created |
| | Test execution key | Key of the Test Execution |
| | Test plan key | Key of the Test Plan |
| | Test environments | List of Test Environments separated by "; |
| | Revision | Source code's revision being target by the Test Execution |
| | Fix version | The Fix Version associated with the test execution (it supports only one value) |

Examples

Cucumber

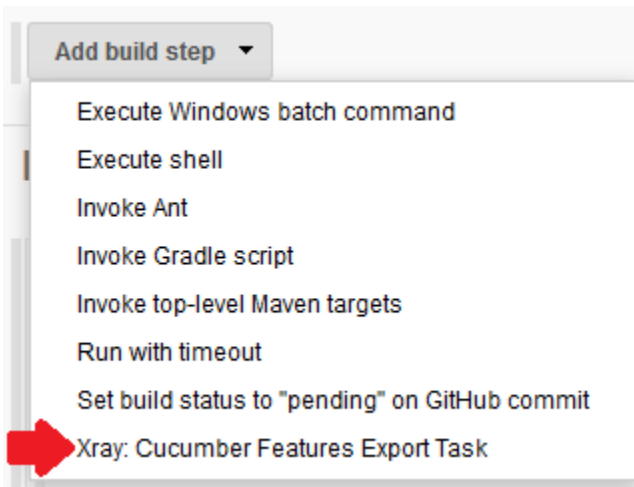
In a typical [Cucumber Workflow](#), after having created a Cucumber project and the Cucumber tests specified in Jira, you may want to have a project that **exports** the features from Jira, executes the automated tests on a CI environment and then **imports** back its results.

For this scenario, the Jenkins project would be configured with a set of tasks responsible for:

1. Pulling the Cucumber project
2. **Exporting Cucumber features from Jira to your Cucumber project**
3. Executing the tests in the CI environment
4. **Importing the execution results back to Jira**

Exporting Cucumber features

To start the configuration, add the build step *Xray: Cucumber Features Export Task*.



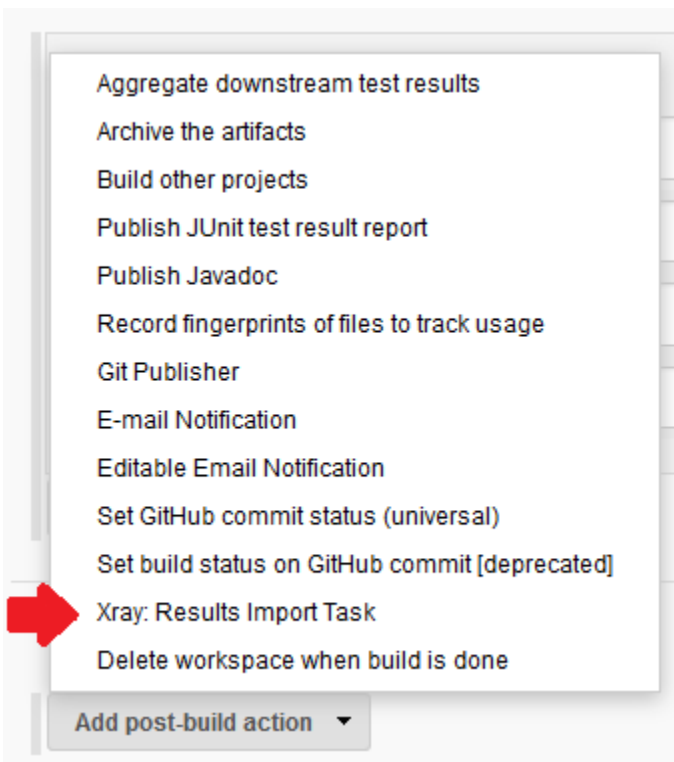
After that, configure it.

In this example, we configured the task to extract the *features* from a set of issues (PROJ-78 and PROJ-79) to the folder that holds the Cucumber project.



Importing the execution results

To start the configuration, add the post-build action *Xray: Results Import Task*.



After that, configure it.

In this example, we configured the task to import the **Cucumber JSON** results back to Jira.

Once all configurations are done, click Save at the bottom of the page.

After running the job, the expected result is a new Test Execution issue created in the Jira instance.

| Project: All ▾ Type: All ▾ Status: All ▾ Assignee: All ▾ Contains text More ▾ 🔍 Advanced | | | | | | | ⋮ ▾ |
|--|----------|-----------------------------------|---|--------|-----------|-----------|-----------|
| Created Date: Within the last... ▾ | | | | | | | |
| 1-1 of 1 📄 | | | | | | | Columns ▾ |
| T | Key | Summary | Tests association with a Test Execution | Status | Created ↓ | Updated | |
| 🟢 | PROJ-177 | Execution results [1489077439985] | PROJ-79 PROJ-78 | OPEN | 09/Mar/17 | 09/Mar/17 | ... |
| 1-1 of 1 📄 | | | | | | | |

Importing the execution results with user-defined field values

For Cucumber, Behave, JUnit, Nunit and Robot, Xray for Jenkins allows you to create new Test Executions and have control over newly-created Test Execution fields. You can send two files, the normal execution result file and a JSON file similar to the one Jira uses to create new issues. More details regarding how Jira creates new issues [here](#).

For this scenario and example, the import task needs to be configured with the **Cucumber JSON Multipart** format. When selecting this option, you can additionally configure the *Test Execution fields* in one of two ways:

- Insert the relative **path** to the JSON file containing the information, or
- Insert the **JSON content** directly in the field.

In this example, we configured the following object:

```
{
  "fields": {
    "project": {
      "key": "PROJ"
    },
    "summary": "Test Execution for Cucumber results (Generated by job: ${BUILD_TAG})",
    "issuetype": {
      "id": "10102"
    }
  }
}
```

And configured the task to import the **Cucumber JSON Multipart** results back to Jira.

Xray: Results Import Task

JIRA Instance: Xray local

Format: Cucumber JSON multipart

Parameters

Execution Report File (file path with file name): report.json

Test Execution fields: JSON Content

```
{
  "fields": {
    "project": {
      "key": "PROJ"
    },
    "summary": "Test Execution for Cucumber results (Generated by job: ${BUILD_TAG})",
    "issuetype": {
      "id": "10102"
    }
  }
}
```

Once all configurations are done, click Save at the bottom of the page.

After running the job, the expected result is a new Test Execution issue created in the Jira instance, with the Test Execution fields as specified in the Jenkins build step configuration.

Project: All Type: All Status: All Assignee: All Contains text More Advanced

Created Date: Within the last ...

1-1 of 1

| T | Key | Summary | Tests association with a Test Execution | Status | Created | Updated | Test Environments | Labels |
|---|----------|---|---|--------|-----------|-----------|-------------------|----------|
| ✓ | PROJ-479 | Test Execution for Cucumber results (Generated by job: jenkins-Xray Automated Tests-26) | PROJ-78 | OPEN | 04/Apr/17 | 04/Apr/17 | None | None ... |

JUnit

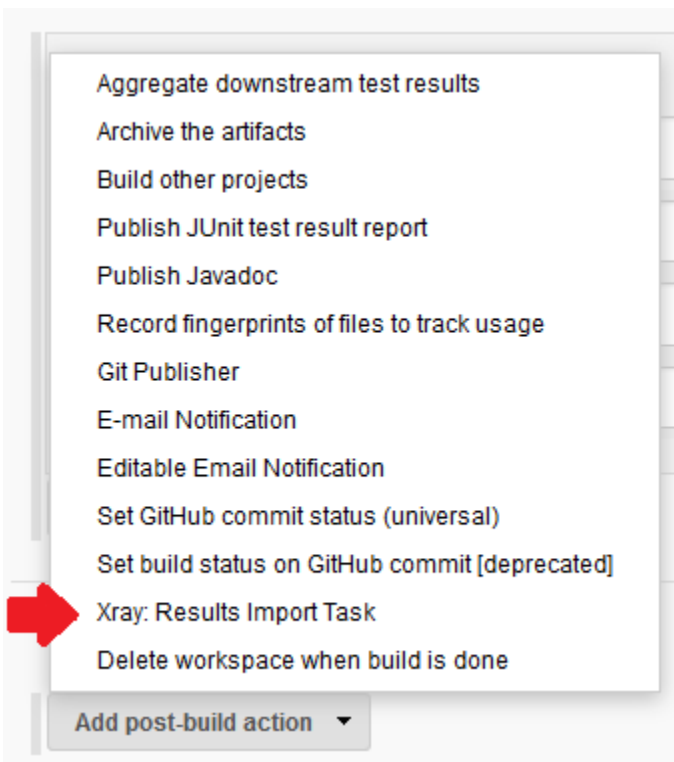
Apart from supporting Cucumber natively, Xray enables you to take advantage of many other testing frameworks like JUnit. In this sense, Xray for Jenkins lets you import results in other formats besides Cucumber JSON.

If you want to import **JUnit XML reports**, a typical Job outline would be:

1. Pulling the JUnit project
2. Executing the tests in the CI environment
3. **Importing the execution results, including Tests, to JIRA**

Importing the execution results

To start the configuration, add the post-build action *Xray: Results Import Task*.



After that, configure it.

In this example, we have a configuration where the **JUnit XML** format is chosen.

After running the plan, the expected result is a new Test Execution issue created in the JIRA instance.

| T | Key | Summary | Tests association with a Test Execution | Status | Created | Updated | Test Environments |
|--------|----------|--|---|--------|-----------|-----------|-------------------------|
| [icon] | PROJ-185 | Execution results - TestResult.xml - [1489165846959] | PROJ-121 | OPEN | 10/Mar/17 | 10/Mar/17 | Android Cordova IOS |

Troubleshooting

*The build process is failing with status code **403***

When you check the log, it has the following:


Console Output

```
Started by user admin
Building in workspace C:\Users\DMDU\.jenkins\workspace\Xray Automated Tests
Starting export task...
#####
###   Xray for JIRA is exporting the feature files   ###
#####
PROJ-78;PROJ-79
Task failed
➡ ERROR: Unable to confirm Result of the download..... Download Failed! Status:403 Response:
```

By default, when you successively try to log into Jira with the wrong credentials, the Jira instance will prompt you to provide a CAPTCHA the next time you try to log in. It is not possible to provide this information via the build process, so it will fail with status code **403 Forbidden**.

You will need to log into Jira via the browser and provide the CAPTCHA.

If you are a Jira administrator, you can go to Jira administration > User Management and reset the failed login.

| | | | | | | |
|---|-----------------------------|---|---------------------|---------------|-------------------------|----------|
|  CI_User | CI_User user@example.com | Count: 9 Last: Today 1:55 PM CAPTCHA required at next login Last failed login: Today 1:57 PM Current failed logins: 7 Total failed logins: 21 ➡ Reset failed login count | jira-software-users | JIRA Software | JIRA Internal Directory | Edit ... |
|---|-----------------------------|---|---------------------|---------------|-------------------------|----------|