Integration with Maven

- List of Available Properties
  - List of Available Result Formats
- Uploading the results to a specific endpoint
- Maven Project Guidelines
  - Settings.xml
  - POM.xml - Simple Project
    - Importing the execution results with user-defined field values
  - POM.xml - Multi-Module Project
- Troubleshooting
  - Unable to upload the results file: The process is failing with status code 403
- Version History

Maven repository URL change

Starting in version 1.1.1, beware that the repository URL changed to https://maven.getxray.app:443

You must update the repository to the following:

```
<pluginRepositories>
...
<pluginRepository>
                <id>xblend-repo-releases-public</id>
                <url>https://maven.getxray.app:443/artifactory/releases</url>
        </pluginRepository>
</pluginRepositories>
```

In order to make the integration in your Java Maven projects easier, we have developed the ***Xray-Maven-Plugin.*** It enables the upload of your Tests directly to Xray with a single Maven command.

The plugin supports single or multi-module projects and multiple test frameworks. With the execution of a single Maven target ***com.xpandit.xray:xray-maven-plugin:<goal-name>,*** the plugin will upload the results of the tests to your Jira instance.

List of Available Properties

Please note

The user present in the properties below must exist in the JIRA instance and have permission to Create Test and Test Execution Issues

Property Name	Required (✓-Yes, ?-Optional)	Description	Examples
`xray.jiraURL`	✓	Full URL of Jira, including the relative path, if its being used	`http://localhost:8080`

xray.resultsFormat	✅	Results format. Available formats: • JUNIT • TESTNG (more on format specifics in Import Execution Results - REST) ⚠ case sensitive	`JUNIT`
xray.username	✅	Jira Username	`admin`
xray.password	✅	Jira Password ⚠ plain-text only	`password`
xray.projectKey	✅	Jira Project Key	MYPROJKEY
xray.testExecKey	❓	Test Execution Issue ID	MYT2-5
xray.testPlanKey	❓	Test Plan Issue ID	MYT2-54
xray.testEnvironments	❓	Test Environment	iOS
xray.fixVersion	❓	Fix Version. It supports only one value.	Release 1.0
xray.revision	❓	Revision	1234
xray.testExec-fields.path	✅ (used only for multipart endpoints)	Path to the JSON info file. Used by JIRA to create new issues.	`${basedir}/reports/info.json`

xray.surefire.location	✅	Folder with .xml files or xml file to be uploaded	`${basedir}/target/surefire-reports/TEST-com.xpandit.xray.service.ResultImporterTest.xml`
			
			`${basedir}/target/surefire-reports`

Please note

You may configure the plugin properties in your project pom.xml, the settings.xml file (see project guidelines below) or via the console with -D option (e.g., -Dxray.projectKey=PROJ).

List of Available Result Formats

Each result format points to a specific Xray REST Endpoint where the results are imported. The result format is configured in the property **xray.resultsFormat** and is **case-sensitive**.

Results Format (xray.resultsFormat)	Description	Support for multipart endpoint
JUNIT	JUnit XML output format (more information regarding its endpoints here)	✅
TESTNG	TestNG XML output format (more information regarding its endpoints here)	✅

Uploading the results to a specific endpoint

Xray-Maven-Plugin provides two plugin specific goals to upload the results to a specific endpoint:

- **xray**: uploads the results file to the normal endpoint;
- **xray_multipart**: uploads the results file to the multipart endpoint;

The goal name must then be specified in maven command so the plugin uploads the results file to the selected Xray endpoint:

- **Build, Test the project and then Upload results**

```
mvn clean package surefire:test com.xpandit.xray:xray-maven-plugin:<goal-name>
```

Or if the project is already built and tested, if you desire only to upload the results:

- **Upload results Only**

```
mvn com.xpandit.xray:xray-maven-plugin:<goal-name>
```

Maven Project Guidelines

There are three options when configuring the properties listed in the 'List of Available Properties': via settings.xml, via the Project's POM file or via the console with -D option (e.g., -Dxray.projectKey=PROJ).

To understand better how the configurations via settings.xml and the Project POM can be combined, we will be describing an example where we want to import Junit results into Jira. We will assume the Test Cases are already implemented so we only focus on the usage of the this plugin.

In sum the goal is to:

- Generate the Junit XML results file;
- Upload the results to a JIRA instance with a specific address, based on a specific credential info.
- Associate the results to a new Test Execution with Fix Version 'V1.0', in the Test Environment 'Android', in an existing Test Plan issue;

Settings.xml

The plugin properties can be specified in the settings.xml file. In this example, in the <*properties*> tag, we configured the address of the JIRA instance and the credentials of the JIRA user. These properties will be within the Active Profile identified by the id 'MyActiveProfile'.

~/.m2/settings.xml

```
<settings xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.
xsd" xmlns="http://maven.apache.org/SETTINGS/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- OTHER CONFS-->
  <profiles>
    <profile>
        <!-- OTHER CONFS-->
      <properties>
        <xray.jiraURL>http://localhost:8080</xray.jiraURL>
        <xray.username>admin</xray.username>
        <xray.password>password</xray.password>
      </properties>
    <id>MyActiveProfile</id>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>MyActiveProfile</activeProfile>
  </activeProfiles>
</settings>
```

POM.xml - Simple Project

The plugin properties listed in 'List of Available Properties' can be scoped to a specific Project, by specifying them in the Project's POM.xml.

Since, in this example, we already defined the JIRA address and the credentials in the *settings.xml*, when Maven loads the Project's POM, it will pick up the activated profiles from the *<activeProfiles>* section of the settings.xml file and inject the properties declared within the profile.

So, in this example, we only need to specify the JIRA Project, the Test Plan issue we want to upload the results into and the field values (Test Environment, Fix version and Revision) for the newly created Test Execution.

```
<xray.projectKey>MYPROJKEY</xray.projectKey>
<xray.testPlanKey>MYPROJKEY-124</xray.testPlanKey>
<xray.testEnvironments>Android<xray.testEnvironments>
<xray.fixVersion>V1.0</xray.fixVersion>
<xray.revision>Build-1.0</xray.revision>
<xray.resultsFormat>JUNIT</xray.resultsFormat>
<xray.surefire.location>${basedir}/target/surefire-reports</xray.surefire.location>
```

These properties will instruct the plugin:

- To upload the results to project with key 'MYPROJECTKEY'
- The newly created Test Execution will be associated with the Test Plan with issue key 'MYPROJECTKEY-124'
- The Test Execution will be associated with the Test Environment 'Android'
- The Test Execution's Fix Version custom field will be populated with V1.0
- The Test Executions Revision custom field will be populated with the Revision 'Build-1.0'
- The results file we are uploading is in Junit format
- The results file can be found in this directory: *${basedir}/target/surefire-reports*

Please note

If the goal was also to upload the results to an existing Test Execution issue, then we should configure the *xray.testExecKey* property by specifying the Test Execution issue key.

These properties will then be added to the POM file as illustrated below.

Please note

This example also describes the usage of the plugin on a standard simple, single-module Project. Note that we declared the plugin inside the *<re porting>* tag, wherein we identify which version to be used. The tag *<pluginRepositories>* specifies where the plugin should be downloaded from.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/maven-v4_0_0.xsd">
    <!-- CONF's -->
    <properties>

        <!--XRay Properties -->

        <!--IN PROFILE ~.m2/settings.xml-->

        <!-- If we want to reuse the configured credentials in settings.xml, then we do not need to configure
them here
                <xray.jiraURL></xray.jiraURL>
        <xray.resultsFormat></xray.resultsFormat>
        <xray.username></xray.username>
        <xray.password></xray.password>
        -->
        <xray.projectKey>MYPROJKEY</xray.projectKey>
        <xray.testPlanKey>MYPROJKEY-124</xray.testPlanKey>
        <xray.testEnvironments>Android</xray.testEnvironments>
        <xray.fixVersion>V1.0</xray.fixVersion>
```

```
        <xray.revision>Build-1.0</xray.revision>
        <xray.resultsFormat>JUNIT</xray.resultsFormat>
        <xray.surefire.location>${basedir}/target/surefire-reports</xray.surefire.location>
        <!--End Xray Properties -->

    </properties>

    <build>
                <pluginManagement>
                <plugins>
             <plugin>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-surefire-plugin</artifactId>
                  <version>2.19.1</version>
                  <configuration>
                      <testFailureIgnore>true</testFailureIgnore>
                  </configuration>
             </plugin>
         </plugins>
          </pluginManagement>
    </build>

    <reporting>
        <plugins>
            <plugin>
                <artifactId>maven-surefire-report-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>com.xpandit.xray</groupId>
                <artifactId>xray-maven-plugin</artifactId>
                <version>1.1.2</version>
            </plugin>
        </plugins>
    </reporting>

    <pluginRepositories>
        <pluginRepository>
            <id>xpand-plugins</id>
            <name>xpand-plugins</name>
            <url>https://maven.getxray.app:443/artifactory/releases</url>
        </pluginRepository>
    </pluginRepositories>

</project>
```

To build, test the Project and upload the results to Jira, we need to execute the command as explained previously in 'Uploading the results to a specific endpoint'.

In this example the goal is to push the results to the normal endpoint, so we choose the 'xray' goal name:

```
mvn clean package surefire:test com.xpandit.xray:xray-maven-plugin:xray
```

Importing the execution results with user-defined field values

If we wanted to upload the Junit results to a newly created Test Execution Issue and having control over its fields, for instance by creating it with a custom Issue Summary, then the option is to use the multipart endpoint.

In addition to the *xray.resultsFormat* and the *xray.surefire.location* we'll need to configure a JSON file which holds some metadata (used by JIRA to create new issues) and save it in a directory accessible by the plugin. More details regarding how Jira creates new issues here.

In this example, we configured the following object:

```
{
  "fields": {
    "project": {
      "key": "MYPROJECTKEY"
    },
    "summary": "Test Execution for Junit results",
    "issuetype": {
```

```
      "id": "10007"
    }
  }
}
```

And in the Project's POM the properties we need to configure are the following:

```
<xray.resultsFormat>JUNIT</xray.resultsFormat>
<xray.surefire.location>${basedir}/target/surefire-reports</xray.surefire.location>
<xray.testExec-fields.path>${basedir}/info.json</xray.testExec-field.path>
```

These properties will instruct the plugin:

- The results file we are uploading is in Junit format
- The results file can be found in this directory: *${basedir}/target/surefire-reports*
- The JSON metadata file can be found in this directory: *${basedir}/info.json*

Lastly, since the goal is to push the results to the multipart endpoint, we run the command by choosing the 'xray_multipart' as the goal name:

```
mvn clean package surefire:test com.xpandit.xray:xray-maven-plugin:xray_multipart
```

POM.xml - Multi-Module Project

The Xray-Maven-Plugin can also be used with Multi-Module Projects.

This example describes the usage of the plugin on a Module Project, in this case just one sub-module.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/maven-v4_0_0.xsd">
    <!-- CONF's -->
        <modules>
            <module>ModuleA</module>
        </modules>
    <properties>

        <!--XRay Properties -->

       <!-- If we want to reuse the configured credentials in settings.xml, then we do not need to configure
them here
                <xray.jiraURL></xray.jiraURL>
        <xray.resultsFormat></xray.resultsFormat>
        <xray.username></xray.username>
        <xray.password></xray.password>
           -->
        <xray.projectKey>MYPROJKEY</xray.projectKey>
        <xray.testPlanKey>MYPROJKEY-124</xray.testPlanKey>
        <xray.testEnvironments>Android</xray.testEnvironments>
        <xray.fixVersion>V1.0</xray.fixVersion>
        <xray.revision>Build-1.0</xray.revision>
        <xray.resultsFormat>JUNIT</xray.resultsFormat>
        <xray.surefire.location>${basedir}/target/surefire-reports</xray.surefire.location>

        <!--End Xray Properties -->

    </properties>

    <build>
                <pluginManagement>
                        <plugins>
                    <plugin>
                          <groupId>org.apache.maven.plugins</groupId>
```

```
                        <artifactId>maven-surefire-plugin</artifactId>
                        <version>2.19.1</version>
                        <configuration>
                            <testFailureIgnore>true</testFailureIgnore>
                        </configuration>
                    </plugin>
                </plugins>
                </pluginManagement>
    </build>

    <reporting>
        <plugins>
            <plugin>
                <artifactId>maven-surefire-report-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>com.xpandit.xray</groupId>
                <artifactId>xray-maven-plugin</artifactId>
                <version>1.0.0</version>
            </plugin>
        </plugins>
    </reporting>

    <pluginRepositories>
        <pluginRepository>
            <id>xpand-plugins</id>
            <name>xpand-plugins</name>
            <url>https://maven.getxray.app:443/artifactory/releases</url>
        </pluginRepository>
    </pluginRepositories>

</project>
```

ModuleA.pom

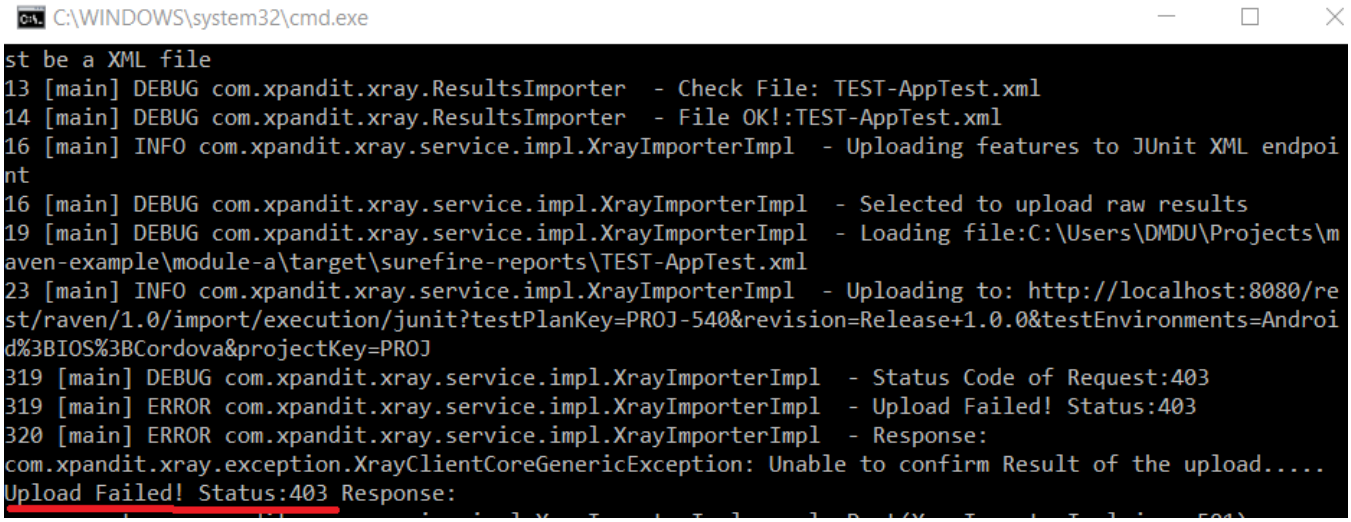
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://maven.apache.org/POM/4.0.0
                            http://maven.apache.org/maven-v4_0_0.xsd">
    <!-- CONF's -->
    <name>Module A</name>

    <properties>
        <!--XRay Properties -->
        <xray.surefire.location>${basedir}/target/surefire-reports</xray.surefire.location>
        <!--End Xray Properties -->
    </properties>
</project>
```

Troubleshooting

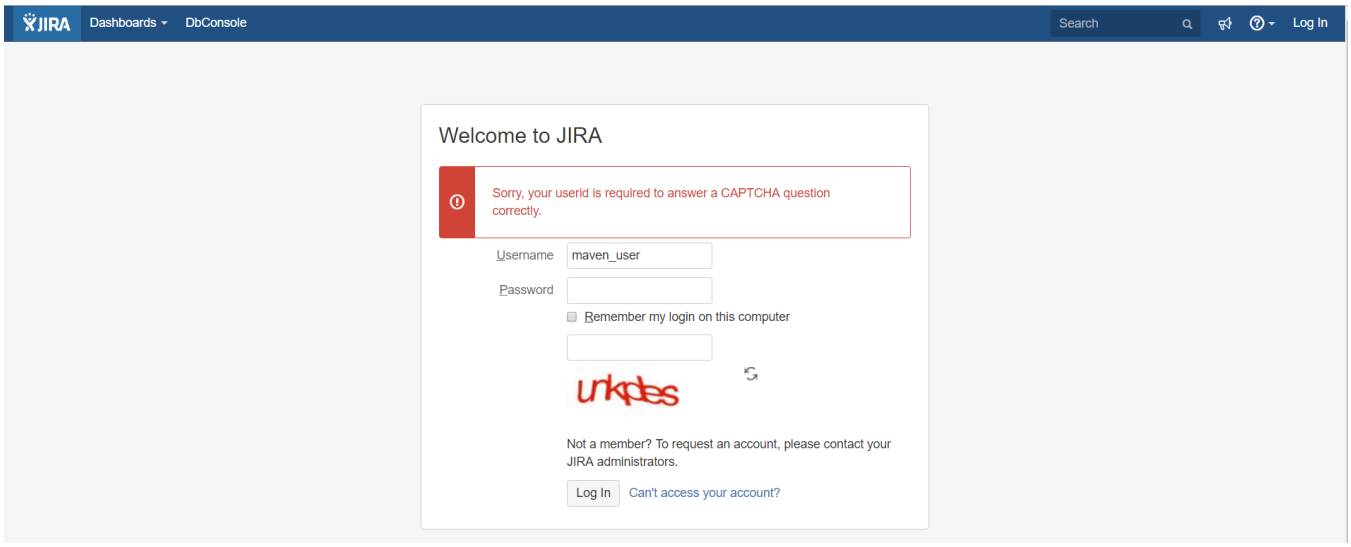
Unable to upload the results file: The process is failing with status code 403

The importing of the execution results file failed and when you check the log, it shows the following:

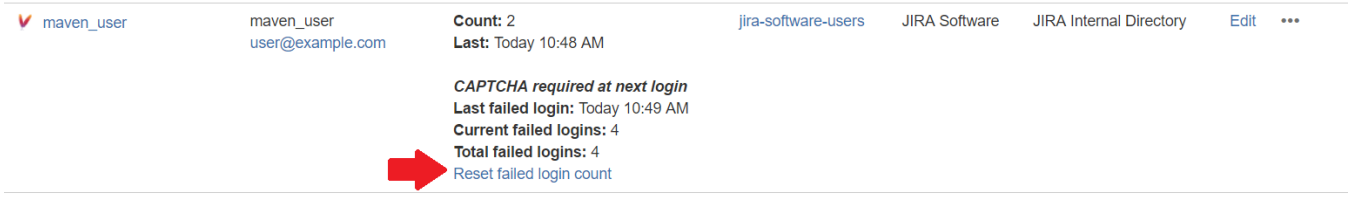


By default, when you successively try to log into Jira with the wrong credentials, the Jira instance will prompt you to provide a CAPTCHA the next time you try to log in. It is not possible to provide this information via the Maven project configuration, so it will fail with status code **403 Forbidden**.

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.



Version History

- 1.0.0 - 9 May 2017
  - Import JUnit results: You can import the JUnit results to Xray either as an XML file or a folder with multiple XML files.
- 1.1.0 - xx xx 2019
  - Import TestNG results: You can import the TestNG results to Xray either as an XML file or a folder with multiple XML files;
  - Support for JUnit and TestNG multipart endpoints.
- 1.1.1 - 17 December 2021
  - Bump log4j to version 2.16.0 due to critical security vulnerabilities:
    - https://nvd.nist.gov/vuln/detail/CVE-2021-44228
    - https://nvd.nist.gov/vuln/detail/CVE-2021-45046

        - https://nvd.nist.gov/vuln/detail/CVE-2019-17571
- 1.1.2 - 20 December 2021
    - Bump log4j to version 2.17.0 due to critical security vulnerability:
        - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-45105