Integrating with Testing Frameworks

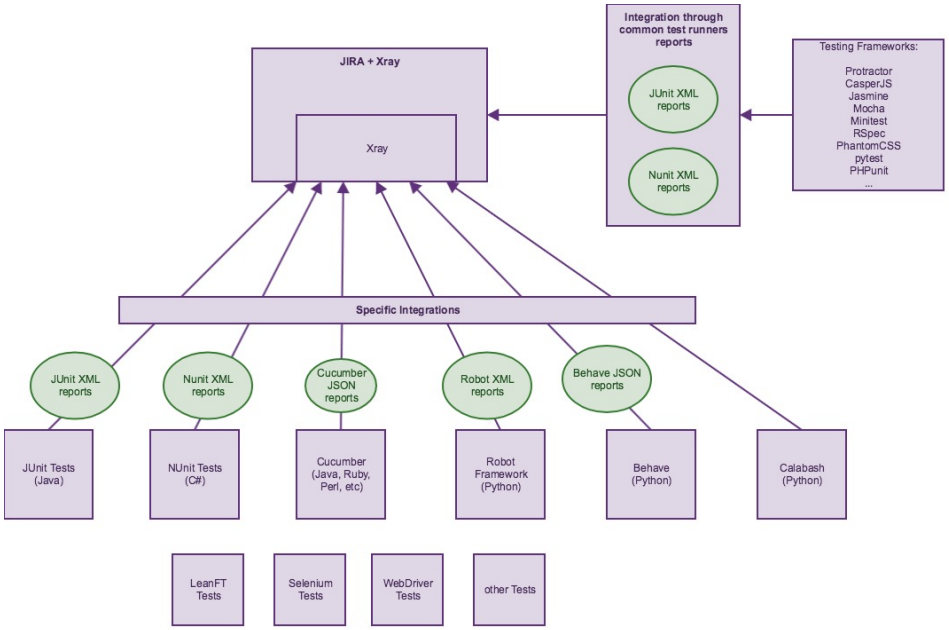
Integration with testing frameworks is achieved by processing the results contained in the reports test runners (e.g., in XML, JSON). The results are mapped to the proper Test issues; if they don't exist then they will be created. This flow is explained in Using Generic Tests for Automation. Xray also supports Cucumber tests natively.

Besides Cucumber, there are currently many testing frameworks available for every language you may think of.

Many of these frameworks provide test runners that are able to output reports in the JUnit XML format. In the "worst" case, your automated tests can be imported to Jira and mapped to "Generic" Test issues by importing JUnit XML test result reports, as detailed in Taking advantage of JUnit XML reports. Note that the JUnit report format is rather limited and is not supported by some testing frameworks.

Another way of integrating with testing frameworks is by using the NUnit test runner's XML report format. The NUnit report format has more features than JUnit 4.0 XML report, which allows Xray to do things such as automatic linking to issues (e.g., requirements) or assignment of labels to the newly created Test entities.

Besides this, Xray also provides specific integrations for Cucumber, Behave, Robot, Xamarin, and other frameworks.



Summary of features per framework

The following table presents the available features when importing automated test results.

The Xray JSON format is more generic and its capabilities, if used for importing, are different.

	Robot framework	JUnit 4	NUnit 2.6/3.x	Cucumber	Xray JSON
Abstract automated test as a Test *(map an automated test to a Test issue in Jira)*	Yes (as a Generic Test)	Yes (as a Generic Test)	Yes (as a Generic Test)	Yes (as a Cucumber Test)	No Tests must exist beforehand
Make Test specification in Jira *(specify the Test itself in Jira)*	No	No	No	Yes	Yes
Create Tests from results *(create Tests whenever importing results)*	Yes	Yes	Yes	No* (but there is and endpoint for importing Cucumber features, which will create Tests for the respective Scenarios/Scenario Outlines)	No
Uniquely identify Tests *(identify existing Tests whenever importing results, avoiding duplication of Test issues)*	Yes	Yes	Yes	Yes	Yes* (based on the provided Test issue keys)
Import results *(importing results by REST API or UI)*	Yes	Yes	Yes	Yes	Yes

Import “labels” *(create labels in the Test issues)*	Yes (labels may be specified in the test's source code)	No	Yes (labels may be specified in the test's source code)	Yes* (this is only available when using the endpoint for importing Cucumber features; it's not possible when importing results)	No
Automatic linking to requirements *(create links to requirements)*	Yes (requirement's issue key may be specified in the test's source code)	No	Yes (requirement's issue key may be specified in the test's source code)	Yes* (this is only available when using the endpoint for importing Cucumber features; it's not possible when importing results)	No
Semantic on the results *(present execution details in the execution screen)*	Yes (keyword “steps”)	-	Yes (Test Suites and parameterized Tests)	Yes (steps for Scenario/Scenario Outline and Background)	Yes* (semantic is implicit, because the Test must be created beforehand)

Integration by platform

blocked URL	C#	C/C++	JS	**blocked URL**	**blocked URL**	**blocked URL**	**blocked URL**
• Load Testing using Gatling and JUnit in Scala • Testing Android Applications using Appium and JUnit in Java • Testing iOS applications using Appium and JUnit in Java • Testing using Cucumber in Java • Testing using Cucumber in Ruby/JRuby • Testing using Robot framework and xUnit reports • Testing using Robot Framework integration in Python or Java • Testing using Robot framework integration, in Python or Java	• Testing using NUnit in C# • Testing using Selenium (WebDriver) and Nunit in C# • Testing using Specflow and NUnit in C# • Testing using UFT Pro (LeanFT) and NUnit in C#	• Testing using Google Test in C++	• Testing using CasperJS and PhantomJS in JavaScript • Testing using Jasmine (Node.js) in JavaScript • Testing using Mocha in JavaScript • Visual Testing using PhantomCSS in JavaScript	• Testing using Cucumber in Ruby/JRuby • Testing using MiniTest in Ruby • Testing using Selenium and RSpec in Ruby	• Testing using PHPUnit in PHP	• Testing using Behave in Python • Testing using pytest in Python • Testing using Robot framework and xUnit reports • Testing using Robot Framework integration in Python or Java • Testing using Robot framework integration, in Python or Java • Testing using the Robot Framework and xUnit reports	• Testing using Cucumber in Perl

• Testing using Selenium and JUnit in Java • Testing using the Robot Framework and xUnit reports • Testing using UFT Pro (LeanFT) and JUnit in Java • Testing Windows Applications using Appium and JUnit in Java							

Integration by testing framework report

blocked URL		**blocked URL**
• Load Testing using Gatling and JUnit in Scala • Taking advantage of JUnit XML reports • Testing Android Applications using Appium and JUnit in Java • Testing Angular apps using Protractor in JavaScript • Testing iOS applications using Appium and JUnit in Java • Testing using CasperJS and PhantomJS in JavaScript • Testing using Google Test in C++ • Testing using Jasmine (Node.js) in JavaScript • Testing using MiniTest in Ruby • Testing using Mocha in JavaScript • Testing using PHPUnit in PHP • Testing using pytest in Python • Testing using Robot framework and xUnit reports • Testing using Selenium and JUnit in Java • Testing using Selenium and RSpec in Ruby	• Taking advantage of NUnit XML reports • Testing using NUnit in C# • Testing using Selenium (WebDriver) and Nunit in C# • Testing using Specflow and NUnit in C# • Testing using UFT Pro (LeanFT) and NUnit in C#	• Testing using Calabash and Xamarin Test Cloud in Ruby • Testing using Cucumber in Java • Testing using Cucumber in Perl • Testing using Cucumber in Ruby/JRuby

Testing by target environment

Browser testing	**Mobile Testing**	**Testing in the Cloud**

• Testing using CasperJS and PhantomJS in JavaScript • Testing using Selenium (WebDriver) and Nunit in C# • Testing using Selenium and JUnit in Java • Testing using Selenium and RSpec in Ruby • Visual Testing using PhantomCSS in JavaScript	• Testing using Calabash and Xamarin Test Cloud in Ruby	• Testing using Calabash and Xamarin Test Cloud in Ruby