Polarion Software®

USER GUIDE

Automated Test Execution with Polarion

Europe, Middle-East, Africa: Polarion Software GmbH
Hedelfinger Straße 60 — 70327 Stuttgart, GERMANY
Tel +49 711 489 9969 - 0
Fax +49 711 489 9969 - 20
www.polarion.com - info@polarion.com

Americas & Asia-Pacific: Polarion Software, Inc.
1001 Marina Village Parkway, Suite 403, Alameda, CA 94501, USA
Tel +1 877 572 4005
Fax +1 510 814 9983
www.polarion.com - info@polarion.com

Introduction

Polarion® QA™ provides an integrated test management solution that manages requirements, tests, defects in one environment, with complete traceability from inception to completion.

This guide outlines how to set-up an environment that will kick off your test automation scripts directly from Polarion and all the test execution results will be pulled right back in to help you see relevant reports and metrics. You may run your test scripts right away or schedule them for a time in the future through a simple and user friendly interface.

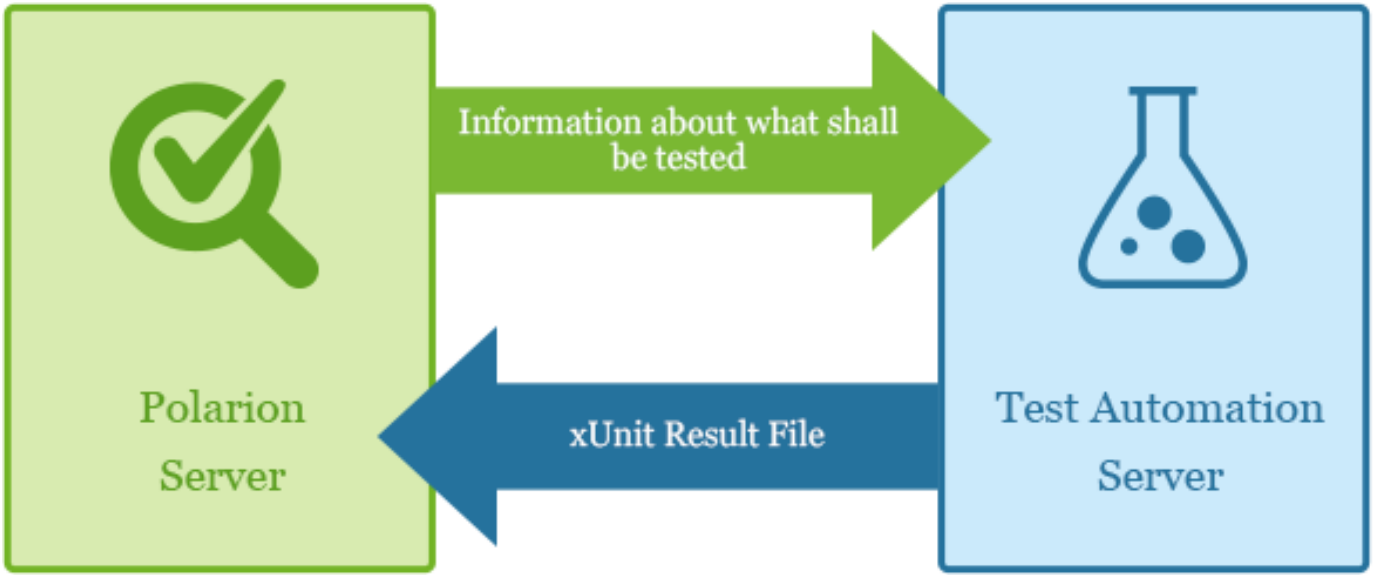
When using Polarion QA for such a scenario, the invoked tool and its role must be clearly defined. In addition to Polarion QA, a tool for automated test execution must be available. The role of the automated test execution tool is to receive a set of data, representing what will be tested and how it shall be tested. Polarion typically holds all of this information and is therefore responsible to somehow deliver the information.

With the information in this document, you will be able to simulate an automatic test run execution and to import results back to Polarion as test runs.

The first section describes the basic configuration between Polarion QA and a Test Automation server and outlines how to invoke a build tool to enable exchange of test data with automated tests using a variety of commercial and open-source test automation engines.

The subsequent section provides a real world example of Polarion working with Jenkins continuous integration server.

Basic Setup



The picture shows the basic concept of data exchange between a Polarion instance and a Test Automation Server (TAS). For most customers, both servers are separate machines. It is important that the TAS contains some kind of “Test Agent” who receives incoming requests from the initiating tool. This is important because test execution often invokes complex third party tools, which must be available in the execution environment. The Test Agent is the part which has to be called from Polarion and which has to be filled with required information.

Invoke a Build

In order to start the process of sharing test information between Polarion QA and a Test Automation server, we must invoke a build process using one of the following tools:

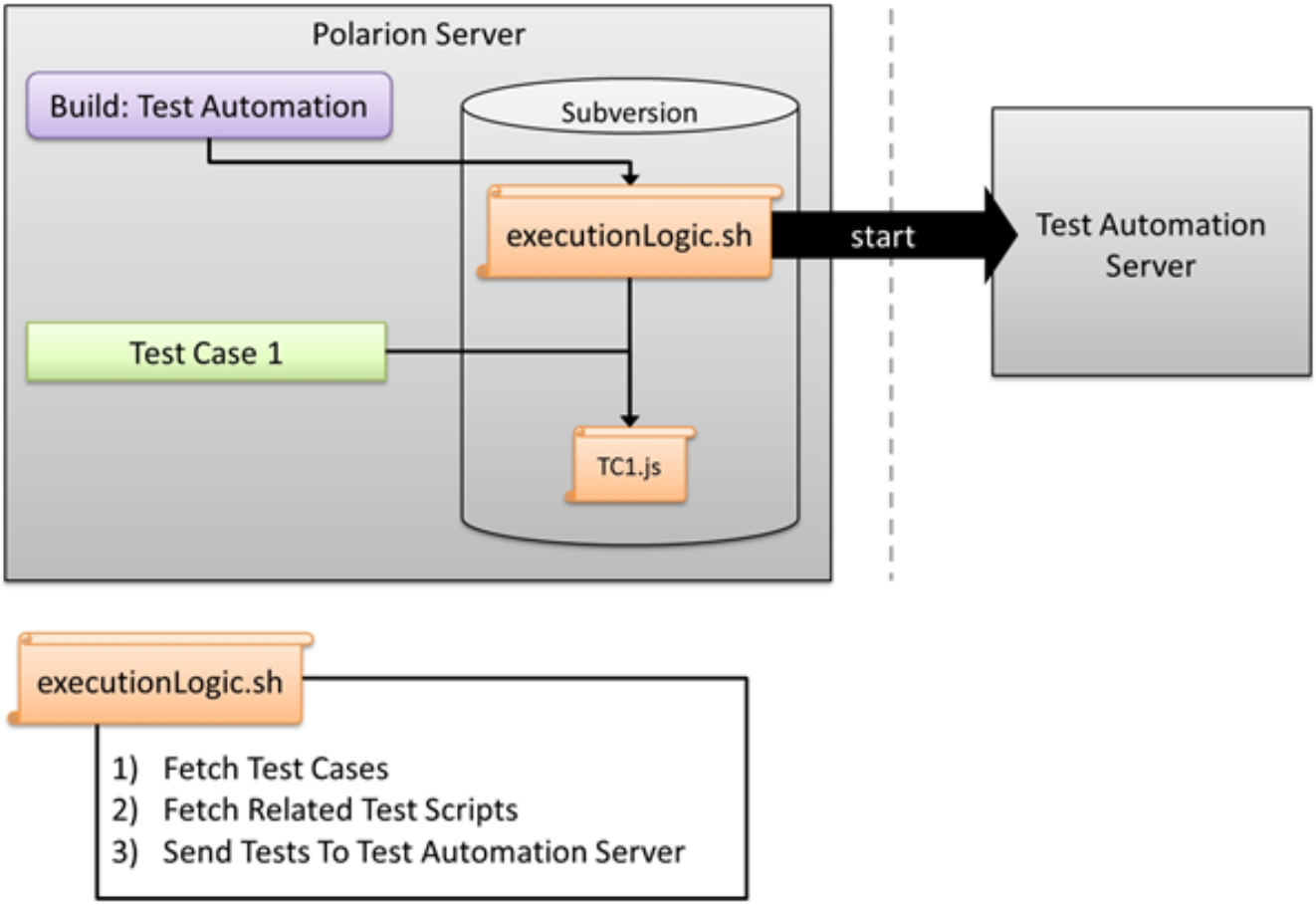
- Jenkins Integration Server
- Hudson
- ANT
- MAVEN
- Polarion Build

Exchange of Test Data

Once we have invoked the appropriate build tool, we will need to define and execute the appropriate scripts for the following:

1. Fetching Test Cases from Polarion QA for what is to be tested
2. Fetching Related Test Scripts for execution
3. Sending Tests to Test Automation Server

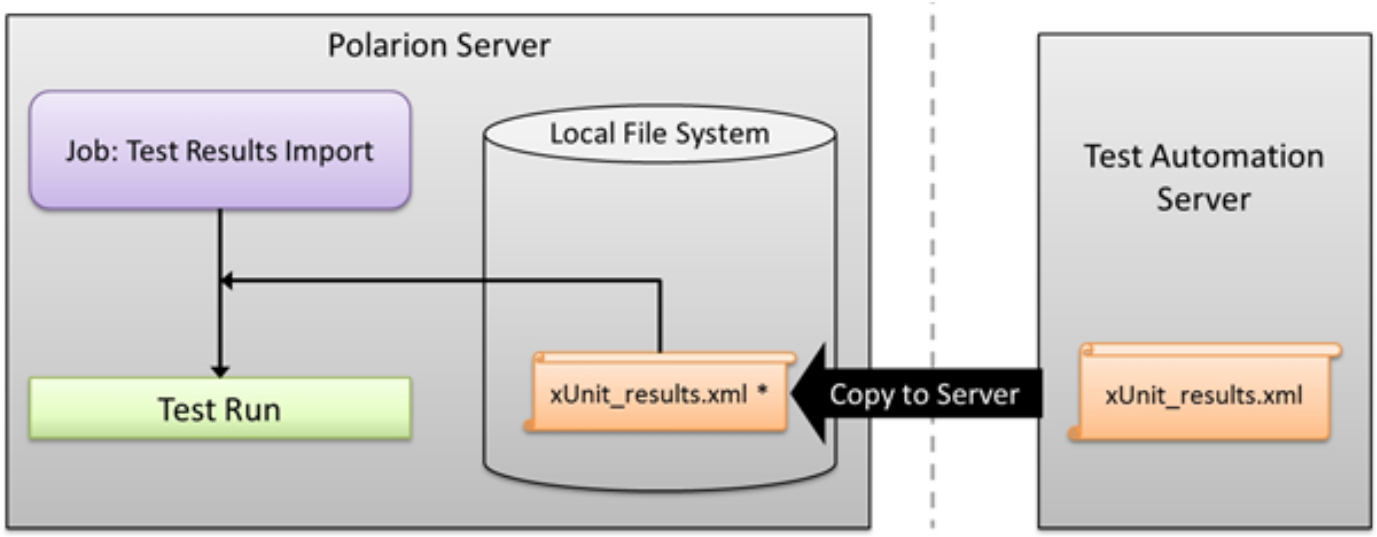
Triggering Test Execution



Receiving Results

Once automated tests are completed, the results must somehow be transferred back to Polarion QA and the appropriate test cases updated.

Overview "Fetching Results"



The picture shows the required architecture for the reimporting of test results. One can see the file "xUnit_results.xml" is originally created on the TAS in a valid xUnit format. The TAS has finished submitting the results to any location Polarion can access. This may be a network share or just a folder on the Polarion server's machine. There is a separate job, which is commonly run every few minutes or hours. The job checks if new xml results are available in the specified location.

The job will successfully create a new Test Run for every valid xml file found in the specified directory. At the end, the containing xml file(s) will be deleted by the job.

Test Execution

The execution itself is responsibility of the TAS. After Polarion has supplied the test data and started the test, it will just wait for results. The results themselves are generated by the TAS and must be copied to a local or network location, which Polarion can access, after each test execution.

Example using Jenkins - How to integrate Polarion with Jenkins to fetch test cases, and import test results automatically

1

Jenkins must be running on a specified server @ http://myjenkins

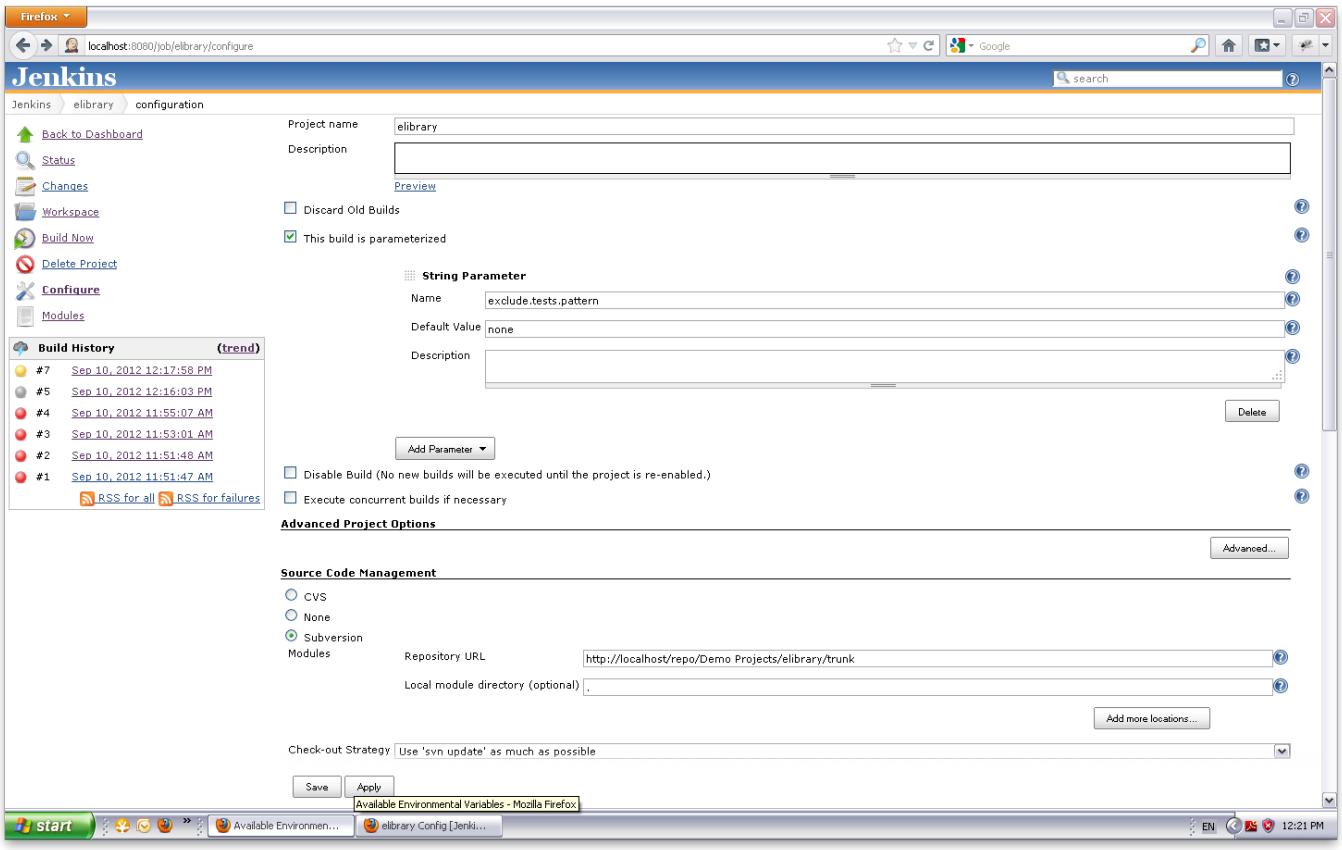
Polarion must be running on a specified server @ http://mypolarion

2

You will set-up a specific Polarion project to host all Polarion artifacts for some project. In this example we will work with the standard elibrary project and we will use a Jenkins project to host the build automation procedure, build the E-Library, and import the test results executed by Jenkins into Polarion QA.

See: subversion url pointing to project url

Note: the E-Library build process is parametrized by exclude-test-pattern variable, typically you will not have it, but in case of elibrary you need to pass the variable

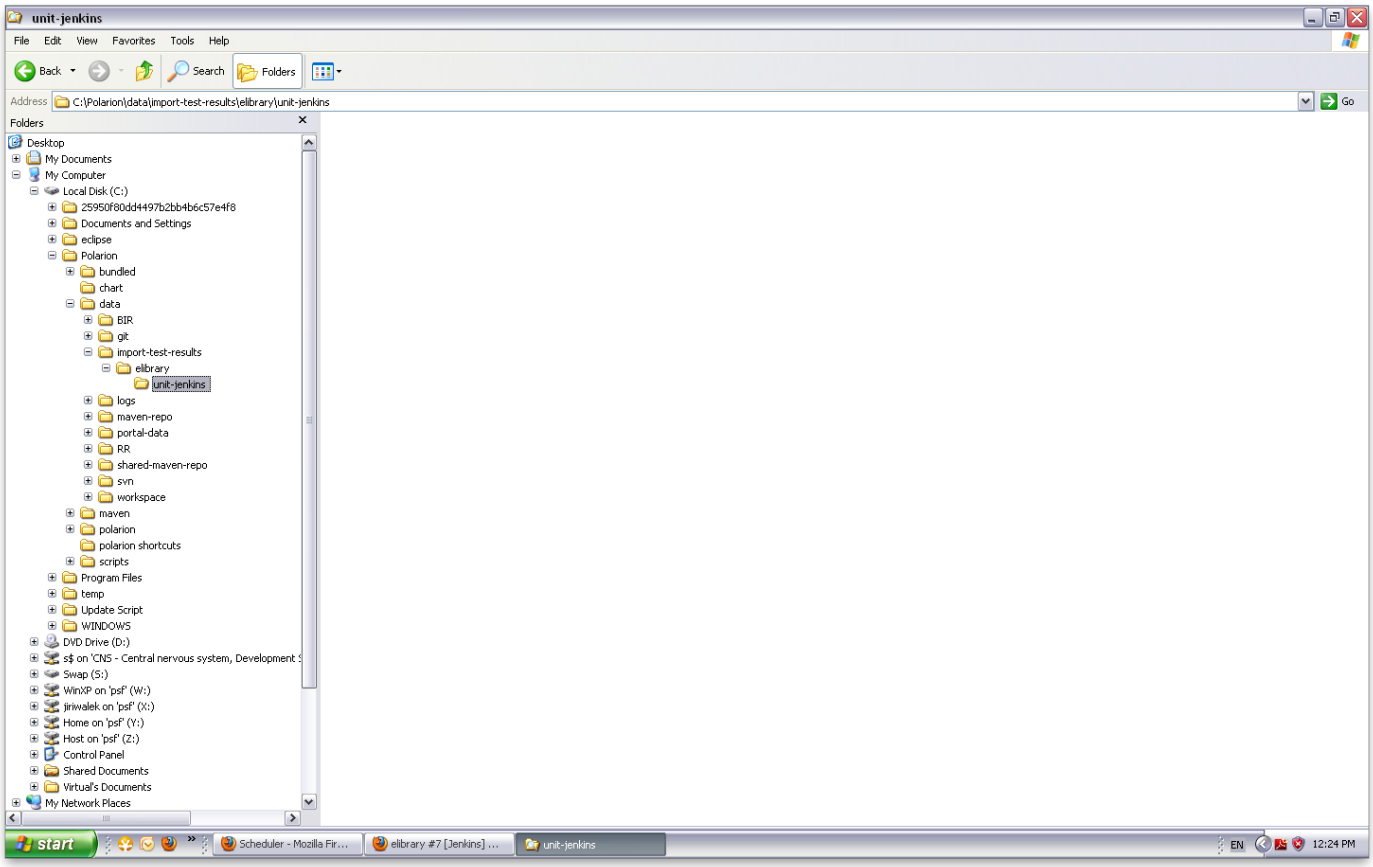


3

Next we must set-up configuration for importing test results to Polarion QA

I) create a folder on Polarion server to import test results:

```
c:/polarion/data/import-test-results/elibrary/jenkins-unit
```



II) Polarion server must be configured to watch the folder and import the test-results when they appear in the project.

The following job is created for this process:

```
<job id="xUnitFileImport" name="Import Elibrary Tests Results" scope="system">
   <path>C:\Polarion\data\import-test-results\elibrary\unit-jenkins</path>
   <project>elibrary</project>
   <userAccountVaultKey>xUnitFileImportUser</userAccountVaultKey>
   <maxCreatedDefects>10</maxCreatedDefects>
   <maxCreatedDefectsPercent>5</maxCreatedDefectsPercent>
   <templateTestRunId>JUnit Build Test</templateTestRunId>
   <idRegex>(.*).xml</idRegex>
   <groupIdRegex>(.*)_.*.xml</groupIdRegex>
</job>
```

note:

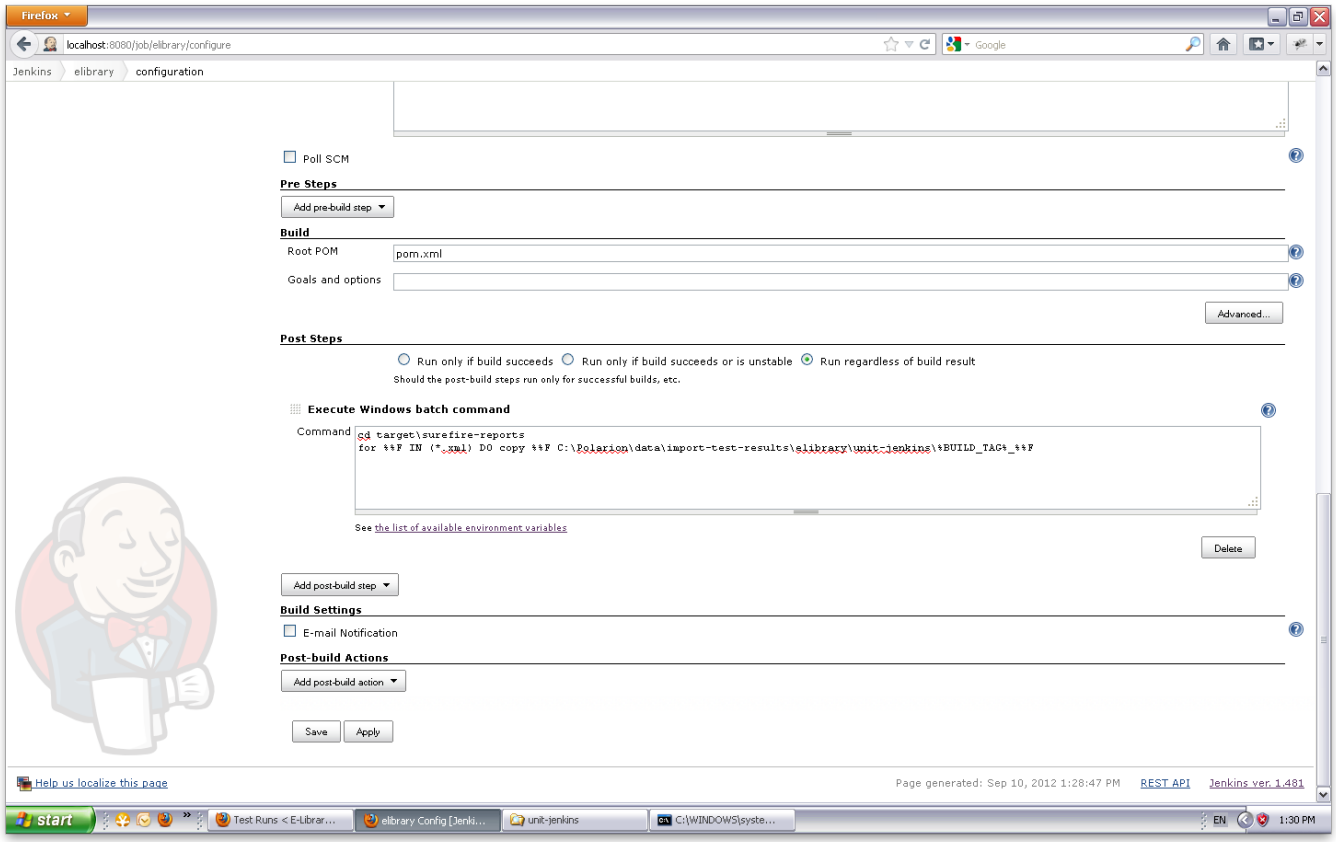
- Make this folder shared so you can access this folder from Jenkins server.

- In this example we have Polarion and Jenkins running on the same server, but you would just use different network drives in your production setup.

III) We must configure Jenkins post-action to copy the test results to the pre-defined folder

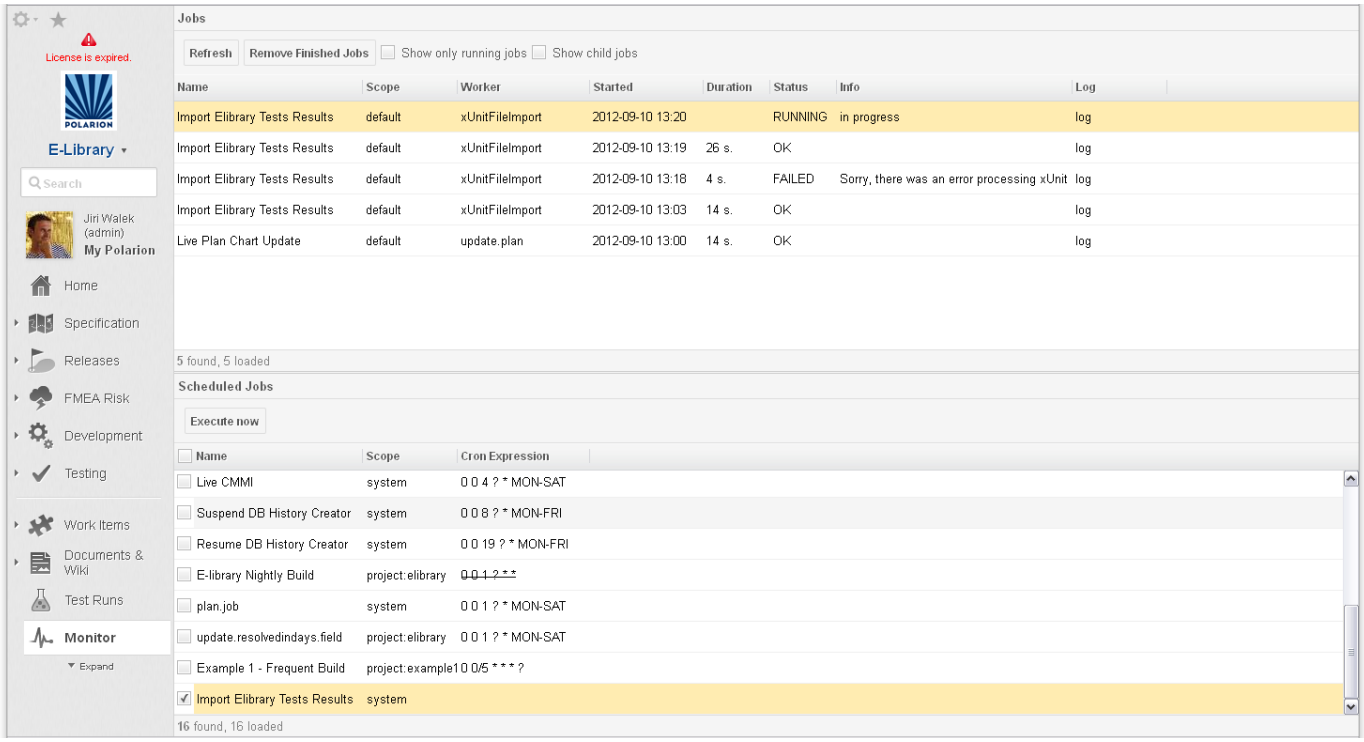
In this case we use Windows command to copy the results, you can also use ant-post build action (and for example merge the test results into one file using http://ant.apache.org/manual/Tasks/junitreport.html)

```
cd target\surefire-reports
for %%F IN (*.xml) DO copy %%F C:\Polarion\data\import-test-results\elibrary\unit-jenkins\%BUILD_TAG%_%%F
```



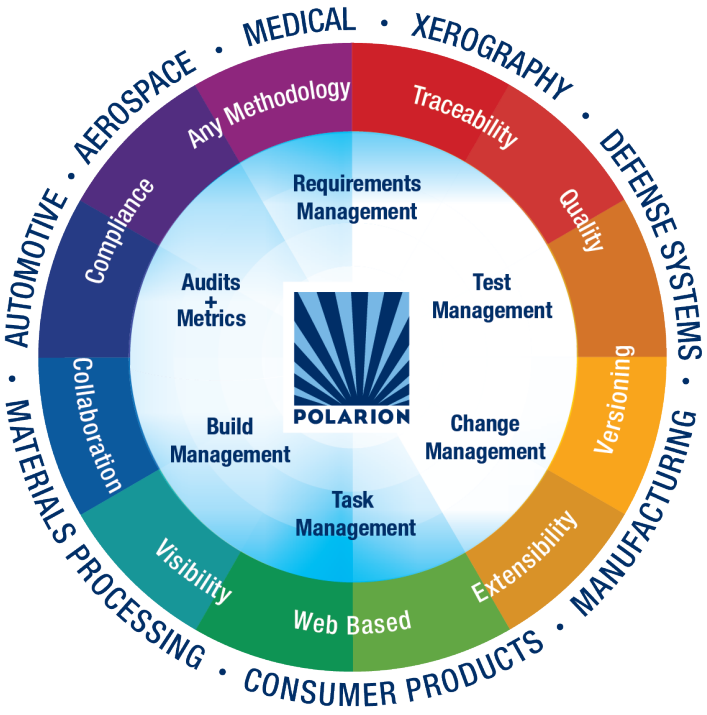
4

Executing the Jenkins build - Now we run the Jenkins build, either manually (if you do not want to wait), or as pre-configured during setup (see section *Receiving Results* for more details on fetching test results).



About Polarion Software

Polarion Software's success is best described by the hundreds of Global 1000 companies and over 1 Million users who rely daily on Polarion's Requirements Management, Quality Assurance and Application Lifecycle Management solutions in their business processes. Polarion is a thriving international company with offices across Europe and North America, and a wide ecosystem of partners world-wide.



Handy Links

- Polarion Application Lifecycle Solutions
- Polarion QA - Collaborative Test Management for QA Managers
- White Paper: Game Testing Evolves
- Polarion Solutions for Automotive OEMs and Suppliers
- Customer Testimonials
- Polarion Events & Webinars
- Contact Polarion Software

Europe, Middle-East, Africa: Polarion Software GmbH
Hedelfinger Straße 60 — 70327 Stuttgart, GERMANY
Tel +49 711 489 9969 - 0
Fax +49 711 489 9969 - 20
www.polarion.com - info@polarion.com

Americas & Asia-Pacific: Polarion Software, Inc.
1001 Marina Village Parkway, Suite 403, Alameda, CA 94501, USA
Tel +1 877 572 4005
Fax +1 510 814 9983
www.polarion.com - info@polarion.com