Continuous integration with Jenkins

Abstract

This document provides information about command line execution of TESSY with continuous integration servers like Jenkins. Refer to the TESSY user manual for an overview about the command line feature of TESSY.
Since TESSY versions 4.1.34, 4.2.20, and 4.3.13 tessyd supports special licenses for the continuous integration (TESSY CI). These licenses can only be utilized by tessyd. Please contact sales@razorcat.com for more information about the pricing and terms. For technical notes refer to application note **License Management Guide**.

Table of contents

Razorcat Development GmbH · Witzlebenplatz 4 · 14057 Berlin
phone: +49 - 30 - 536 357 - 0 · fax: +49 - 30 - 536 357 – 60 · email: support@razorcat.com

1 Introduction

TESSY provides command line operations for usage with continuous integration servers like Jenkins. This allows automatic regression testing of software against the available tests. For such purposes TESSY can run in headless mode without any GUI.

This application note describes the important TESSY-related steps for automatic test execution on a Jenkins server. It is assumed that you are already familiar with the general usage of TESSY and Jenkins.

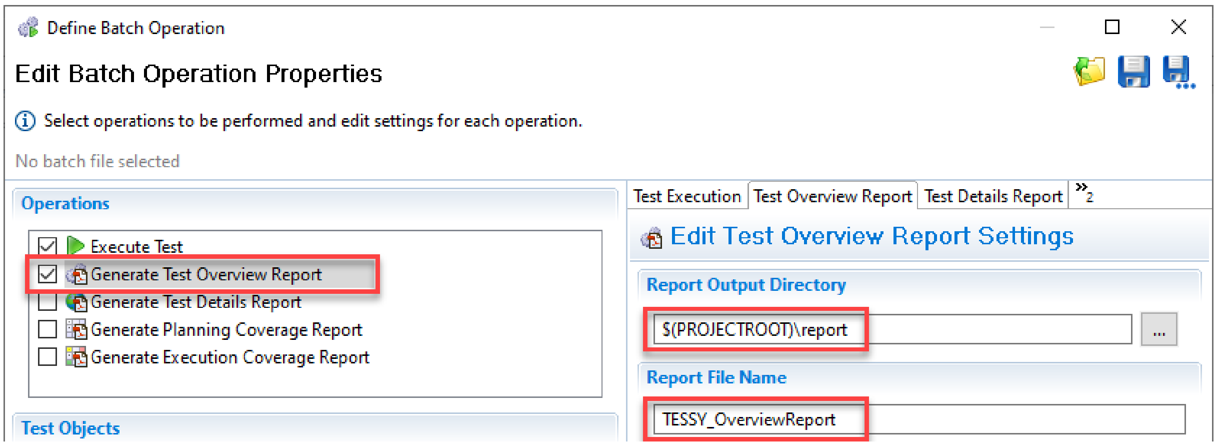
2 Preparing tests

The basis for executing batch tests is a readily prepared test project with all necessary configurations and files. If you can run a test project interactively on a developer computer, you need to save the tests into TMB files like described within section “6.16 Backup, restore, version control” of the TESSY user manual. When checking-in the required files into the version control system, you should be able to restore the whole project into the Jenkins workspace.

2.1 Prepare a batch file (TBS)

Additionally, you need to prepare a TESSY batch script (TBS) file containing the batch operations to be done (i.e. which tests to execute and which reports to generate).

Beside the test that you want to run, you need to create an overview report to check the successful completion of the batch test. Choose an appropriate directory within the project root directory as output directory for the report and a report name without variable parts (i.e. without date/time) like shown below:


Define Batch Operation

Edit Batch Operation Properties

Select operations to be performed and edit settings for each operation.

No batch file selected

Operations

- [✓] Execute Test
- [✓] Generate Test Overview Report
- [ ] Generate Test Details Report
- [ ] Generate Planning Coverage Report
- [ ] Generate Execution Coverage Report

Test Objects

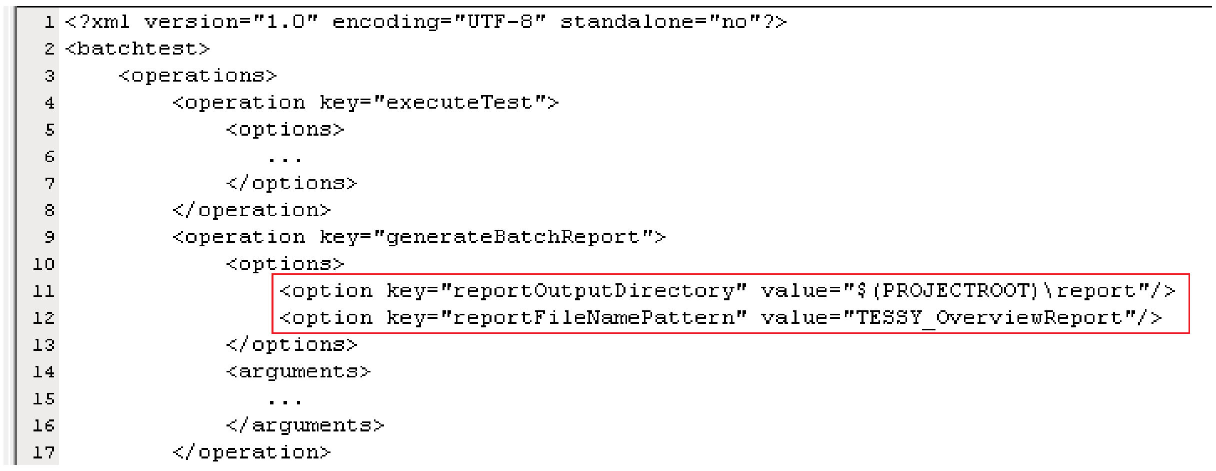
Test Execution | Test Overview Report | Test Details Report

Edit Test Overview Report Settings

Report Output Directory: $(PROJECTROOT)\report

Report File Name: TESSY_OverviewReport

The report output path will be collapsed using the PROJECTROOT variable, so that the resulting TBS file should look like the excerpt shown below:



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<batchtest>
    <operations>
        <operation key="executeTest">
            <options>
               ...
            </options>
        </operation>
        <operation key="generateBatchReport">
            <options>
                <option key="reportOutputDirectory" value="$(PROJECTROOT)\report"/>
                <option key="reportFileNamePattern" value="TESSY_OverviewReport"/>
            </options>
            <arguments>
               ...
            </arguments>
        </operation>
```

Save the batch operation settings into a batch file (TBS) within your project directory. This file will be used for the batch test execution.

3 Running tests

In order to run tests against the current source code available within the version control system, you need to check-out the source code and the test project into the Jenkins workspace. This configuration depends on the version control system being used and will be left out of scope here. We assume that the source code and tests are already checked-out into the current Jenkins workspace by previous Jenkins build steps.

The very simplified command sequence to execute tests with TESSY in headless mode is as follows:

```
tessyd --file <tessy.pdbx>
tessycmd connect
tessycmd restore-db -target project
tessycmd exec-test <path to TBS file>
tessycmd disconnect
tessyd shutdown
```

Please note: In order to run tessyd and tessycmd, you need to add the path to the bin directory of the desired TESSY installation to the PATH variable.

In order to safely run the tests, you should insert checks of the return codes of each command issued, so that the test script aborts on any failure. You will find an example DOS batch script including error handling within the following TESSY installation directory:

```
C:\Program Files\Razorcat\TESSY_4.x\Examples\CommandLine
```

Below is an example of a batch file that can be run interactively to try out the command line execution (of course, you need to replace the project and file names to match the ones you intend to use):

```
set PATH=%PATH%;"C:\Program Files\Razorcat\TESSY_4.x\bin"
tessyd --file C:\Projects\TESSY43\CI\tessy\tessy.pdbx
tessycmd connect
tessycmd restore-db -target project
tessycmd -a exec-test C:\Projects\TESSY43\CI\tessy\ci.tbs
tessycmd disconnect
tessyd shutdown
pause
```

3.1 Starting TESSY without GUI

You need a running instance of TESSY in order to execute any tests. For command line execution, use the “tessyd” application to start TESSY in headless mode (i.e. without GUI) with a given PDBX project file. Afterwards, you need to connect to the running instance and select the project:

```
tessyd --file <tessy.pdbx>
tessycmd connect
```

The respective project will be selected automatically and can be further manipulated.

3.2 Restoring the test project

As a first step of the test, you need to restore the TMB files into the initially empty test project. The following command can be used to conduct this restore operation:

```
tessycmd restore-db -target project
```

This imports all TMB files available within the backup directory of the project. The project is now ready for execution.

3.3 Executing the tests

In order to execute the tests, you need to run the prepared TBS file using the following operation:

```
tessycmd –a exec-test <path to TBS file>
```

This operation executes all tests and creates all reports as defined within the TBS file. You can specify the TBS file either with a path based on the current working directory or with the full absolute path name. The “-a” option of “tessycmd” shows progress information when running tests interactively (e.g. from a local command line window).

Please note: The test execution will not be aborted if single tests fail or cannot be executed. You need to create an overview test report and evaluated this report for failed or not executed tests as described within the next chapter.

4 Evaluating results

When you need to further process the results to provide custom reports “tessycmd” also provides an `xslt` command. It also provides a default XSL Script which creates a short text-based summary of the specified overview report file.

4.1 Execute with default Script

Assumed your overview report file is generated to the workspace relative path `report\TESSY_OverviewReport.xml` the following command will generate a summary of the report to the standard output.

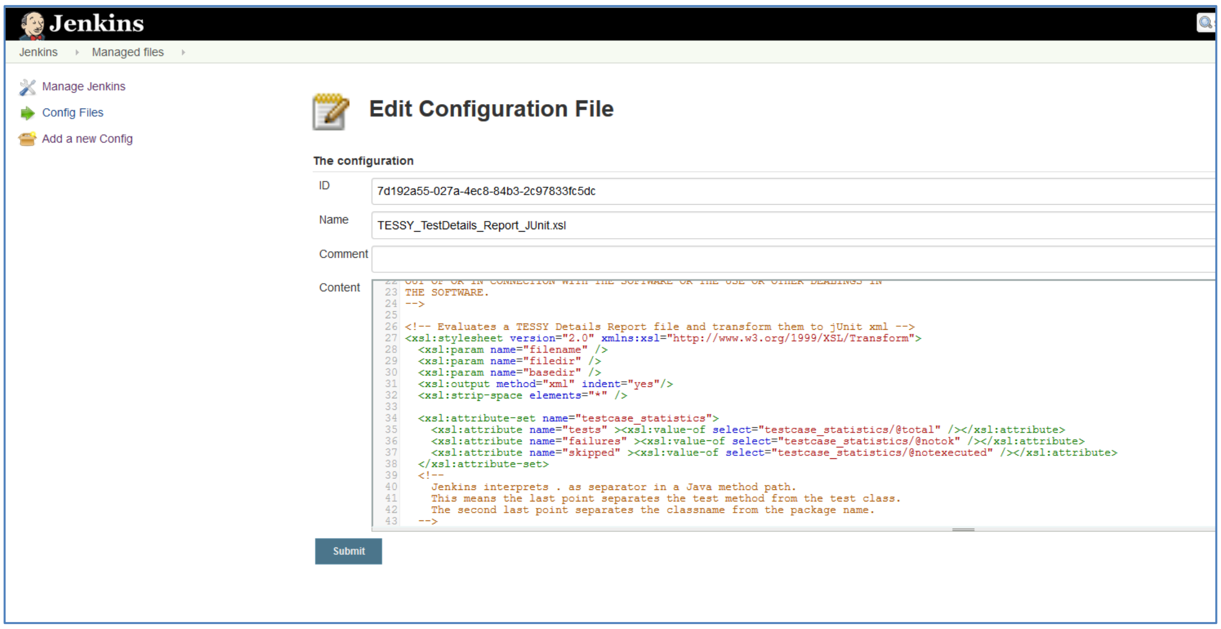
```
tessycmd xslt %WORKSPACE%\report\TESSY_OverviewReport.xml
```

4.2 Provide XSL script as configuration file

To provide an XSL script the “Managed Scripts” plugin needs to be installed. An XSL script that converts TESSY overview report in a JUnit compatible format is provided within the TESSY installation in the following folder:

```
C:\Programme\Razorcat\TESSY_4.x\bin\plugins\com.razorcat.tessy
.reporting.templates_4.x.xx\ci\TESSY_TestDetails_Report_JUnit.
xsl
```

For usage with Jenkins, we recommend adding the XSL script contents as configuration file (“Simple XML file”) like shown below:



Edit Configuration File

The configuration

ID: 7d192a55-027a-4ec8-84b3-2c97833fc5dc

Name: TESSY_TestDetails_Report_JUnit.xsl

Comment:

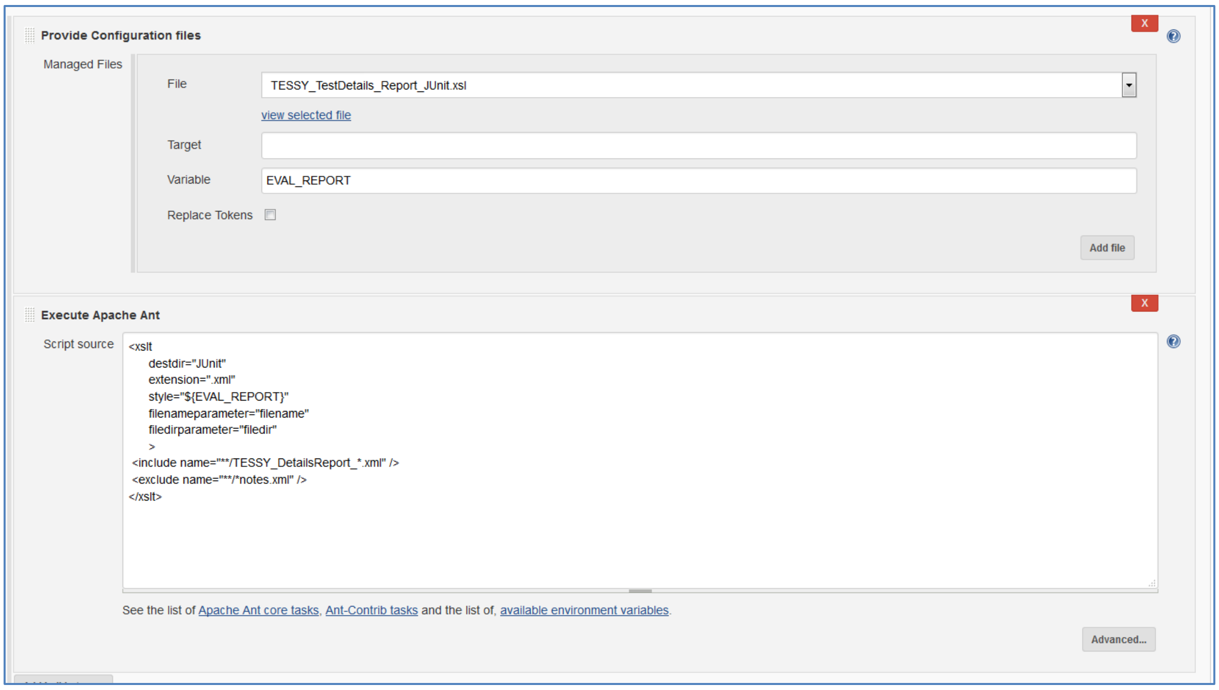
Content:

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
-->

<!-- Evaluates a TESSY Details Report file and transform them to jUnit xml -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="filename" />
  <xsl:param name="filedir" />
  <xsl:param name="basedir" />
  <xsl:output method="xml" indent="yes"/>
  <xsl:strip-space elements="*" />

  <xsl:attribute-set name="testcase_statistics">
    <xsl:attribute name="tests" ><xsl:value-of select="testcase_statistics/@total" /></xsl:attribute>
    <xsl:attribute name="failures" ><xsl:value-of select="testcase_statistics/@notok" /></xsl:attribute>
    <xsl:attribute name="skipped" ><xsl:value-of select="testcase_statistics/@notexecuted" /></xsl:attribute>
  </xsl:attribute-set>
  <!--
    Jenkins interprets . as separator in a Java method path.
    This means the last point separates the test method from the test class.
    The second last point separates the classname from the package name.
  -->
```

This configuration file can conveniently be used in the later report evaluation build steps.

4.3 Run the XSL check as ANT task

If you have added the XSL script as Jenkins configuration file, you can use it within an ANT task that executes the XSL script. Therefore the “AntExec” plugin needs to be installed. The configuration can be done like shown below:


Provide Configuration files

Managed Files

File: TESSY_TestDetails_Report_JUnit.xsl

Target:

Variable: EVAL_REPORT

Replace Tokens: ☐

Execute Apache Ant

Script source:

```
<xslt
     destdir="JUnit"
     extension=".xml"
     style="${EVAL_REPORT}"
     filenameparameter="filename"
     filedirparameter="filedir"
     >
 <include name="**/TESSY_DetailsReport_*.xml" />
 <exclude name="**/*notes.xml" />
</xslt>
```

See the list of Apache Ant core tasks, Ant-Contrib tasks and the list of, available environment variables.

The first build step provides the configuration file which can be referenced later using the specified variable name “EVAL_REPORT”. The next build step is the ANT invocation using the XSL script referenced by the “EVAL_REPORT” variable. The complete command line is here:

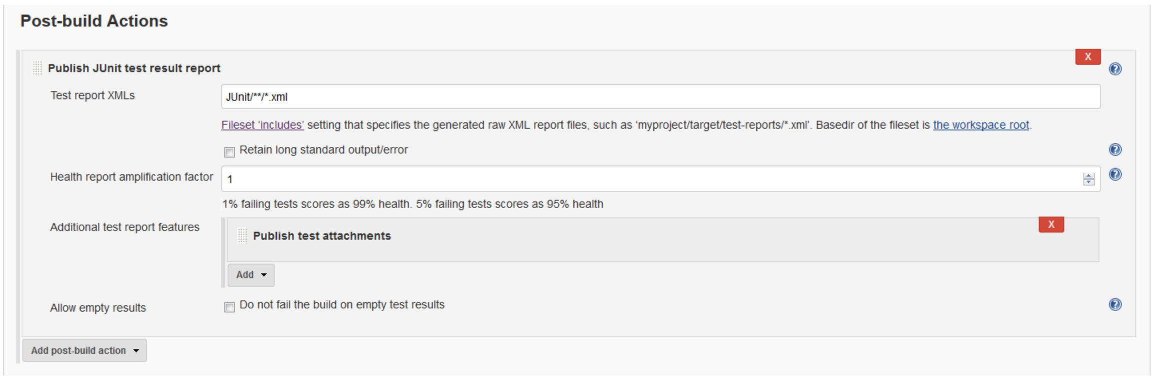
```
<xslt
      destdir="JUnit"
      basedir="."
      extension=".xml"
      style="${EVAL_REPORT}" >
 <include name="**/TESSY_DetailsReport_*.xml" />
 <exclude name="**/*notes.xml" />
</xslt>
```

The ANT script processes all test reports matching the pattern in the `<include` … element. The current working directory will be the workspace of the job, you can change that by providing another value for the attribute `basedir`. If the report file is not available or if the XSL script fails, the whole build step will be marked as failed by Jenkins.

More information about the `xslt` ANT task can be found in the ANT documentation at https://ant.apache.org/manual/Tasks/style.html

4.4 Publish JUnit reports

The XSL script provided by TESSY adds special information to the JUnit report which allows the “JUnit Attachments” Plugin archive and link the original TESSY reports.



Post-build Actions

Publish JUnit test result report

Test report XMLs: JUnit/**/*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

Retain long standard output/error

Health report amplification factor: 1

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Additional test report features: Publish test attachments

Allow empty results: Do not fail the build on empty test results

Add a Post-build Action “Publish JUnit test result report” and adjust it like shown in the figure above. Don’t forget to add the “Additional test report features” named “Publish test attachments”.

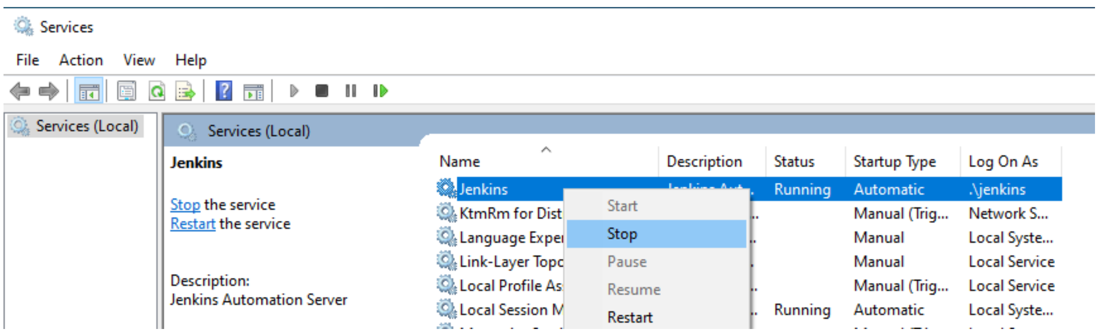
5 Setting up Jenkins

After a standard local Jenkins installation, the Jenkins service normally runs using an internal system account. When running TESSY test executions locally, using such a Jenkins service will fail because such an internal system account does not provide the necessary environment for TESSY tests.

For usage with TESSY, the Jenkins service itself (if TESSY tests shall be run on the Jenkins host) and additional agents need to be run with accounts suitable for TESSY test execution. Such an account shall be able to login as user with all necessary Windows settings for standard users (e.g. accessibility of %APPDATA% and %USERPROFILE% folders).

5.1 Prepare a user account for Jenkins

Add an administrative user profile to your local computer (e.g. named "jenkins") and login once to activate this account. Then open the services panel as administrator and select the "Jenkins" service:



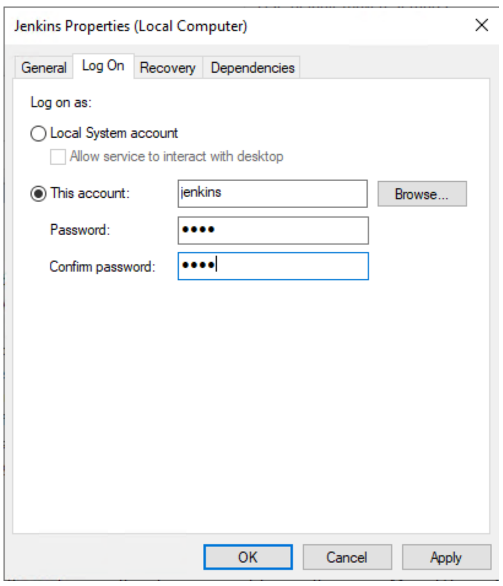
Jenkins

Stop the service
Restart the service

Description:
Jenkins Automation Server

Name	Description	Status	Startup Type	Log On As
Jenkins	Jenkins Aut...	Running	Automatic	.\jenkins
KtmRm for Dist			Manual (Trig...	Network S...
Language Exper			Manual	Local Syste...
Link-Layer Topc			Manual	Local Service
Local Profile As			Manual (Trig...	Local Service
Local Session M		Running	Automatic	Local Syste...

Stop the running service and open the properties. Enter the user account data within the "Log On" tab as shown below:



Log on as:

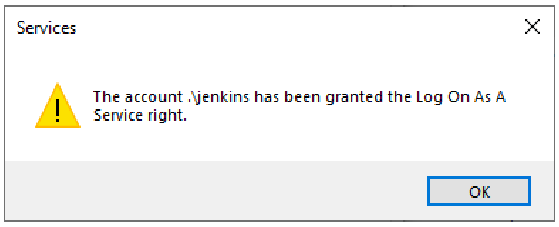
( ) Local System account
[ ] Allow service to interact with desktop

(●) This account: jenkins

Password: ••••

Confirm password: ••••

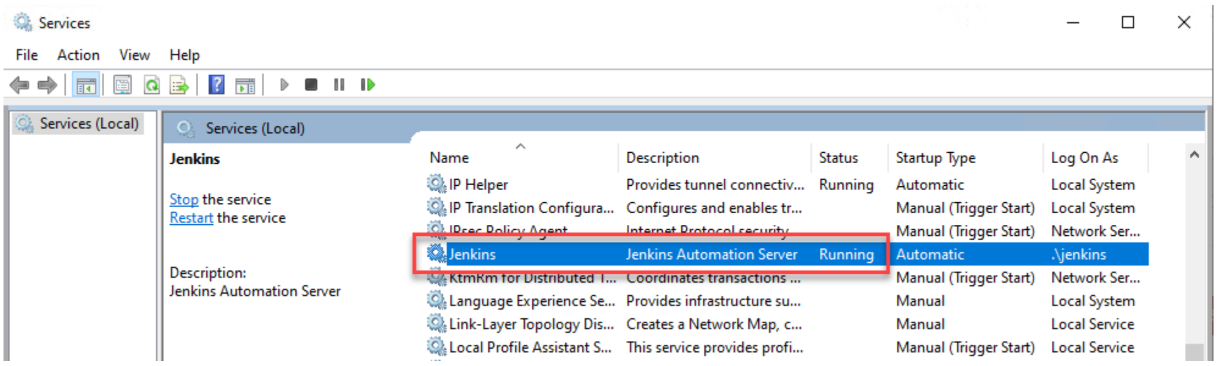
Click on “OK” to save these settings. Most probably Windows will open the following dialog to show that a special right was granted to the chosen user account:



Services

The account .\jenkins has been granted the Log On As A Service right.

Now start the service and make sure it is running:



Jenkins

Stop the service
Restart the service

Description:
Jenkins Automation Server

Name	Description	Status	Startup Type	Log On As
IP Helper	Provides tunnel connectiv...	Running	Automatic	Local System
IP Translation Configura...	Configures and enables tr...		Manual (Trigger Start)	Local System
IPsec Policy Agent	Internet Protocol security...		Manual (Trigger Start)	Network Ser...
Jenkins	Jenkins Automation Server	Running	Automatic	.\jenkins
KtmRm for Distributed T...	Coordinates transactions ...		Manual (Trigger Start)	Network Ser...
Language Experience Se...	Provides infrastructure su...		Manual	Local System
Link-Layer Topology Dis...	Creates a Network Map, c...		Manual	Local Service
Local Profile Assistant S...	This service provides profi...		Manual (Trigger Start)	Local Service

5.2 Preparing Jenkins agents

The setup of Jenkins agents is out of scope of this document, please refer to the Jenkins documentation. But as already mentioned, it is important that the Jenkins agent also runs using a normal login user account with all necessary Windows settings for standard users (e.g. accessibility of %APPDATA% and %USERPROFILE% folders).

6 Using Jenkins TESSY plugin

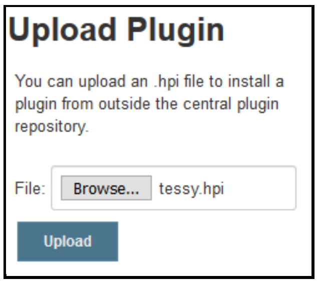
The TESSY plugin for Jenkins is available here:

https://www.razorcat.com/files/files/tessy/jenkins/tessy.hpi

Download the “tessy.hpi” file and store it to your local computer.

6.1 Install plugins

Go to “Manage Jenkins -> Manage Plugins” and then select “Advanced”. Select the plugin file “tessy.hpi” and upload it.



Upload Plugin

You can upload an .hpi file to install a plugin from outside the central plugin repository.

File: Browse... tessy.hpi

Ensure that you also have the following plugins installed:

- “Git plugin”
- “Workspace Cleanup Plugin”

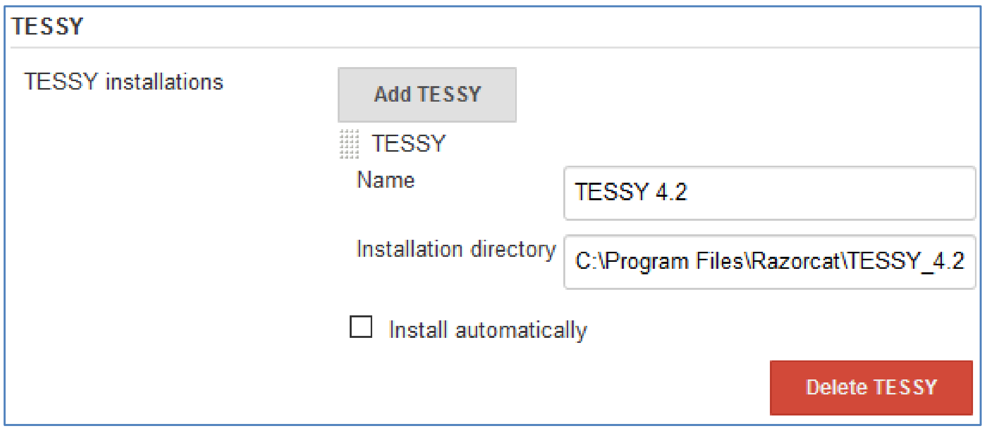
6.2 Setting up the TESSY plugin

After the installation you must set up the plugin. The TESSY license server and the TESSY installations to be used need to be configured.

6.2.1 Adding TESSY installations

Navigate to “Manage Jenkins -> Global Tool Configuration” and expand the “TESSY” section.

Fill in the name and the directory of your installation of TESSY. You can add several versions of TESSY if desired. Within the job configuration you can later choose the respective version to be used for each job.



TESSY

TESSY installations

TESSY

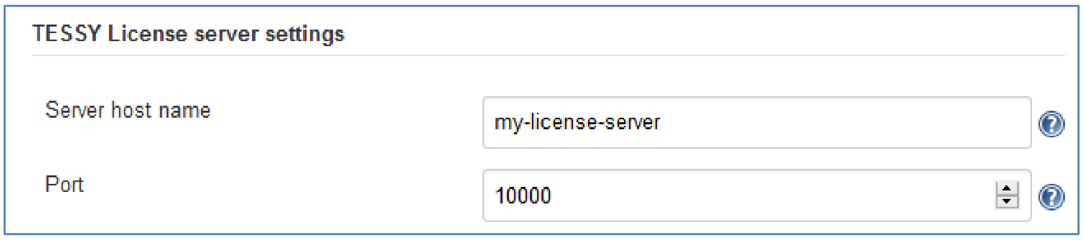
Name: TESSY 4.2

Installation directory: C:\Program Files\Razorcat\TESSY_4.2

☐ Install automatically

6.2.2 TESSY license server settings

Navigate to “Manage Jenkins -> Configure System” and scroll down to the “TESSY License server settings” section. Enter the license server host name and port to be used as shown below:



TESSY License server settings

Server host name	my-license-server
Port	10000

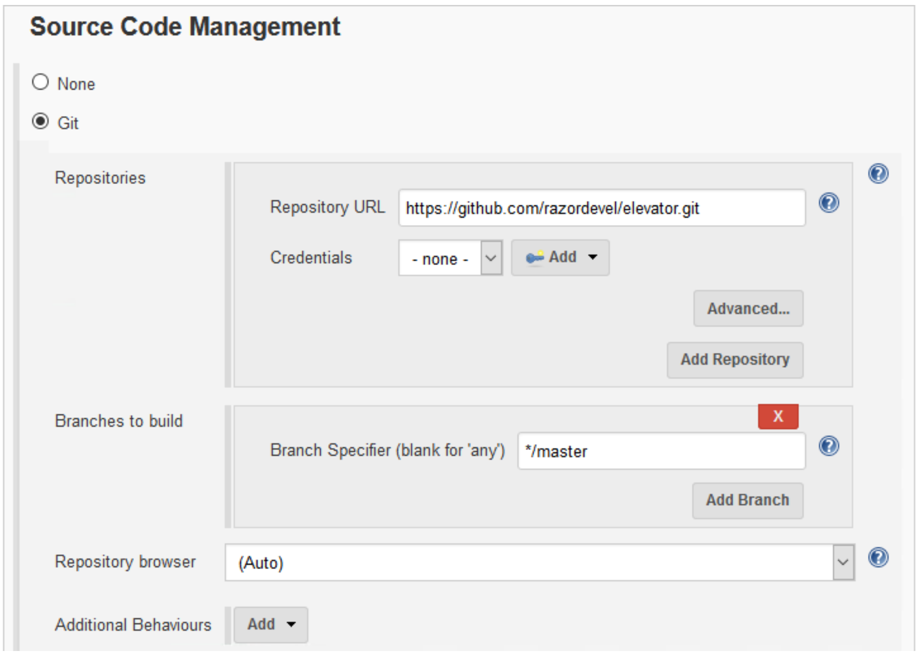
6.3 Creating a job

Create a “Freestyle project” within Jenkins to use the TESSY plugin features.

6.3.1 Source Code Management

Check out your test project (source files and tests) from your version control system. As an example, you can check out our example project from GITHUB:

https://github.com/razordevel/elevator.git



Source Code Management

None [✓ Git]

Repositories
- Repository URL: https://github.com/razordevel/elevator.git
- Credentials: - none -

Branches to build
- Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

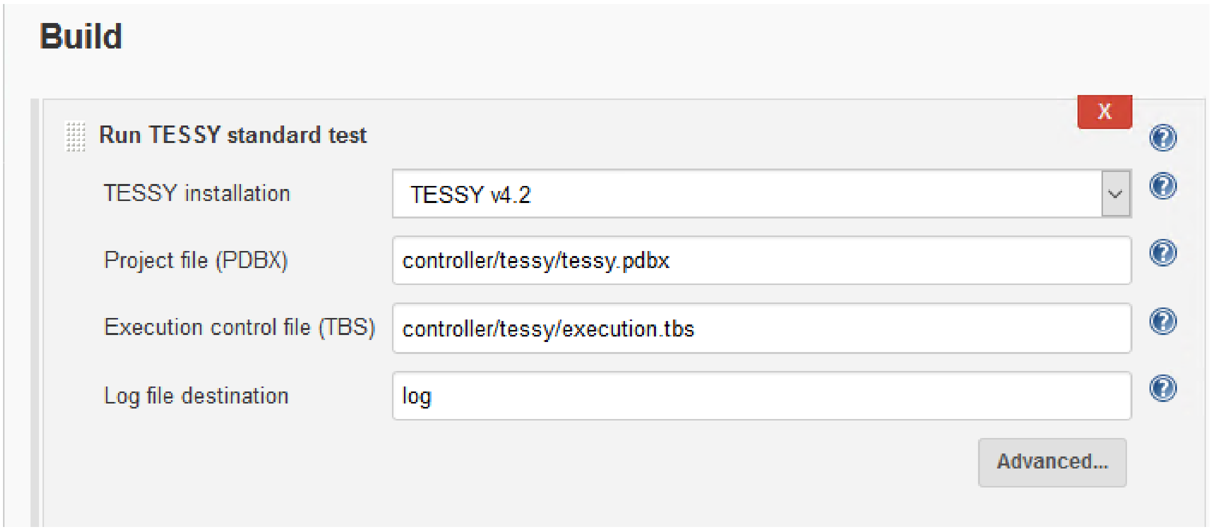
Additional Behaviours

Please note: There are a couple of test objects within this project so that the test execution will take some minutes.

6.3.2 Build step

The TESSY plugin provides a build step. Select “Add build step -> Run TESSY standard test”. Then select the TESSY installation you want to use and fill in the path to your PDBX and TBS file. You can either use relative or absolute paths here. Build parameter substitution is also supported for these entries.

The log file destination shall be a directory within your workspace. The problems log file will be copied into this directory after shutdown of TESSY. This directory as well as any report output directory should be archived.



Build

Run TESSY standard test

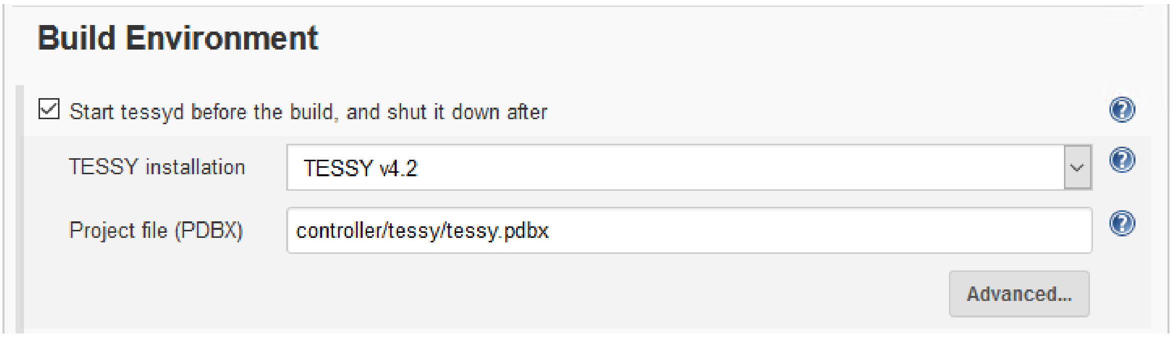
TESSY installation	TESSY v4.2
Project file (PDBX)	controller/tessy/tessy.pdbx
Execution control file (TBS)	controller/tessy/execution.tbs
Log file destination	log

Please note: The “Log file destination” option is only supported for TESSY v4.2.6 and later. Leave this field empty if you are using any older TESSY version.

In case of any failures, this build step will automatically shut down TESSY so that a clean restart is ensured.

6.3.3 Build environment

In the case you want to run your own commands via “Execute Windows batch command” you should use the following option from the “Build Environment”:



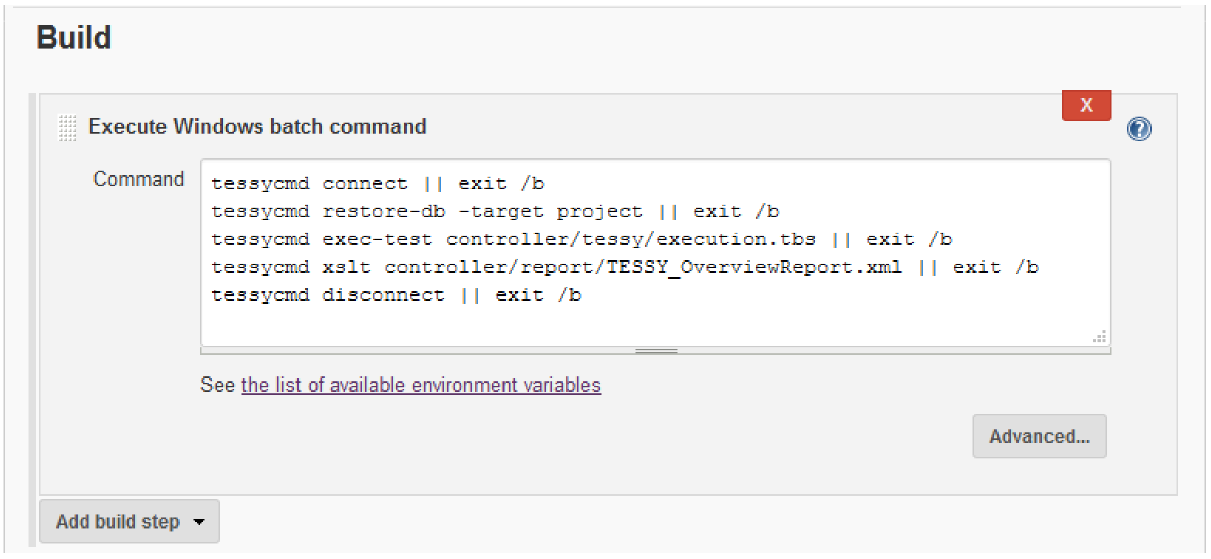
Build Environment

☑ Start tessyd before the build, and shut it down after

TESSY installation	TESSY v4.2
Project file (PDBX)	controller/tessy/tessy.pdbx

With this option set you don’t have to start and stop “tessyd” and you don’t need to catch failures. This will be done automatically by Jenkins. This option will automatically shut down TESSY so that a clean restart is ensured.

Also, the path to the TESSY binary directory will be added to the PATH variable so that you can use “tessycmd” without full path. An example build step is shown below:


Build
Execute Windows batch command
X
Command

```
tessycmd connect || exit /b
tessycmd restore-db -target project || exit /b
tessycmd exec-test controller/tessy/execution.tbs || exit /b
tessycmd xslt controller/report/TESSY_OverviewReport.xml || exit /b
tessycmd disconnect || exit /b
```

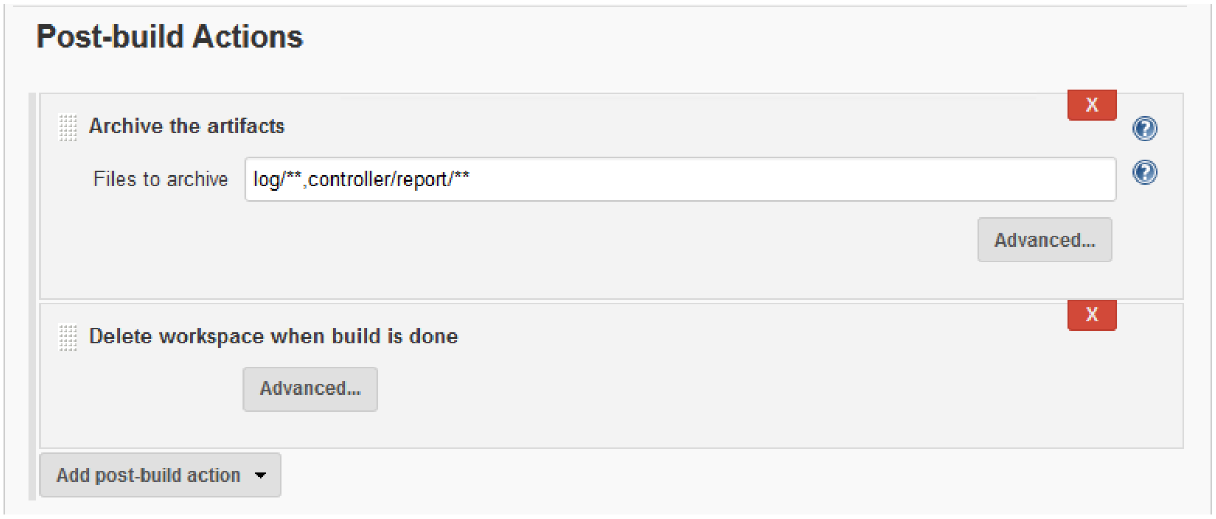
See the list of available environment variables
Advanced...
Add build step

Please note: Do not use this option, if you are using the “Run TESSY standard test” build step as described within section 6.3.2! The job will fail in this case.

6.3.4 Archive the test reports

If you have configured within your TBS file that your reports shall be generated into the “report” directory, you need to select the files contained within this directory to be archived. If you configured another path for the output directory, please change the configuration in this step accordingly.

Also, the collected problems log shall be archived for further investigation in case of test execution or any other kind of problems.


Post-build Actions

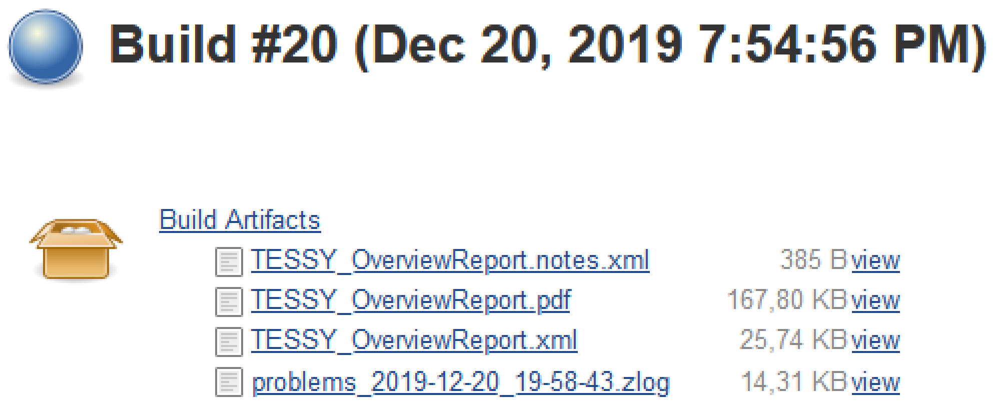
Archive the artifacts

Files to archive: log/**,controller/report/**

Delete workspace when build is done

As you see in the screenshot above, we delete the workspace when the build is done. We do this, to ensure that there are no reports or logs left in the workspace from previous jobs and we have a clean build.

After a successful build you should see the following artefacts of the build job:


Build #20 (Dec 20, 2019 7:54:56 PM)

Build Artifacts

TESSY_OverviewReport.notes.xml	385 B	view
TESSY_OverviewReport.pdf	167,80 KB	view
TESSY_OverviewReport.xml	25,74 KB	view
problems_2019-12-20_19-58-43.zlog	14,31 KB	view

The problems log file can be downloaded and opened with a TESSY version on any computer to review if there were problems during test execution (only supported by TESSY v4.2.5 and later).

7 Troubleshooting

7.1 Starting tessyd in special process environments

When trying to start “tessyd” from a process that is running inside a job and that job hasn't set the JOB_OBJECT_LIMIT_BREAKAWAY_OK the process creation will fail with error code 5 (ERROR_ACCESS_DENIED). Starting “tessyd” from inside Eclipse is an example where the starter process used to launch an application creates a job object without the JOB_OBJECT_LIMIT_BREAKAWAY_OK attribute.

To start “tessyd” in such an environment you have to perform the following steps.

- In the environment of the process from which you start “tessyd” there must exist a variable `TE_FLAGS` with the value `–dont-breakaway-from-job`. Therefore, you may add the entry `TE_FLAGS=--dont-breakaway-from-job` to the environment variables section (`[Environment Variables]`) of your TESSY configuration file (`%APPDATA%\Razorcat\TESSY\4.3.x\config\tessy.conf`) or you have to set the environment variable before executing “tessyd”.

- You have to specify the command line option --dont-breakaway-from-job when invoking the “tessyd” command.

  - `tessyd --dont-breakaway-from-job –-file <tessy.pdbx>`