**Readme for Java utility**

Purpose of the Java utility is to convert the Zimbra automation harness report in Testrail’s json format. Once the json is available, same would be used to publish the automation results to Testrail using existing testrail\_api.rb utility.

Current Java utility implementation has been developed considered Nunit test framework, but can be very well extended to other frameworks like TestNG. Currently, accepted test report source is XML format. If the automation harness report format is non-XML, this utility needs to be extended.

Setup:

In case of command line execution, we would need to download jackson jars. If we have setup zm-qa project in Eclipse, Jackson jars are downloaded automatically using ivy.

Download Jackson jars needed from the Maven Central repository. Below jars are needed

jackson-core-2.8.5.jar

jackson-annotations-2.8.5.jar

jackson-databind-2.8.5.jar

To download jars

* browse site <http://search.maven.org/>
* Search jackson-core and download the latest version jar
* Others jars can be downloaded in same way

For commandline execution, I copy these jars to zm-qa/src/java

Source code (.java) and properties files are available at zm-qa/src/java/com/zimbra/qa/testrail

Steps to run Java utility (command-line):

1. Go to java utility root folder i.e. zm-qa/src/java

Then, compile the java program as

javac -cp jackson-core-2.8.5.jar:jackson-databind-2.8.5.jar:jackson-annotations-2.8.5.jar:. com/zimbra/qa/testrail/ProcessXMLToJSON.java

1. Run the java program as

java -cp jackson-core-2.8.5.jar:jackson-databind-2.8.5.jar:jackson-annotations-2.8.5.jar:. com/zimbra/qa/testrail/ProcessXMLToJSON <Path of Testreport.xml>

Running java program will create json format file named TestrailResult.json in testrail folder. Log file will also be generated with name Log-M-d\_HHmmss.log in testrail folder. It will also create mapping.properties in in testrail folder if createmapping=true in config.properties

Steps to run Java utility (Eclipse):

1. zm-qa repository has to be correctly setup in Eclipse, resolving dependancies on other repos like zm-native, zm-common, zm-soap, zm-client, zm-store
2. Set Arguments as TestResult.xml path in Run Configurations for class com.zimbra.qa.testrail.ProcessXMLToJSON
3. Right click “ProcessXMLToJSON.java” file, select Run As > Java Application

Java run progress/log can be seen in Console view in Eclipse

After program is run, it will create TestrailResult.json, Log file and mapping.properties in testrail folder.

In addition to the automation report (.xml), Java program also needs two properties file to be created and reside in same folder:

1. config.properties – this file has all configuration information related to Testrail project, suite, plan, milestone, etc

Sample config properties would look like

#Testrail configuration file

project\_id=<Id of the project having suite and testcases>[Mandatory field]

run\_name=<Run name to be created and updated to Testplan>[Mandatory field]

description=<Run name description>[Optional field, can be left empty]

milestone\_id=<Milestone Id linked with plan\_name plan>[Optional field, can be left empty]

suite\_id=<Id of the suite where testcases reside>[Mandatory field]

#assignedto\_id=

#include\_all=

plan\_name=<Plan name against which run has to be linked>[Mandatory field]

#plan\_description=

comment=<Comment to be added for each testcase if test has passed, if test has failed, Java program updates the failure message> [Mandatory field]

createmapping=true <set this as true or false>[Mandatory field]

testrail\_url=<Testrail API URL>[Mandatory field]

testrail\_user=<email\_address\_Testrail\_user>[Mandatory field]

testrail\_userkey=<Testrail User API Key>[Mandatory field]

1. mapping.properties – This file has mapping between Harness Testcase-Id and Testrail Testcase-id. This file will be created dynamically if createmapping property is equal to true in config.properties.

Once mapping.properties is created and if no more automation tests are added, user can set createmapping=false and Java program will not create mapping.properties (will re-use existing mapping.properties)

Sample mapping properties would look like

#AutomationTestCaseName TestrailTestCaseID

clientTests.Client.Calendar.AppointmentRequests.AppointmentAction.AppointmentAction\_01=294358

clientTests.Client.Calendar.AppointmentRequests.AppointmentAction.AppointmentAction\_02=294359

Before running the Java program, config.properties needs to be updated with correct project/suite/plan/run and Testrail user email address and API key.

Updating test results in Testrail:

Once the TestrailResult.json file is created by Java program, same can be consumed by testrail\_api.rb to update the results in Testrail

ruby testrail\_api.rb -f TestrailResult.json -u prashant.ajari@synacor.com -k IumjpZHsuXp9JBKdexRD-XXXXXXXXX