**Command Entry User Guide**

V1.0

10/31/2016

Shawn Yan

**Contents**

[1. INTRODUCTION 3](#_Toc468110120)

[1.1 Components 3](#_Toc468110121)

[1.2 Major Features 3](#_Toc468110122)

[2. Getting started with Command Entry 3](#_Toc468110123)

[2.1 Upload suites 3](#_Toc468110124)

[2.2 Create test run/plan 4](#_Toc468110125)

[2.3 Diamond auto installation 6](#_Toc468110126)

# INTRODUCTION

## Components

* + 1. Upload test suites
    2. Create test run/plan
    3. Diamond auto installation

## Major Features

* + 1. **Upload test suites via customers configuration file (Excel File)**

The scripts will upload all suites/sections/cases to TestRail database which are parsed from the Excel file.

* + 1. **Create test run/plan**

Sometimes you may want to create test run/plan in batch mode instead of the operation on the web. It’s a better way to manage your test runs/plans.

* + 1. **Install Diamond on Windows and Linux**

Here you can install Diamond on Windows and Linux in batch mode. And it will create a mark file to make the yose client can add the Diamond into its configuration file.

# Getting started with Command Entry

## Upload suites

* + 1. URL

<http://lshlabd0001/viewvc/platform/trunk/platforms/tmp/cmd_entry/uploadSuites/>

* + 1. Description

The scripts will parse the Excel file and get the suite contents, section list and case list. Then it will execute the TestRail API to add/update test suites, add/update test sections and add/update/delete test cases.

* + 1. Usage

Script name: excel2testrail.py

Usage: excel2testrail.py [options]

Options:

-h, --help show this help message and exit

-d, --debug Debug only

-f INFILE, --infile=INFILE

specify Excel conf file

-u USERNAME, --username=USERNAME

specify testrail user name.

-p PASSWORD, --password=PASSWORD

specify testrail password

-t TESTRAIL, --testrail=TESTRAIL

specify testrail URL, format: <http://$server/testrail>

>> python excel2testrail.py -f $your\_suite\_excel\_file --debug

It will create a folder named after the excel file name. Excel file will be transferred to csv format thus make the scripts can parse the contents from them. The file \_suite\_section\_case.ini file is the inner temporary file for uploading. You can check this file before execute the uploading flow. The “debug” option will abandon the uploading operation.

>> python excel2testrail.py --infile=$your\_suite\_excel\_file

Generate the inner temporary file and upload the suites, sections and cases.

>> python excel2testrail.py --infile=$your\_suite\_excel\_file -u xx --password=yy -t zz

The scripts have default username, password and TestRail server name. You can specify them in the command line.

* + 1. Tips

1. The suite name is the key string to find the suite in the project. If you want to change the suite name, you should modify it via web.
2. If you want to remove a suite/section, remove it via web.
3. Default section name is “Test Cases”, if you do not specify the section name for the case, it will use the default one.
4. The contents in TestRail database is not the duplicate of your Excel file’s. You need double check the TestRail data after uploading.

## Create test run/plan

This tool is developed for creating test run or test plan in batch mode.

* + 1. URL

http://lshlabd0001/viewvc/platform/trunk/platforms/tmp/cmd\_entry/runTestRail/

* + 1. Create test run

Generally, if you want to create a run, you should define the run name. You should know the suite id, project id and milestone id for this run, and also you need specify your own configuration for the test run.

The standard ini format for test run is:

|  |
| --- |
| [run $run\_name]  project\_id = 34  project\_name = xx  milestone\_id = 28  milestone\_name = xx  suite\_id = 99  suite\_name = xx  description = <[Machine] >  <group = rna\_regression\_group >  <[CaseInfo] >  <repository = http://lshlabd0001/diamond/silicon >  <suite\_path = 06\_macgyver >  <[System] >  <[Environment] >  <[LaunchCommand] >  <cmd = python DEV/bin/run\_diamond.py --run-map-trce >  <[Software] >  <diamond=3.8.0.115 > |

1. You can comments the options with “;” ahead of a line, not any space before the “;”.
2. The option xx\_id has the higher priority than xx\_name. And the option like xx\_id is recommended.
3. All prefix blanks in description will be kept and will upload to the TestRail database.
4. The scripts will exit with error message if you specify the wrong id or wrong name.
   * 1. Create test plan

Generally, if you want to create a plan, you should define the plan name. You should know the project\_id, suite\_ids and milestone id. Sometimes you will need specify your own configuration for the test plan.

The standard ini format for test plan is:

|  |
| --- |
| [plan $plan\_name]  project\_id = 4  project\_name =  milestone\_id = 16  milestone\_name =  description = <[one] >  <two=1 >  ;suite\_id\_list = 105, 97  ;suite\_name\_list = FE\_21\_LSE\_Ind, MISC\_customer\_cr  ex\_suite\_id\_list = 23,229  ;ex\_suite\_name\_list = skjh  ;include\_all = 1 |

1. xx\_id has higher priority than xx\_name;
2. if include\_all is True, all suites under this project will be added into the test plan;
3. if suite\_id\_list or suite\_name\_list exists, the ex\_suite\_id\_list and ex\_suite\_name\_list will be omitted;
4. you can select test suites or skip test suites by different options.
   * 1. Usage
5. Print help message

>> python run\_testrail.py --help>python run\_testrail.py -h

Usage: run\_testrail.py [options]

Options:

-h, --help show this help message and exit

-D, --debug print debug message

-F INFILE, --infile=INFILE

specify input configuration file

-U USERNAME, --username=USERNAME

specify TestRail API username

-P PASSWORD, --password=PASSWORD

specify TestRail API password

-T TESTRAIL, --testrail=TESTRAIL

specify TestRail Web Address, format:

http://$server/testrail

1. Debug only

>> python run\_testrail.py -D --infile=xxxx.ini

1. Update according your conf file

>>python run\_testrail.py --infile=xxxx.ini

1. The scripts already set the default username/password and testrail URL for linux-d50553. you can change then by the args like: ”-U my\_name --password=lattice --testrail=http://linux17v/testrail”

## Diamond auto installation

* + 1. URL

http://lshlabd0001/viewvc/platform/trunk/platforms/tmp/cmd\_entry/biweeklyFlow

* + 1. Requirements

1. RabbitMQ Service on linux-d50553;
2. config file: install/uninstall iss file for Diamond installation file(on Windows) and other arguments
3. Trigger file
   * 1. RabbitMQ Service

The communication between server and clients uses RabbitMQ. So you should start the service firstly. Now we use linux-d50553 as the RabbitMQ server.

* + 1. Config file

In the “install” section, you should define the pairs of exe and iss files.

option dst\_folder is defined for specifying the diamond installation path.

In the “uninstall”section, we only need specify the pair of diamond base exe and uninstall iss files.

in the “others” section, the conf file will copy the exe file to local\_exe\_path if copy\_exe is True.

In the beginning, you can execute the following command line to generate the iss files.

>> python run\_flow.py --flow=creator --conf-file=./exe\_iss/sample\_conf.ini

* + 1. Trigger file

It is the trigger settings for checking the build in specified time point.

day\_of\_week: valid option is 1~7

day\_of\_month: valid option is 1 ~ 12

timestamp: format is HH:MM

build: if build specified, the trigger will be activated when the time meet the (day\_of\_week or day\_of\_month) after timestamp. The whole flow will be terminated when the build is found. if build not specified, the scripts will scan and get the newest and stable build version from $release\_folder. and the whole flow will try to meet the next trigger and search the newest build.

run\_conf\_file: if run\_conf\_file specified, the scripts will create the test runs after delivering the install message to the clients.

release\_folder: will scan and search the right build it this folder.

machine\_list: will deliver out the install message to these clients.

* + 1. Usage

1. create iss files

>> python run\_flow.py --flow=creator --conf-file=./exe\_iss/sample\_conf.ini

1. auto send out install message (Server)

>> python run\_flow.py --flow=server --debug --trigger=trigger.ini \

--conf-file=.\exe\_iss\sample\_conf.ini

1. receive and install Diamond (Client)

>> python run\_flow.py --flow=client