User Platform General Usage

**Software Quality Team**

ABSTRACT

This usage guide document briefly describes the usage of Test Management Platform (TMP) for software regression testing. It will include the following areas:

1. Preparation
2. Core script (case level)
3. Client (suite level)
4. Platform (project level)
5. Work flow Demo

REVISION HISTORY

|  |  |  |
| --- | --- | --- |
| REVISION | RELEASE DATE | COMMENTS |
| V1.0 | 4/28/2017 | Initial draft |
| V1.1 | 5/23/2017 | Update based on ping’s feedback |

CONTENTS

[1. Scope 5](#_Toc484620007)

[2. What is TMP 6](#_Toc484620008)

[2.1 TMP logical structure 6](#_Toc484620009)

[2.2 TMP User perspective 6](#_Toc484620010)

[2.2.1 TMP Website (server)—Project level 7](#_Toc484620011)

[2.2.2 TMP Client –suite level 7](#_Toc484620012)

[2.2.3 Core scripts –case level 7](#_Toc484620013)

[3. Preparation 8](#_Toc484620014)

[3.1 TMP Website setup 8](#_Toc484620015)

[3.1.1 TMP account 8](#_Toc484620016)

[3.1.2 TMP website 8](#_Toc484620017)

[3.1.3 Webpage explorer 8](#_Toc484620018)

[3.2 TMP Client setup 8](#_Toc484620019)

[3.2.1 Operation System requirements 8](#_Toc484620020)

[3.2.2 Hard Ware suggestion 8](#_Toc484620021)

[3.2.3 System setting (Linux only) 8](#_Toc484620022)

[3.2.4 Dependency of software and version 8](#_Toc484620023)

[3.3 Core scripts setup 9](#_Toc484620024)

[4. Case level test with built-in core scripts 10](#_Toc484620025)

[4.1 Current support flow 10](#_Toc484620026)

[4.2 Case format requirements 10](#_Toc484620027)

[4.2.1 Case Structure 10](#_Toc484620028)

[4.2.2 Info file building 10](#_Toc484620029)

[4.2.3 conf file building 11](#_Toc484620030)

[4.2.4 testdata folder building 13](#_Toc484620031)

[4.2.5 testmethod folder building 13](#_Toc484620032)

[4.2.6 testdesign folder building 13](#_Toc484620033)

[4.3 Built-in core script introduction 14](#_Toc484620034)

[4.3.1 Diamond run with core script 14](#_Toc484620035)

[4.3.2 Radiant run with core script 14](#_Toc484620036)

[4.3.3 ICECube2 run with core script 15](#_Toc484620037)

[4.4 Case-Script directory layout 15](#_Toc484620038)

[5. Suite level test with TMP client 16](#_Toc484620039)

[5.1 Current support flow 16](#_Toc484620040)

[5.1.1 Built-in flow 16](#_Toc484620041)

[5.1.2 Customized flow 16](#_Toc484620042)

[5.2 Suite file format requirements 16](#_Toc484620043)

[5.2.1 Common section 17](#_Toc484620044)

[5.2.2 Specific section 17](#_Toc484620045)

[5.2.3 Detailed requirements 17](#_Toc484620046)

[5.3 Client introduction 18](#_Toc484620047)

[5.3.1 Export TMP client 18](#_Toc484620048)

[5.3.2 Installation with wizard 18](#_Toc484620049)

[5.3.3 Manually install 20](#_Toc484620050)

[5.3.4 Launch TMP Client 21](#_Toc484620051)

[5.3.5 TMP Client usage 24](#_Toc484620052)

[5.3.6 Simple usage 29](#_Toc484620053)

[6. Project level test with TMP Website 30](#_Toc484620054)

[6.1 TMP website overview 30](#_Toc484620055)

[6.1.1 TMP Management Structure 30](#_Toc484620056)

[6.2 How to launch a test suite 31](#_Toc484620057)

[6.2.1 Add Test Run 32](#_Toc484620058)

[6.2.2 Pause/stop/play a test run 33](#_Toc484620059)

[6.2.3 Retest cases in a test run 33](#_Toc484620060)

[6.2.4 Rerun 34](#_Toc484620061)

[6.3 How to launch a test plan 34](#_Toc484620062)

[7. Demo work flow 35](#_Toc484620063)

[7.1 Test case build 35](#_Toc484620064)

[7.2 Test suite build 35](#_Toc484620065)

[7.3 Local test/validation 35](#_Toc484620066)

[7.4 Upload test suite to TMP website 35](#_Toc484620067)

[7.5 Run test suite on TMP webpage 36](#_Toc484620068)

[8. Appendix 37](#_Toc484620069)

# Scope

Test Management Platform (TMP) has two kinds of users:

1. TMP Admin
2. TMP User

TMP Admin: Administrator is responsible for TMP server installation, setup and maintenance. Also administrator will setup the initial common machine farm and update software build on these machines.

TMP User: TMP user is general user, can modify project and run regression based on webpages. TMP user also can launch test suites on personal clients.

This document will focus on TMP user including general TMP structure, regression project/suite launch on webpage/client, standard suite build and run on client, standard case build and run with core scripts.

# What is TMP

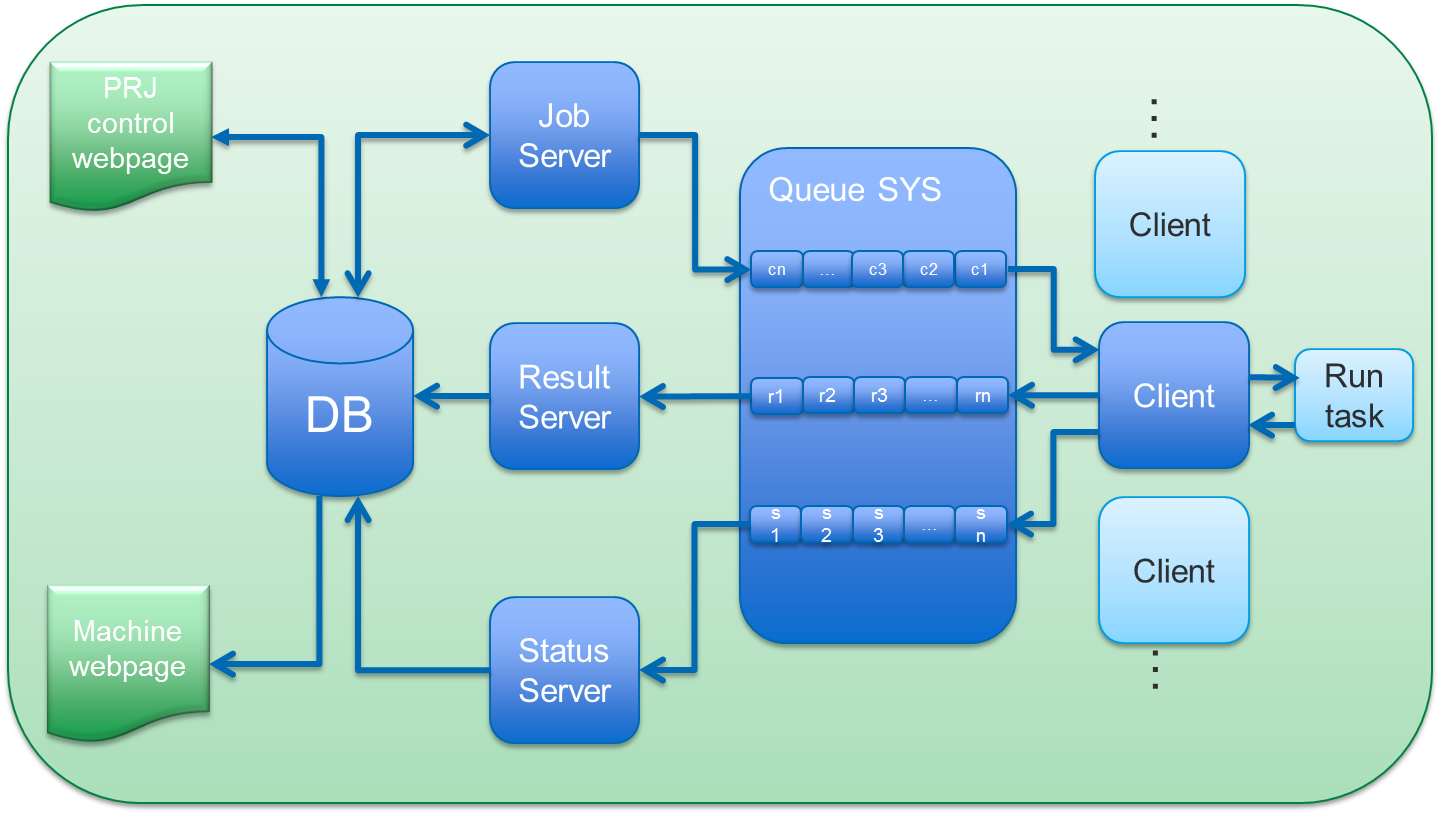
Test Management Platform (TMP) is a regression platform which makes regression testing be much more standard, efficient and visible.

TMP also offers user some powerful capabilities including:

1. Project level management
2. Regression flow control
3. Accelerates regression with machine farm
4. OS cover support
5. Customizes report generate

## TMP logical structure

Here is the overview of logical blocks for TMP.

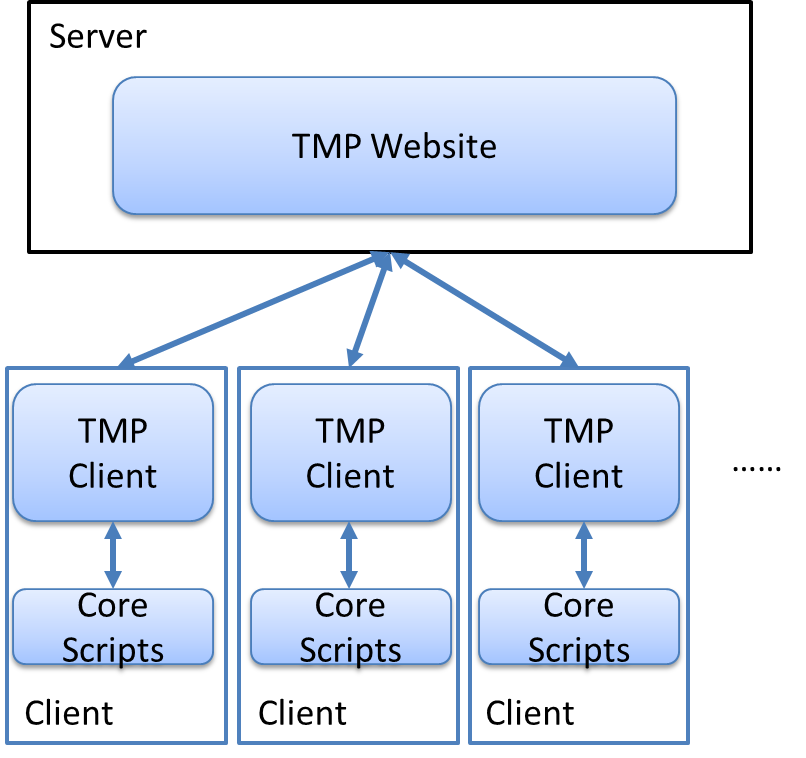


1. TMP logical structure

When a regression project/run is created, web server will pass down the task data into database. Job server will assign tasks to queue system one by one based on the task requirements. Clients will run tasks and return results base on self-condition (meet task requirements or not). Result server will get the task result from queue system and put it into database for finally web report.

## TMP User perspective

TMP is a collection of software which including Core scripts, TMP client and TMP server. Here is a picture of general structure based on user perspective.



1. TMP deployment structure

### TMP Website (server)—Project level

In logical structure view, most of the blocks will be placed in server side, i.e. web server, database, job server, result server, queue system…

TMP Server will offer project/suite level management and regression.

More details can be found in [section 6](#_Project_level_test).

### TMP Client –suite level

TMP client allow TMP user to do suite level test and suite upload to TMP server.

More details can be found in [section 5](#_Suite_level_test).

### Core scripts –case level

Core scripts allow TMP user to do case level test. A client can invoke built-in core scripts for case level test.

The client also can invoke third-party scripts (with special script output required) for case level test.

More details can be found in [section 4](#_Case_level_test).

# Preparation

## TMP Website setup

### TMP account

TMP user may get TMP account from local TMP admin.

### TMP website

TMP user may get TMP website address from local TMP admin.

### Webpage explorer

IE11 is recommended for TMP web access.

## TMP Client setup

### Operation System requirements

The following OS is supported by TMP:

Window:

Windows 7 or later.

Linux:

RedHat 6 or later

CentOS 7 or later

OpenSUSE 42 or later

Unbuntu 16 or later

### Hard Ware suggestion

CPU core number: 4

Physical memory: 8G

### System setting (Linux only)

TMP Client will launch many test cases concurrently (Depends on user’s setting in Menu 🡪 Setting 🡪 Preference 🡪 Max Threads) which may exceed the default user open files limitation (1024) on Linux platform.

To avoid this limitation we need to add following two lines into file /etc/security/limits.conf.

\* soft nofile 32768

\* hard nofile 65536

To verify this modification, user can type: ulimit –a

### Dependency of software and version

#### 3.2.4.1 Java

TMP Client is built with Java 8 build121 32 bits. Although it’s patched in the whole JRE in software build but still suggest user install JRE on your operation system (Especially on Linux platform).

Minimum Java version 1.8 is required, user can check Java version by type following command in console:

java –version

#### 3.2.4.2 Python

TMP Client will invoke built-in core scripts to launch each test case. Core scripts are built by Python software, so Python software is required on your machine, and Python version is 2.7.x in 32 bits , user can check Python version by typing following command in console:

Python –version

TMP client uses a python package “psutil” to run system check, so it is also need to install this package.

Python “psutil” version 5.0.0 in 32 bits is required.

#### 3.2.4.3 Subversion

In most case, core scripts and user case are stored in subversion. So TMP client needs subversion ready on run machine.

Subversion version 1.6 or higher is required, user can check it by typing following command in console:

svn –version

## Core scripts setup

If user would like to do case level test with Core scripts, you need a Python environment.

Python version 2.7.x in 32 bits is required, you can check Python version by typing following command in console:

Python –version

# Case level test with built-in core scripts

## Current support flow

Built-in core script supports the following software/flows:

Diamond:

Engine Command line flow, i.e. synthesis, map, par

Implementation flow, i.e. run synthesis, run map, run par trce

Simulation flow, i.e. RTL simulation, post map simulation, post par simulation

General GUI flow

iCECube2:

Implementation flow, i.e. run synthesis, run map, run par trce

Simulation flow, i.e. RTL simulation, post map simulation, post par simulation

General GUI flow

Radiant (Developing):

Engine Command line flow, i.e. synthesis, map, par

Implementation flow, i.e. run synthesis, run map, run par trce

Simulation flow, i.e. RTL simulation, post map simulation, post par simulation

General GUI flow

## Case format requirements

This section will introduce the typical case requirements. For detailed case building, please see:

<<User Standard Case Build Usage.docx>>, you may get it from your local TMP Admin.

### Case Structure

The general structure for a standard case will like the following:

--testdata/ data for data driven case test

--testdesign/ real test design

--testmethod/ test scripts or method

--bqs.conf result check file

--bqs.info run boot file (flow, simulation run)

--readme.txt case info

### Info file building

Info file is also called boot file which is used to leading the built-in core scripts to get the case information, i.e. project file location, simulation information…

Typically an info file was constructed with “project” section and “simulation” section.

The following description is based on FPGA flow (Diamond, ICECube2 and Radiant)

#### 4.2.2.1 Project Section

Project section is used to record project information. This section should start with “[qa]” and use the following options:

1. ldf\_file= used to address the ldf project file
2. Inc\_path= include extra path for rebuilding project file use
3. others\_path= used to address other path for rebuilding project
4. base\_lpf= used to address original lpf file for rebuilding project file
5. devkit= device information for rebuilding project file
6. top\_module=<rtl\_top> specify the top module for rebuilding project file
7. src\_files= list all source files for rebuilding project file
8. edf\_file= edf files
9. map\_ngd= map NGD file for MAP CMD flow implement
10. map\_ncd= map NCD file for PAR CMD flow implement
11. lpf\_file= lpf file for PAR CMD flow implement
12. par\_ncd= par NCD file for TRCE CMD flow implement
13. par\_prf= par PRF file for TRCE CMD flow implement
14. project\_name= project name setting
15. impl\_name= implementation name setting
16. …

\*value list in “<>" are default values. If the value not given or these options not shown in project section will be considered as use the default values

We can simply divide these options into three classes:

1. LDF entry: use this ldf file for design entry.(a)
2. SRC entry: use these source files to rebuilding project file and entry LDF entry.(b-g)
3. CMD entry: use the given individual files for standalone command run.(h-p)

#### 4.2.2.2 Simulation Section

Simulation section is used to record simulation information. This section should start with “[sim]” and with the following options:

1. dev\_lib=<> used to address which device lib should be used
2. pri\_lib=<work> used to address which lib should be used first
3. tb\_file= used to address the test bench file
4. tb\_vector=<test\_vector.in> used to address the simulation input vector
5. sim\_top=<sim\_top> used to address test bench top model name
6. uut\_name=<UUT> used to address instance name of top RTL module
7. sim\_time= <10 us> how long the simulation will run
8. do\_msim= ModleSIM simulation macro file
9. do\_ahdl= active-HDL simulation macro file
10. do\_qsim= questasim simulation macro file

\*value list in “<>" are default values, if the value not given or these options not shown in simulation section will be considered as use the default values

\*\*For “dev\_lib” please leave it blank there or just omit this option to make the script search the right device simulation library automatically (according current device and HDL language). If you write any device library here, script will use the specific device library and ignore the current project device and language style. So take care!

\*\*\*please always write the right simulation time for “sim\_time” which can be “<x> us” or “-all”. Please replace “<x>” to the right number you wanted.

We can simply divide these options into three classes:

1. LIB path: simulation libraries should be used.(a-b)
2. SRC entry: source files for simulation use.(c-g)
3. Macro entry: macro files for simulation use.(h-j)

LIB path should be always used for simulation. While the different entry (b, c) implements different simulation flow and the priority will be: b > c.

### conf file building

conf file is used to guide the built-in core scripts check the run results.

Conf file is divided into three sections: configuration section, method section, check section. With these configurations built-in core scripts can implement check and report functions.

#### 4.2.3.1 Configuration Section

This section is for report title configurations:

1. area= configure the top model name
2. type= configure the sub model name
3. cr\_note= current test case fail reason CR or others
4. cr\_status= CRs open or closed

If area = STA, type= Features you will see the following information in the case report folder:

1. Report example

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area | Type | Case | Device | Result | Comments |
| STA | Features | Xxxx | xxxx | xxxx |  |

#### 4.2.3.2 Method Section

This section is for self-check result purpose. Here is an example:

1. check\_block\_1 = 1
2. check\_block\_2 = 0
3. check\_lines\_1 = 1
4. check\_data\_1 = 1

Let’s take item a as an example: “check\_block\_1” is a method name, “1” means check flow will check this method, in this condition there must be a “check\_block\_1” method to specify how to run check flow. While “check\_block\_2= 0” mean check flow will not run check this check method.

If all themethods list in this Section passed check the final check result will be “Pass” otherwise “Fail”.

#### 4.2.3.3 Check Section

Check section is used to define how to run check flow, here are some frequently-used check methods:

##### 4.2.3.3.1 check\_lines

1) format

[check\_lines]

title = check\_test1

file = <path>\<file>

check\_1 = <string\_1>

times = <number> --optional

check\_<num> = <string\_2>

2) description

This method will try to find the <string\_1> assigned by check\_1 in <file> and treats it as a start point (line 1),

(If times option is used, scripts will find <string\_1> <number> times and then treats the last one as start point),

Then try to check whether line <num> (a shift value) has <string\_2>. if yes, the result of this method is true.

##### 4.2.3.3.2 check\_block

1) Format

[check\_block]

Title = check\_blocks

compare\_file = <path>\<file>

golden\_file = <Gold\_file>

2) Description

This method will try to compare the <Gold\_file> with the <file>, if <Gold\_file> is included in <file>, return true.

##### 4.2.3.3.3 check\_multiline

1) Format

[check\_multiline]

file = <path>\<file>

check\_line = <total\_string>

2) Description

This method will try to find the <total\_string> assigned by check\_line in <file> regardless the "space" and "line feed".

This method will benefit the situation that the <total\_string> won't be at the same line and changing for each regression.

### testdata folder building

This folder is used for the collection of test data if have.

### testmethod folder building

This folder is used for the collection of test scripts/method if have.

### testdesign folder building

This folder is used for the collection of real test design.

The following description is based on FPGA (Diamond, ICECube2) while other software may have different definition.

#### 4.2.6.1 models folder

“models” folder is used to place the models your design need, actually these files was generated by SCUBA engine,

Recommend setting:

a) If you only have one model file it is recommended to use the name of “model\_top” as both the top module and the file name.

#### 4.2.6.2 others folder

“others” folder used to place IP core files, NGO files and some case level script files.

Recommend setting:

NA

#### 4.2.6.3 par folder

“par” folder used to place the project files.

Recommend setting:

1. it’s better to name project file as prj\_top.ldf
2. it’s better to use default implementation(impl1)

#### 4.2.6.4 sim folder

“sim” folder used to collect all simulation files:

Recommend setting:

1. The default test bench file name is “sim\_tb.v/vhd”
2. The default active-HDL macro file name is “ahdl.do”
3. The default ModelSIM macro file name is “msim.do”
4. The test vector input file name is “test\_vector.in” if you have.

#### 4.2.6.5 source folder

“source” folder is used to collect all RTL source files:

Recommend setting:

1. the top RTL file name is “rtl\_top”
2. the top module name is “rtl\_top”

## Built-in core script introduction

See appendix for build location

Here is the brief directory structure of DEV

DEV/

Bin/

run\_diamond.py

run\_diamondng.py

run\_icecube.py

conf/

default.ini

docs/

tools/

### Diamond run with core script

We use run\_diamond.py to launch Diamond associated test including command flow, implementation flow and simulation flow. For the detailed usage, please refer to: <<BQS\_script\_usage\_model.docx>>

You can get script help by run following command in console:

“Python DEV/bin/run\_diamond.py ––help”

Here are some frequently-used commands

1. Run PAR and trace flow:

“***python DEV/bin/run\_diamond.py --design=xxx --diamond=xxx --run-par-trce***”

1. Run MAP flow:

“***python DEV/bin/run\_ diamond.py --design=xxx --diamond=xxx --run-map***”

1. Run IBIS export:

“***python DEV/bin/run\_ diamond.py --design=xxx --diamond=xxx --run-export-ibis***”

1. Run RTL simulation with Active-HDL:

***“python DEV/bin/run\_ diamond.py --design=xxx --diamond=xxx --sim-rtl”***

1. Run post MAP simulation with ModelSIM:

***“python DEV/bin/run\_ diamond.py --design=xxx --diamond=xxx --sim-map-vlg –sim-modelsim”***

1. Run post PAR simulation with Active-HDL

***“python DEV/bin/run\_ diamond.py --design=xxx --diamond=xxx --sim-par-vlg”***

### Radiant run with core script

We use run\_diamondng.py to launch Radiant associated test including command flow, implementation flow and simulation flow. For the detailed usage, please refer to: <<BQS\_script\_usage\_model.docx>>

You can get script help by run following command in console:

“Python DEV/bin/run\_diamondng.py ––help”

Here are some frequently-used commands

1. Run PAR and Trace flow:

“***python DEV/bin/run\_diamondng.py --design=xxx --diamond=xxx --run-par-trce***”

1. Run MAP flow:

“***python DEV/bin/run\_ diamondng.py --design=xxx --diamond=xxx --run-map***”

1. Run IBIS export:

“***python DEV/bin/run\_ diamondng.py --design=xxx --diamond=xxx --run-export-ibis***”

1. Run RTL simulation with Active-HDL:

***“python DEV/bin/run\_ diamondng.py --design=xxx --diamond=xxx --sim-rtl”***

1. Run post MAP simulation with ModelSIM:

***“python DEV/bin/run\_ diamondng.py --design=xxx --diamond=xxx --sim-map-vlg –sim-modelsim”***

### ICECube2 run with core script

We use run\_icecube.py to launch iCECube2 associated test including implementation flow and simulation flow. For the detailed usage, please refer to: <<BQS\_script\_usage\_model.docx>>

You can get script help by run following command in console:

“Python DEV/bin/run\_icecube.py ––help”

Here are some frequently-used commands

1. Run PAR flow:

“***python DEV/bin/run\_icecube.py --design=xxx --ice=xxx --pushbutton***”

1. Run par simulation flow with active-HDL:

“***python DEV/bin/run\_icecube.py --design=xxx --ice=xxx --sim-vhd***”

## Case-Script directory layout

Here is the recommend layout for your test case and core-script collection:

<Test folder>/

DEV/

Case1/

Case2/

# Suite level test with TMP client

## Current support flow

TMP client is just a suite level management software which will parse the whole suite file into test cases, launch these cases with given scripts/command and then get the case result for suite level report. In summary, TMP client will:

1. Parse tasks from local suite file or remote TMP website into cases
2. Launch case with built-in core scripts or user scripts
3. Get case report

### Built-in flow

All test flow launched with built-in core script (as we discussed in “case level test” section

) will be called as Built-in flow. So same as case level introduction, current support will be:

1. Diamond implementation, simulation flow and GUI flow
2. Radiant implementation, simulation flow and GUI flow
3. ICECube2 implementation, simulation flow and GUI flow

For details, please see [section 4.1](#_Current_support_flow).

### Customized flow

TMP also supports user script launch with some special requirements listed here:

#### Case-script/command layout

Here is the must layout for your test case and core-script collection:

<Test folder>/

<User script>/

Case1/

Case2/

In general, user script and case must in the same folder.

#### Script/Command return value

When user script/command launch a test case TMP client will read this script/command return value and here is the definition:

Exit\_value == 0 --PASS

Exit\_value == 1 --FAIL

Exit\_value == 2 --TBD (to be decided/to be manually checked)

#### Script/Command console output

TMP client also parsers user script/command out. The last error output will be caught and shown in case report. For example:

“Error: detail information”

## Suite file format requirements

Please ask your local “TMP Admin” for a demo suite file.

Test suite file is a collection of test cases including case information and case requirements and using Microsoft Excel file to keep all these information. For details, please refer to: << User Standard suite build usage.docx>>

For a specific test case the following information is required:

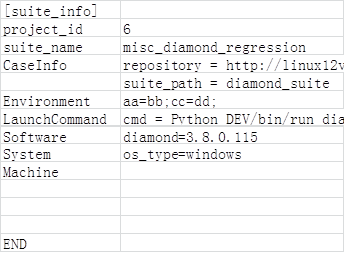
1. CaseInfo: where is the test case physical address, where is the script address.
2. Environment: runtime environment for this test case run.
3. LaunchCommand: launch command for test case run
4. Software: software requirement for test case run
5. System: System requirement for test case run
6. Machine: Machine requirement for test case run

We divide all these test case information into two sections: common section, specific section.

### Common section

Common section: common information and requirements for all test cases. We put all these information into “suite” sheet in suite file.

Here is an example:

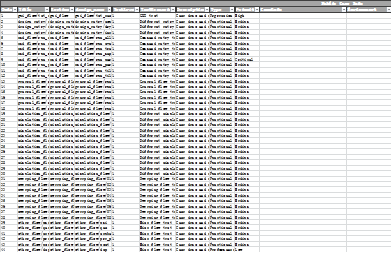


1. Suite sheet

### Specific section

Specific section: specific information and requirements for each detailed case. We put all these information into “case” sheet in suite file.

Here is an example:



1. Case sheet

### Detailed requirements

In demo suite file (version higher than 1.05) there is a “description” sheet. This sheet has a detailed introduction on how to build a suite file.

## Client introduction

### Export TMP client

See appendix for build location.

TMP Client is built with the rules as follows:

Tmp\_client\_a.bb\_installer\_ccc\_ddd.dd

a: Public release version

bb: Internal release version

ccc: bin/src, binary package or source package

ddd.dd: install platform which can be:

1. linux.run: package for Linux with install wizard.
2. windows.exe: package for windows with install wizard.
3. all.tar.gz: package for all platform with manually install.

For example:

1. tmp\_client\_2.06\_installer\_bin\_linux.run:

Build version 2.06 for Linux (RedHat, Ubuntu, SUSE) install with wizard available.

1. tmp\_client\_2.06\_installer\_bin\_windows.exe:

Build version 2.06 for Windows (win7 or later) install with wizard available.

1. tmp\_client\_2.06\_installer\_src\_all.tar.gz:

Build version 2.06 for all platform install without install wizard (manually install).

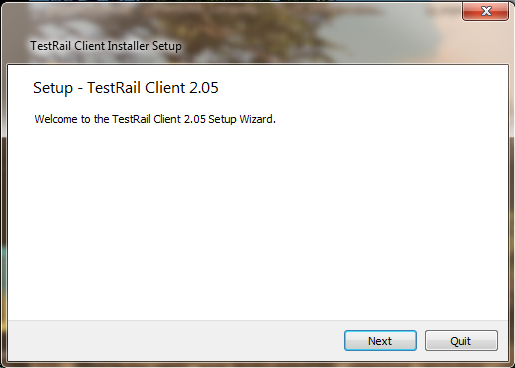
Please export the right build according to your requirements.

### Installation with wizard

TMP Client package offers general GUI setup, so you can simply following with the wizard to install it.

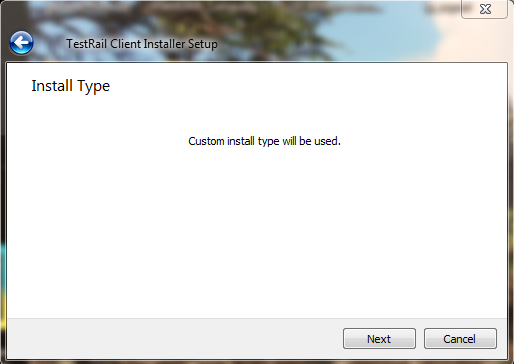
Step 1: Double click installed package.

Step 2: Confirm installed build and click “Next”



1. Confirm install build

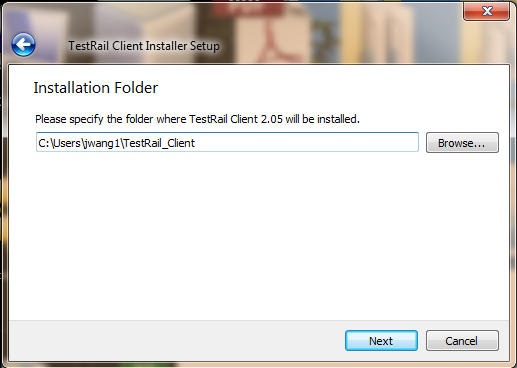
Step 3: Select installed type



1. Select install type

Step 4: Select installed folder

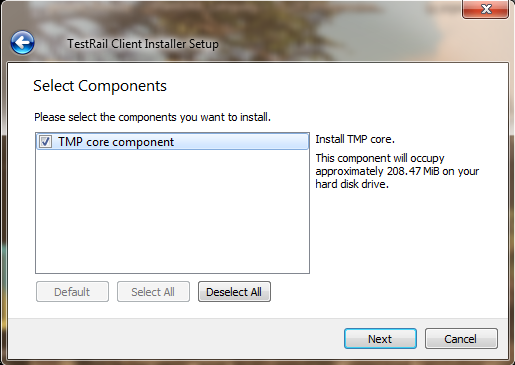
The default install path will be <User home>/TestRail\_Client. You can select another path with “Browse” button.



1. Select install folder

Step 5: Select installed components

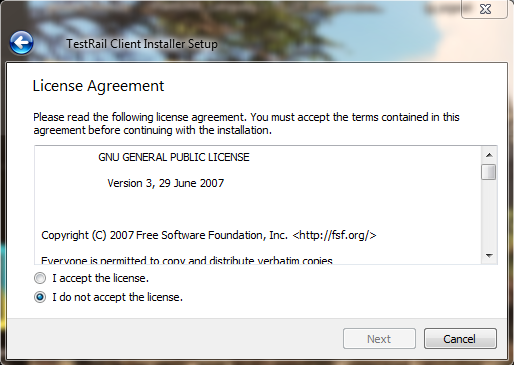
Select install components, the core component “TMP core component” is the must item to install.



1. Install components select

Step 6 License agreement

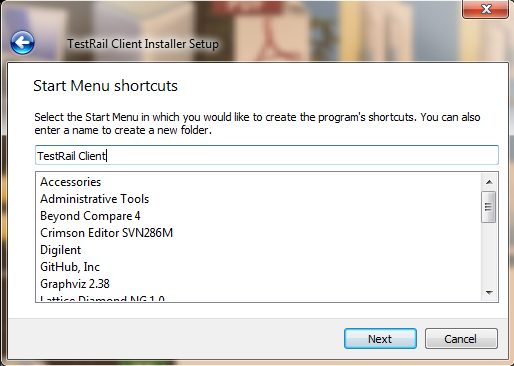
Accept the license before next step.



1. Accept the license

Step 7 Create Start Menu shortcuts (Windows)

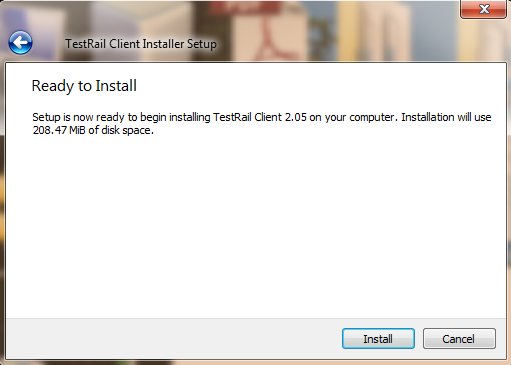
On Windows platform, Setup Wizard will create shortcuts in “Start Menu” for the future launch.



1. Start Menu shortcut create

Step 8 install

Click “install” button to install software.



1. Install software

### Manually install

If you download a source package like: tmp\_client\_2.06\_installer\_src\_all.tar.gz

You can use 7zip extract the source code on windows or tar command line (tar -xv -f tmp\_client\_2.06\_installer\_src\_all.tar.gz) on Linux.

Place the source folder as you want.

### Launch TMP Client

#### 5.3.4.1 TMP Client initial settings (console mode)

This section is for TMP Client run in console mode only. For GUI mode initial setting will be launched after GUI starts.

We can find the initial settings from: <client\_install\_path>/conf/default.ini.

The default.ini file contains two kinds of sections:

1. Section without “tmp\_” : Client software setting sections
2. Section with “tmp\_” : Client setting sections.

##### 5.3.4.1.1 Software setting section

Please use software name as the section name.

max\_insts : Used to define the maximum instance number this software can be launched.

scan\_dir : Client will automatically scan this path to add new software build path.

version and path pair: user specifies the software version and the path manually.

You can specify as many version path pairs as you like.

##### 5.3.4.1.2 Client setting section

Group: used to set the client group name.

Private: The value can be “0” or “1”. The default value is “1” if option not present.

“0”: public client will run assignment and match task.

“1”: private client will run assignment task only.

Terminal: client name setting, the client will scan machine name if option not present.

Unattended Mode: The value can be “0” or “1”. The default value is “0” if option not present.

“0”: Attended Mode. SW will do some special work (i.e. SW update) under user’s operation.

“1”: Unattended Mode. SW will do some special work (i.e. SW update) automatically.

##### 5.3.4.1.3 Preference setting section

Link\_mode: value can be “both”, “remote” and “local”. Client linked server select.

Max\_threads: Maximum threads (test cases) can be launched.

Task\_mode: auto, serial and parallel.

Auto: client will take the highest priority task and run it.

Serial: client will run task one by one according to the task priority

Parallel: client will run tasks in parallel mode

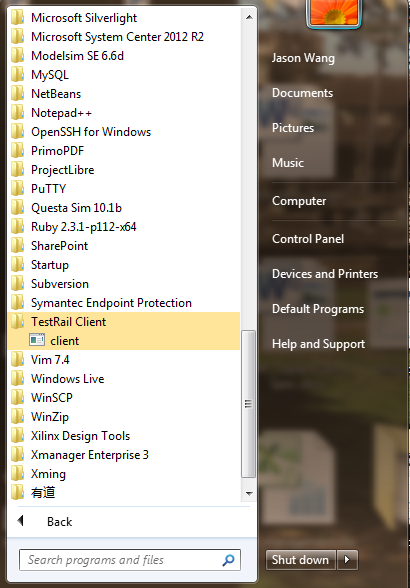
Work\_path: client work space, client will download and run cases in this folder.

Save\_path: client result save path. If not present client will skip copy case runtime result.

#### 5.3.4.2 Launch from Start Menu (Windows)

On Windows platform we can launch TMP client from start menu.

Please click “Start Menu” 🡪 “All Programs” 🡪 “TestRail Client” 🡪 “client”



1. Launch from start menu

#### 5.3.4.3 Launch from command line

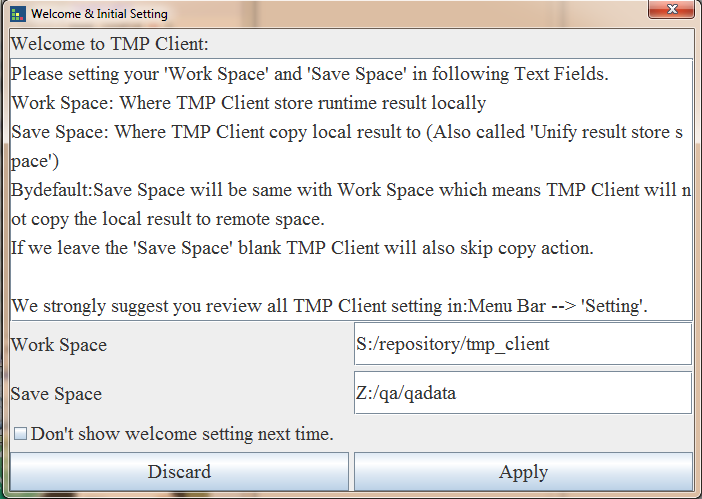
Please use following command line to launch client:

1. Java –jar <client\_install\_path>/bin/client.jar –c (console mode)
2. Java –jar <client\_install\_path>/bin/client.jar (GUI mode)

#### 5.3.4.4 TMP Client initial setting (GUI mode)

##### 5.3.4.4.1 Welcome setting

You will be prompt a welcome setting dialog like following:



1. Welcome setting

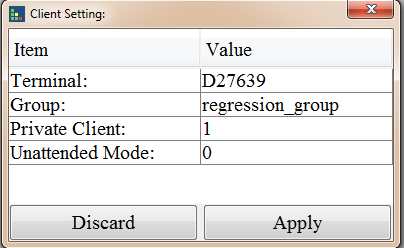
You can specify ‘work space’ and ‘save path’ and click ‘Apply’ button to finish the setting.

Also you can click the “Don’t show welcome setting next time” check box to disable this dialog prompt next time.

##### 5.3.4.4.2 Client setting

Click “Setting” 🡪 “Client…”

Run client setting in GUI way. Please refer [5.3.4.1.2](#_5.3.4.1.2_Client_setting) for details

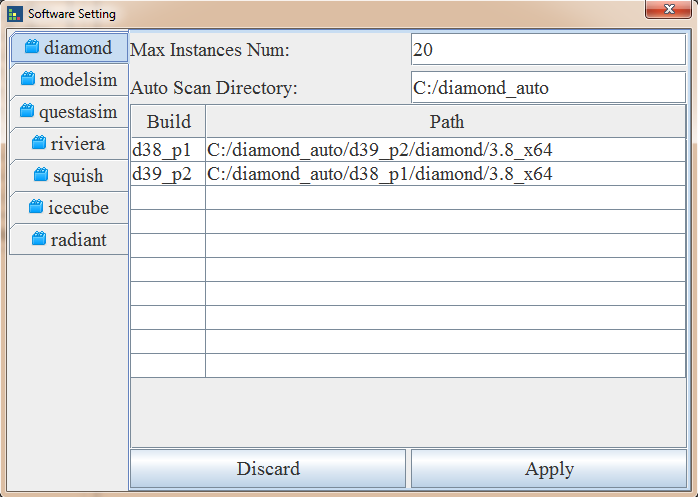


1. Client Setting

##### 5.3.4.4.3 Software setting

Click “Setting” 🡪 “Software…”

Run software setting in GUI way. Please refer [5.3.4.1.1](#_5.3.4.1.1_Software_setting) for details

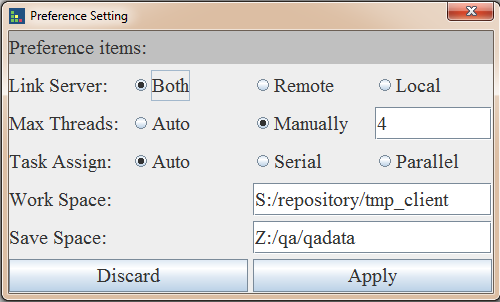


1. Software Setting

##### 5.3.4.4.4 Preference setting

Click “Setting” 🡪 “Preference…”

Run Preference setting in GUI way. Please refer to [5.3.4.1.3](#_5.3.4.1.3_Preference_setting) for details



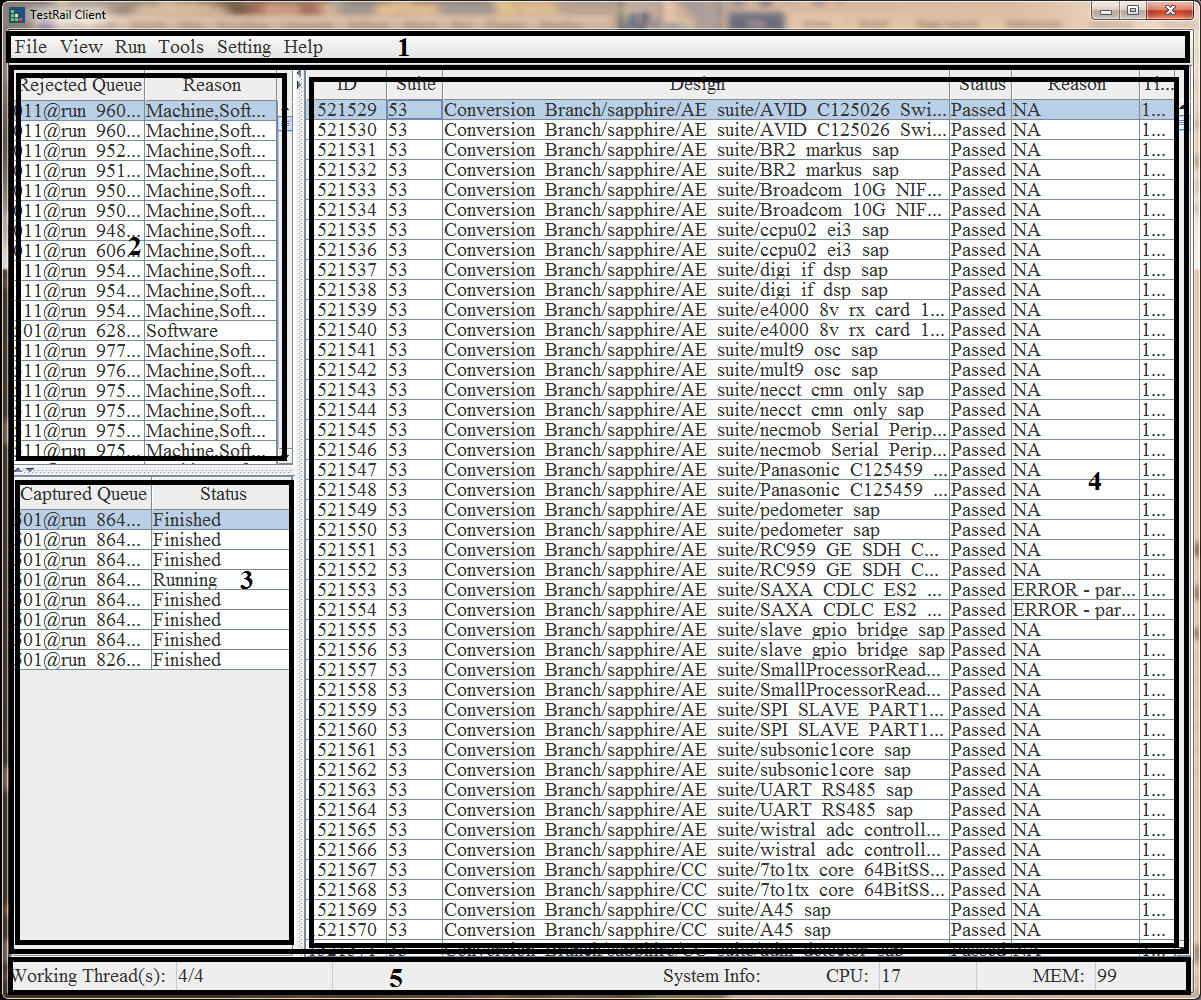
1. Preference Setting

### TMP Client usage

#### 5.3.5.1 Main window introduction

Client main view constructed by 5 sections:

1. Menu bar
2. Rejected queue panel
3. Captured queue panel
4. Watching task panel
5. Status bar



1. Client main view

#### 5.3.5.2 Menu bar

##### 5.3.5.2.1 File menu

“Import…”: Allow you import and run a standard test suite(suite Excel file)

“Export…”: Export test report, not ready now.

“Exit”: Exit/Close software. (If you just click close button on the main view, the client will just close the window and hide. You can re-open the client window from system tray.)

##### 5.3.5.2.2 View menu

Actions for “watching task panel (4)”:

“All”: Show all task cases in this task.

“Waiting”: Show waiting task cases in this task.

“Processing”: Show processing task cases in this task.

“Failed”: Show failed task cases in this task.

“Passed”: Show passed task cases in this task.

“TBD”: Show TBD task cases in this task.

“Timeout”: Show timeout task cases in this task.

##### 5.3.5.2.3 Run menu

Actions for “Captured queue panel (3)” and “watching task panel (4)”

“Play”: action for “Captured queue panel (3)”, when there is a captured task selected, “Play” will restart this task.

“Pause”: action for “Captured queue panel (3)”, when there is a captured task selected, “Pause” will pause this task.

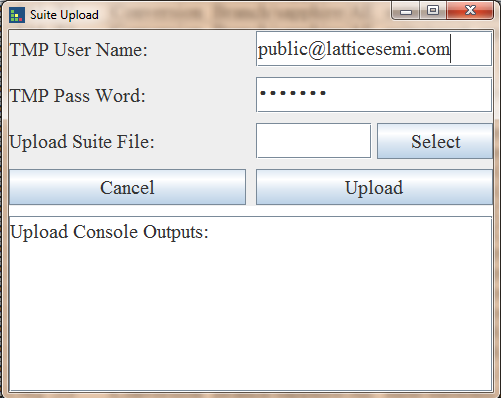
“Stop”: action for “Captured queue panel (3)”, when there is a captured task selected, “Stop” will stop this task.

“Retest” action for “watching task panel (4)”. Retest the test case in watching task according with the condition selected.

##### 5.3.5.2.4 Tools menu

###### 5.3.5.2.4.1 Upload

Integration tool allow you to update standard test suite to TMP server.



1. Suite upload dialog

Please input user name and pass word given by TMP Admin.

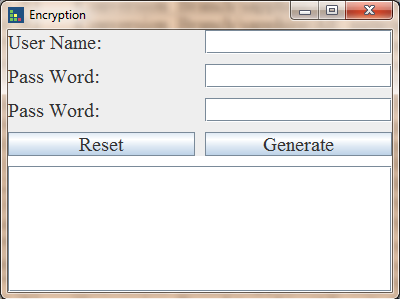
Use “Select” button to select test suite to be uploaded to server.

Click “Upload” button to start upload.

All output lines will be shown in “Upload Console Outputs” window.

###### 5.3.5.2.4.2 Keygen

Integration tool allow user to generate encryption code for user account and password which will be show on TMP website (auth\_key = xxx)



1. Key generate

##### 5.3.5.2.5 Setting menu

Client setting, refer to [5.3.4.4](#_5.3.4.4_TMP_Client) section

##### 5.3.5.2.6 Help menu

“Usage”: show this help document.

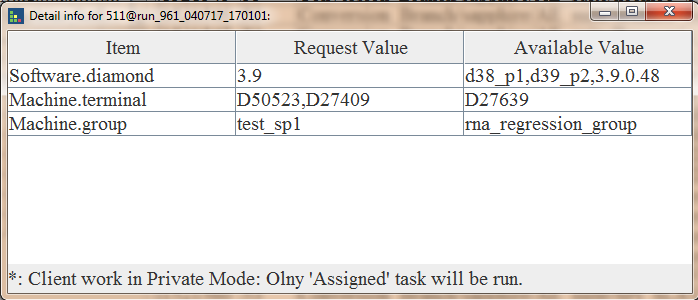
“Contact”: create email and send to Client developer.

“About”: show client version information

#### 5.3.5.3 Rejected queue panel

All rejected task will be shown here with queue name and rejected reason.

You can select one queue and make a right click to popup a “popup menu”, click “Details” to show details information:



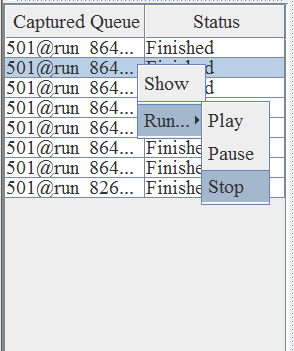
1. Detail rejects view

You will be shown request values and current client value.

#### 5.3.5.4 Captured queue panel

All captured tasks will be shown here with queue name and current status.

You can select one queue and make a right click to popup a “popup menu” like following:



1. Captured panel popup menu

“Show”: click this button will show detail run case info in “watching task panel (4)”

“Run 🡪 Play”: same function as 5.3.5.2.3 Run menu, make the select captured queue re-start

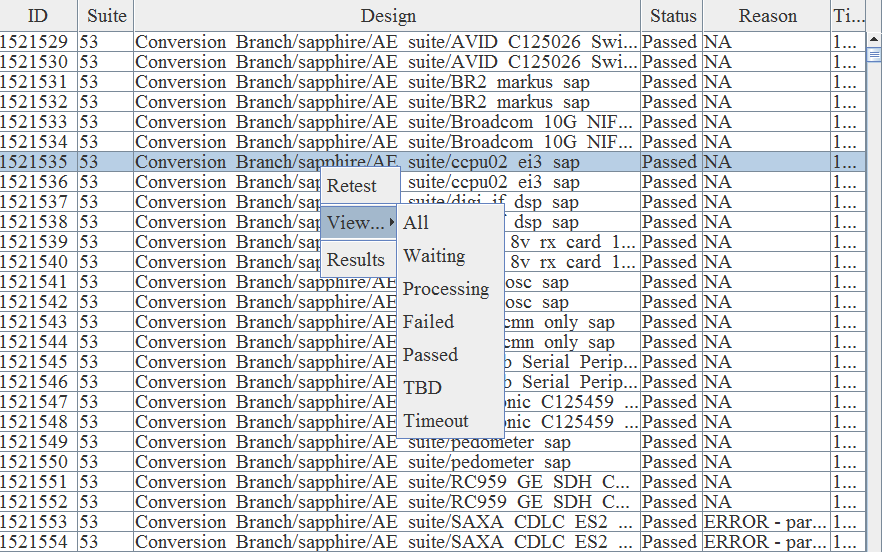
“Run 🡪 Pause”: same function as 5.3.5.2.3 Run menu, make the select captured queue pause

“Run 🡪 Stop”: same function as 5.3.5.2.3 Run menu, make the select captured queue stop

#### 5.3.5.5 Watching task panel

Watching task panel will show task queue data.

You can select one task case and make a right click to popup a “popup menu” like following:



1. Task panel popup menu

“Retest”: will retest this case.

“View”: same function as 5.3.5.2.2 view menu.

“Results”: open the task case result folder.

#### 5.3.5.6 Status bar

Status bar will show you current running threads number and current system info like MEM and CPU.

### Simple usage

#### 5.3.6.1 Launch a local test suite

Step 1: make sure your client not running in remote mode (Setting 🡪 Preference 🡪 link server)

Step 2: click File 🡪 import…

Step 3: select your test suite file

#### 5.3.6.2 Launch remote test suite

Make sure your client not running in local mode (Setting 🡪 Preference 🡪 link server).

#### 5.3.6.3 Report output

Report export is not ready now.

Workaround: Manually copy task case status in “Watching task panel (4)”

# Project level test with TMP Website

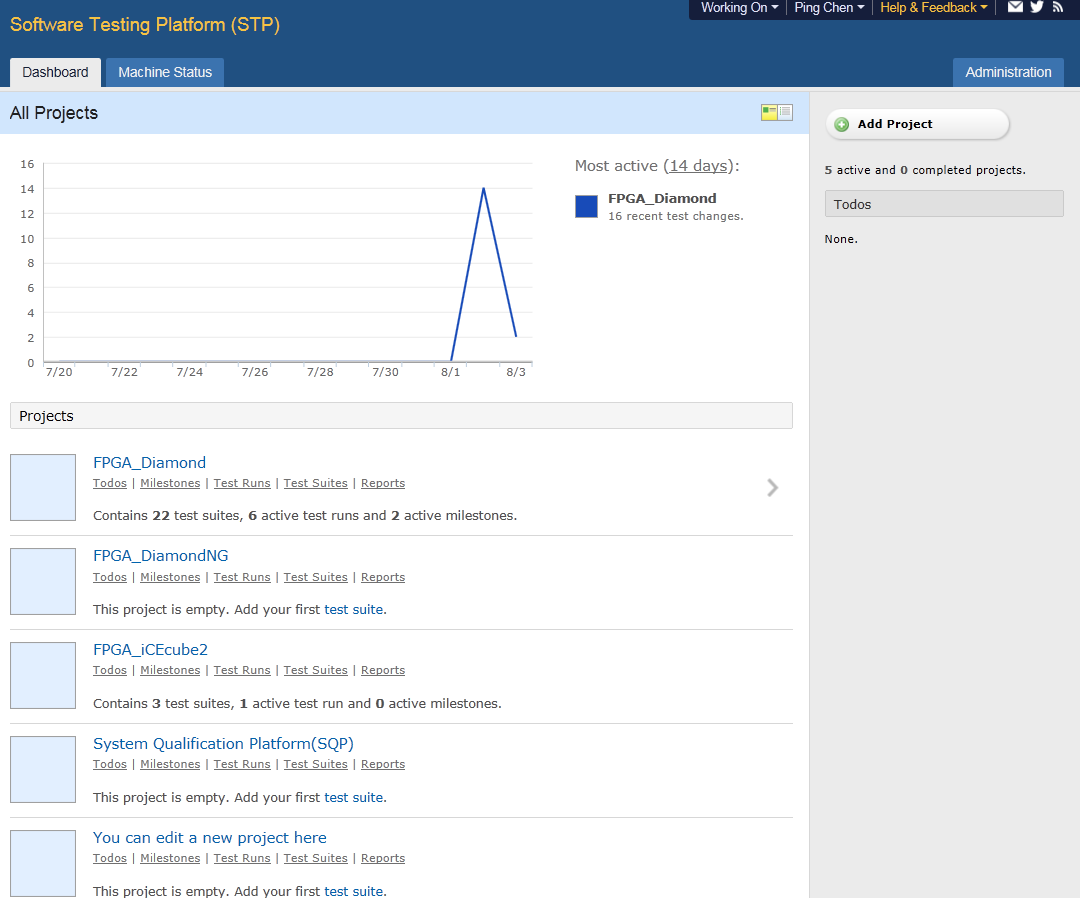
In this section, some typical usages on TMP website will be introduced based on user perspective. All website management usages will not be shown here.

Assuming user:

1. Already have a TMP web account for following actions (if not please contact your local Admin user).
2. Already upload the local suite information to TMP website (if not please refer [5.3.5.2.4.1](#_5.3.5.2.4.1_Upload) for how to upload a local suite to website)

## TMP website overview

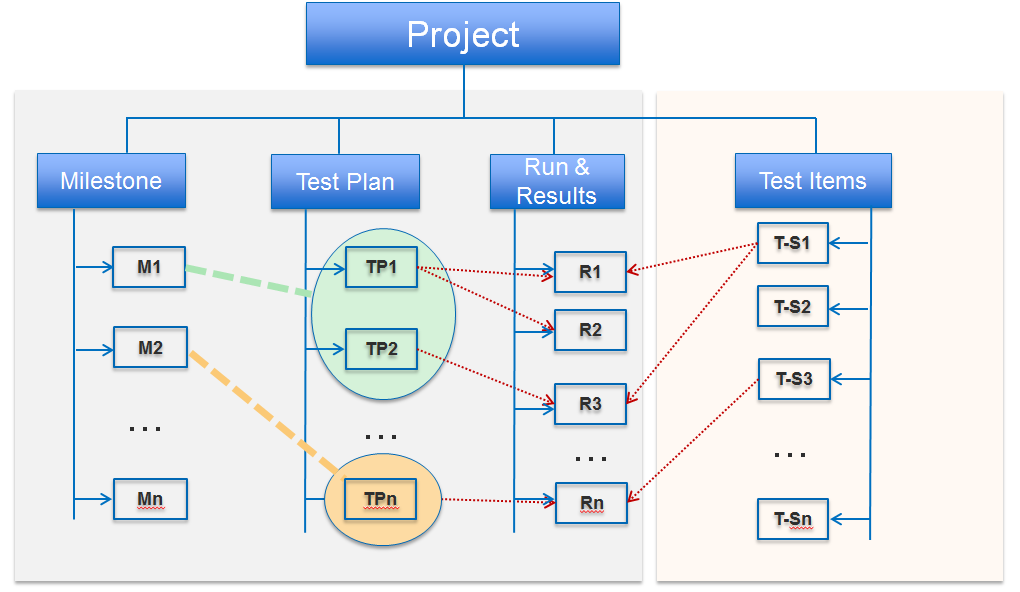
After log in to TMP, the first page is the dashboard. The dashboard provides user with an overview of available projects, recent project activities and the todos. When user navigates to a project (by clicking on a project title), then switch to the project view and land on the project's overview page, showing project details such as test suites, active test runs, project activity and so on. Whenever user needs to switch to another project, just return to the dashboard by clicking the link in the upper left corner.



1. TMP overview

### TMP Management Structure

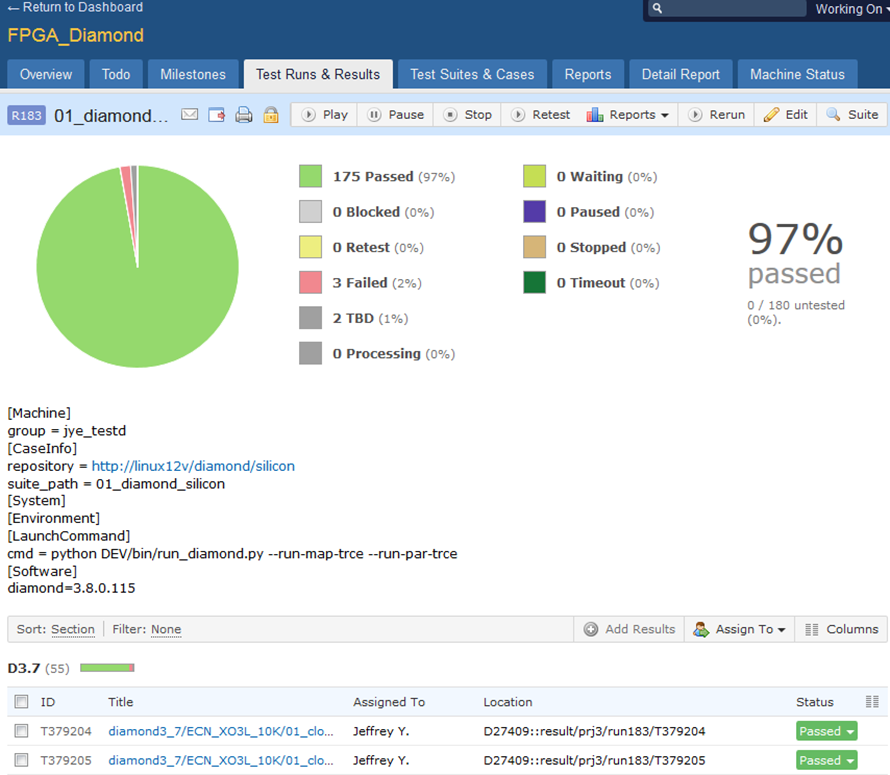
Test Rail is applied as top level TMP, the key elements of test management structure include Project, Milestone, Test Plan, Test Run &Results, Test Suites & Cases.



1. Project management

## How to launch a test suite

To run a test and enter test results for the cases added, user can start a test run for a particular test suite. While a test suite is just like a plan that specifies how an application is tested, a test run is an actual test user is conducting.



1. Launch test run

### Add Test Run

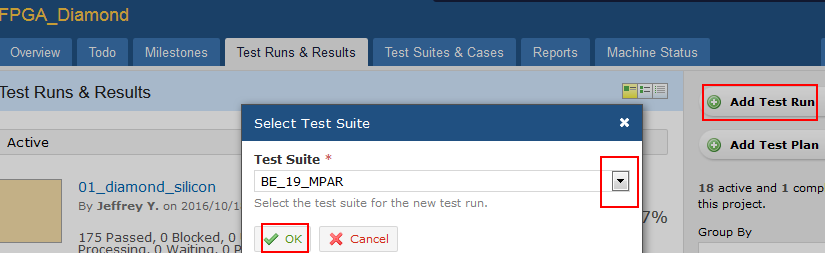
User can create a run by selecting a test suite. In the next page, user can add more customer configurations.

Default run name is suite name.User can use any meaningful name for the run.

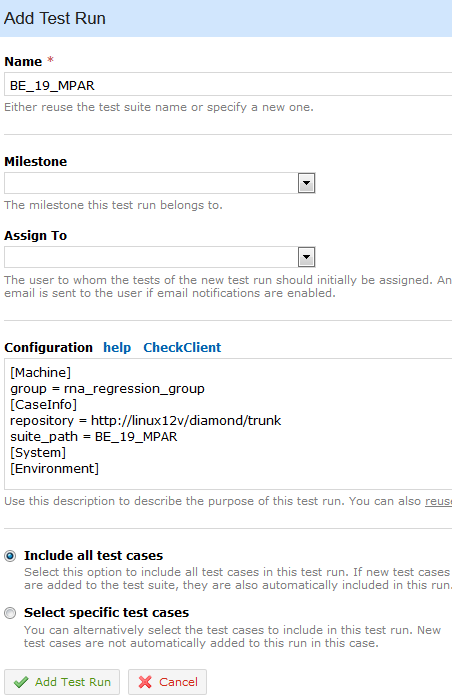
Milestone is a track name for the run.

It’s very convenient to modify the test run’s configuration as user wishes.The “CheckClient” can tell user which clients have been selected to launch the run currently.

User can run all test cases in this suite by “Include all test cases”, otherwise “Select specific test cases” will show a page to select cases by different filters.



1. Add test run



1. Detail add

### Pause/stop/play a test run

The button for “pause/stop/play” is:



1. Buttons for run control

If user wants to “pause” the run and resume the run later , user can click pause button. In a short time the status of the test case will be changed to “Paused” from “Waiting”. Click “Play” button will continue to run this run.

If user wants to stop a run, can click stop button. In a short time the status of all test cases will changed to Stopped. Click “Play” button will rerun all test cases in this run.

### Retest cases in a test run

Click “retest” will provide a page for selecting some cases and the platform will rerun them. The filter is the test cases status name.

### Rerun

The platform will create a totally new run which will inherit all configurations from current run. The rerun function provides some filter for the new run.

## How to launch a test plan

When user needs to manage multiple test runs and configurations for a project, TMP's test plans come in handy. A test plan allows user to start multiple test runs at once, either if user has many test suites or if user want to test against multiple configurations (where a configuration can be anything which needs to test the project against, such as different operating systems or web browsers).

To create a test plan, simply click the “Add Test Plan” button from the Test Runs & Results page. Now just select one or more test suites from the sidebar to add it to the plan. Just like to do with a single test run, user can configure all properties of the test runs such as the name, the cases which are to include and so on.

# Demo work flow

## Test case build

First build test case according to the requirements list in section 4.2.

## Test suite build

Then put all test cases into one folder and make a Microsoft Excel file (you can get a demo suite file from your local admin user).

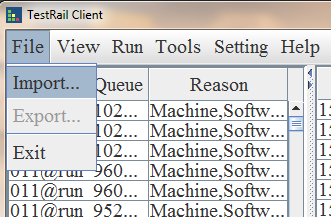
Modify the suite and case sheet based on the requirements list in section 5.2.

## Local test/validation

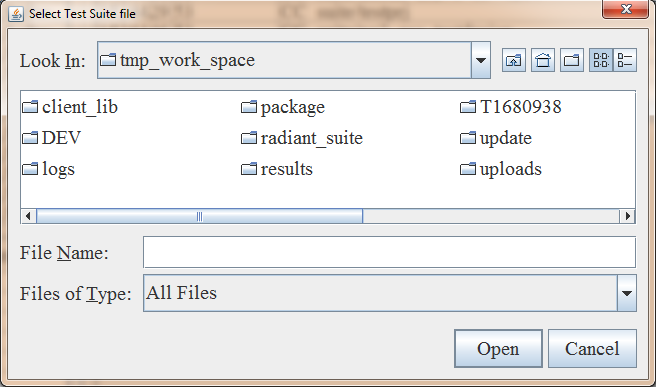
When the suite file is completed we can load it into TMP client to run initial test.

A test suite file is a collection of test cases. Test suite file list all requirements to run these cases.

TMP client will parser these requirements and take the matched cases for future run.



1. Import suite file



1. Select suite file

## Upload test suite to TMP website

After finish local test suite validation, user can upload test suite to TMP webpage with TMP client (see section 5.3.5.2.4)

## Run test suite on TMP webpage

Please refer section 6.2 for how to launch a test suite on webpage.

# Appendix

Software address: [\\lsh-smb01\sw\qa\qa\_store\TMP\_tools\client\client\_software](file:///\\lsh-smb01\sw\qa\qa_store\TMP_tools\client\client_software)

TMP client build: Software QA SVN: /release/tmp\_client/