XSTAF document Documentation

Release pre-alpha

xcgspring

July 28, 2015

Contents:

XSTAF introduce

Page Status Development

Last Reviewed

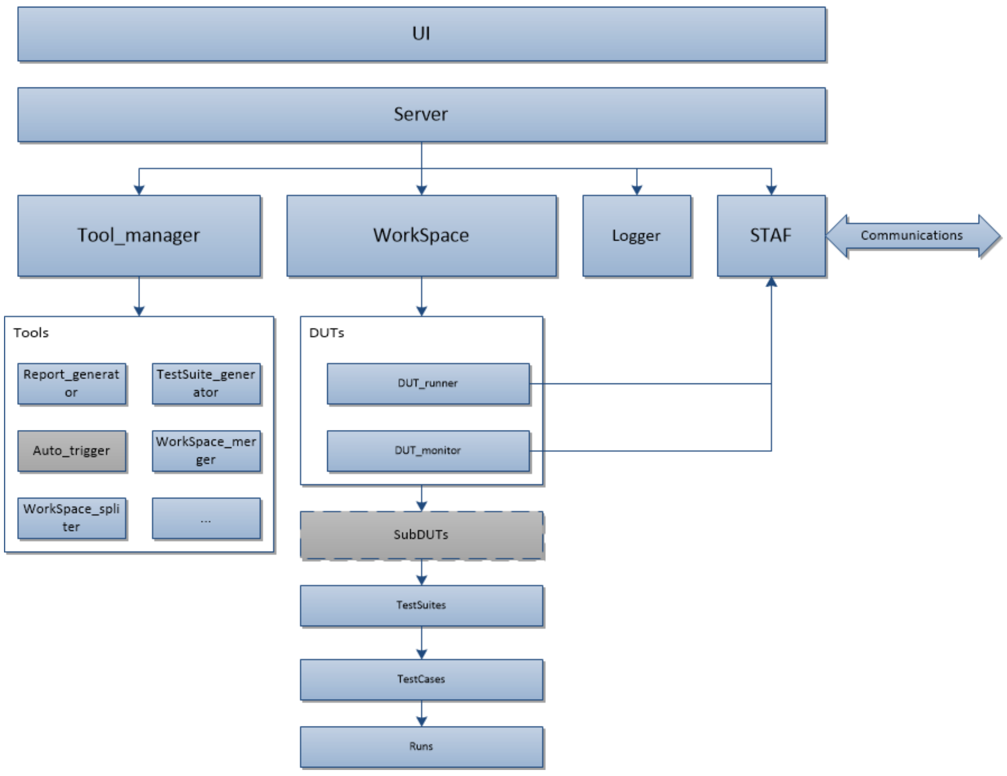
XSTAF is an execution framework providing a bunch of features for testers' convenience:

- XSTAF is a disturbed execution framework based on STAF project, XSTAF can connect and run tests on multiple DUTs at same time
- XSTAF provide test case management, and testers can have detail controls, down to atomic testcases
- XSTAF provide a tool plugin mechanism, runtime data is exposed so user can develop plugins to control XSTAF runtime data
- XSTAF provide some basic tools, like test report generator, test suite generator to handle common met tasks

XSTAF's competition is other test manage and execution frameworks, like robot, spoon, testNG, or other test back-end of CI system.

1.1 XSTAF top view

Below image give an overview of XSTAF modules.



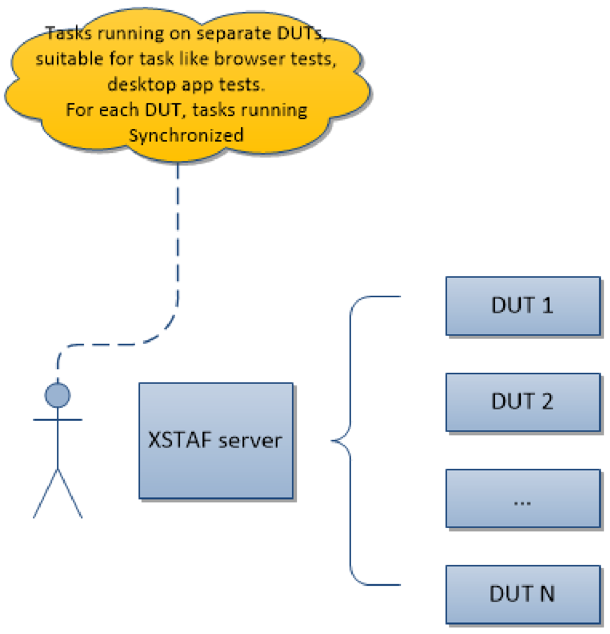
1.2 XSTAF use scenarios

This chapter gives a preview of typical use scenarios XSTAF designed for:

1.2.1 PC like DUTs

Browser tests, desktop app tests, and PC function tests are always running on DUTs just like PC. STAF normally provides support for these platform, and these platform are power enough to support some extra tasks from STAF. So usually we deploy scripts, libraries and STAF client to DUTs directly, and use XSTAF to control the DUT directly.

The use scenario for this kind of tests, you can deploy XSTAF just like below:

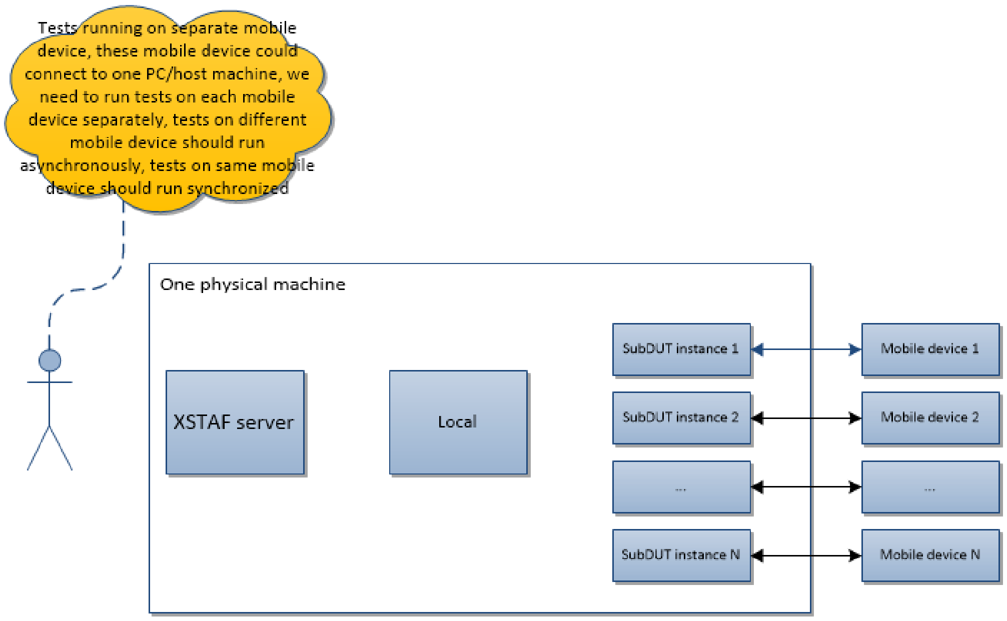

Tasks running on separate DUTs, suitable for task like browser tests, desktop app tests.
For each DUT, tasks running Synchronized
XSTAF server
DUT 1
DUT 2
...
DUT N

Note: For PC like DUTs, XSTAF run tests synchronously, which means these is only one test running on one DUT at some time

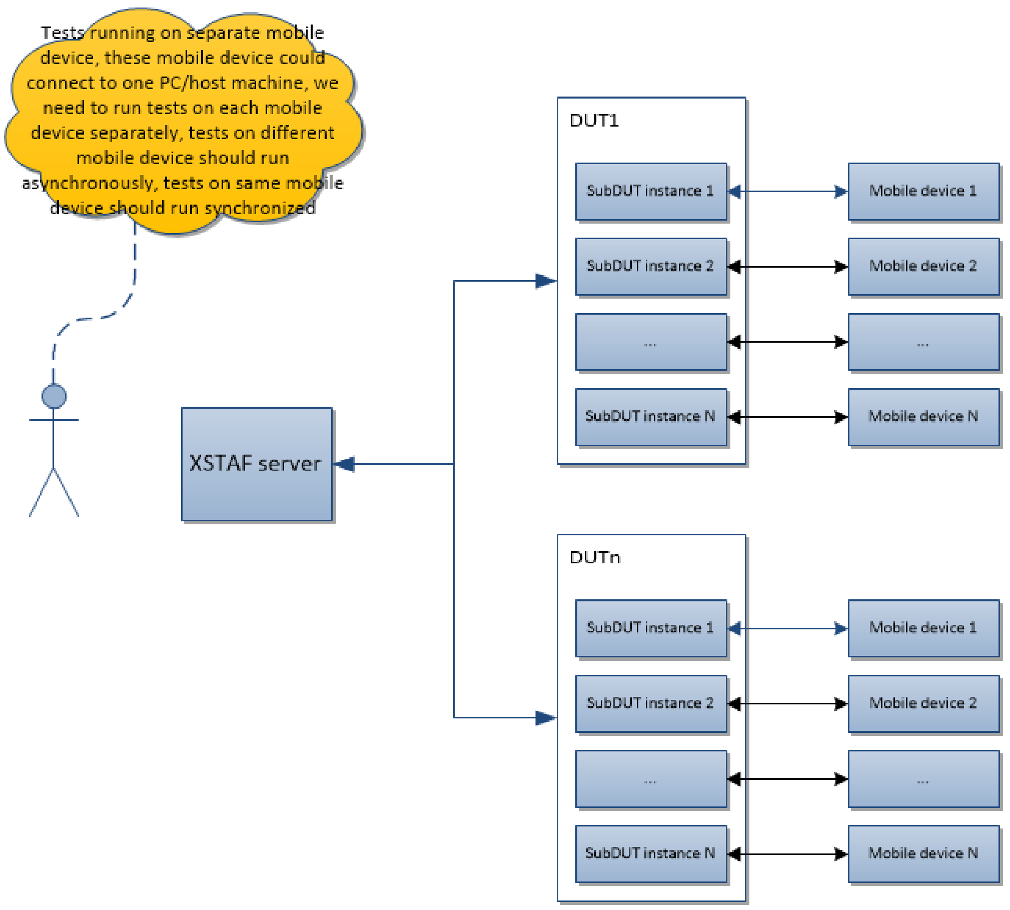
1.2.2 mobile DUTs (not ready yet)

Mobile app tests are running on mobile devices. STAF doesn't support mobile device, and for tests on mobile devices, usually we need an extra PC to setup the test environment, like libraries, platform tools(adb). So XSTAF use a different structure to support tests on mobile devices, like below:

XSTAF server and client running on same machine, with multiple mobile devices


Tests running on separate mobile device, these mobile device could connect to one PC/host machine, we need to run tests on each mobile device separately, tests on different mobile device should run asynchronously, tests on same mobile device should run synchronized
One physical machine
XSTAF server
Local
SubDUT instance 1
SubDUT instance 2
...
SubDUT instance N
Mobile device 1
Mobile device 2
...
Mobile device N

XSTAF server control multiple machines, with multiple mobile devices connect with each machine


Tests running on separate mobile device, these mobile device could connect to one PC/host machine, we need to run tests on each mobile device separately, tests on different mobile device should run asynchronously, tests on same mobile device should run synchronized
XSTAF server
DUT1
SubDUT instance 1
SubDUT instance 2
...
SubDUT instance N
Mobile device 1
Mobile device 2
...
Mobile device N
DUTn
SubDUT instance 1
SubDUT instance 2
...
SubDUT instance N
Mobile device 1
Mobile device 2
...
Mobile device N

Note: For mobile tests, XSTAF can run tests asynchronously, but for one subDUT, tests are running synchronously

CHAPTER 2

XSTAF workspace

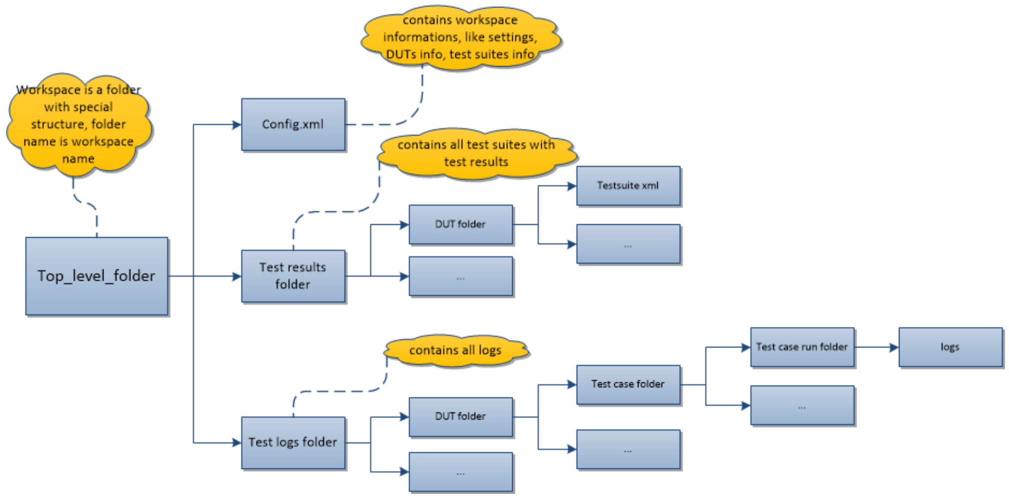
Page Status Development

Last Reviewed

XSTAF provide a mechanism to store a stand-alone test configuration all together (DUTs, testsuites, test results, test logs...). The benefit of this mechanism is you can have distinguishing workspace for different configurations, make your test plans clean and clear.

2.1 XSTAF workspace overview

Currently a workspace is a folder of pre-defined structure, like below:


contains workspace informations, like settings, DUTs info, test suites info
Workspace is a folder with special structure, folder name is workspace name
Config.xml
contains all test suites with test results
Testsuite xml
DUT folder
...
Top_level_folder
Test results folder
...
contains all logs
Test case run folder
logs
Test case folder
...
DUT folder
Test logs folder
...
...

2.2 config.xml

config.xml is workspace config file, contains informations like DUTs info, testsuites info. XSTAF load the DUTs and testsuites according to infomations in this config file.

User need not create a config.xml file. When you new and save a workspace, a config.xml will be auto generated into the workspace folder, containing DUTs and testsuites info in the XSTAF runtime.

A sample config xml like below:

```
<XSTAF>
  <settings>
    <setting name="DefaultWorkspace">.default</setting>
    <setting name="WorkspaceLocation">c:\XSTAF\workspaces</setting>
  </settings>
  <DUTs>
    <DUT ip="10.239.36.123" name="">
      <testsuites>
        <testsuite name="testsuite1.xml" />
      </testsuites>
    </DUT>
    <DUT ip="10.239.36.124" name="">
      <testsuites>
        <testsuite name="testsuite2.xml" />
      </testsuites>
      <testsuites>
        <testsuite name="testsuite3.xml" />
      </testsuites>
    </DUT>
  </DUTs>
</XSTAF>
```

2.3 test results folder

test results folder contains testsuites info and test results for each DUT, test suites and test results are stored with a XML file, format like below:

```
<TestSuite>
  <TestCases>
    <TestCase>
      <ID>19730170-ced9-11e4-bd02-001b211bc956</ID>
      <Name>test_name</Name>
      <Command>C:/python27/python.exe python_script</Command>
      <Auto>False</Auto>
      <Timeout>6000</Timeout>
      <Description>demo test</Description>
      <Runs>
        <Run>
          <Start>1427087989.969</Start>
          <End>1427087995.672</End>
          <Result>fail</Result>
          <Status />
          <Log>C:\XSTAF\workspaces\workspace_name\test_logs\19730170-ced9-11e4-bd02-001b211bc956\142
        </Run>
      </Runs>
    </TestCase>
  </TestCases>
</TestSuite>
```

2.4 test logs folder

test logs folder is used to store test logs for each run of test case. Logs location is like:

```
abs_workspaces_location/workspace_folder/test_logs/Test_ID/start_time
```

Note: Test logs include stdout/stderr of test scripts and all files in tmp log location.

CHAPTER 3

XSTAF test management

Page Status Development

Last Reviewed

XSTAF manage test cases via a three level structure:

- **test suite**, test suite is a test case container, could contain multiple test cases, is used to separate test cases into several categories, make test cases well organized.
- **test case**, represent a basic test unit, contains information of how to run a test case
- **run**, represent one execution of one test case, contains information of execution results

3.1 Test suite format

Test suite is a XML document with pre-defined format, like below:

```
<TestSuite>
  <TestCases>
    <TestCase>
      <ID>19730170-ced9-11e4-bd02-001b211bc956</ID>
      <Name>sample_test_case</Name>
      <Command>run_test_script_command</Command>
      <Auto>true</Auto>
      <Timeout>6000</Timeout>
      <Description>This is a sample</Description>
      <Runs>
        <Run>
          <Start>1427087989.969</Start>
          <End>1427087995.672</End>
          <Result>fail</Result>
          <Status />
          <Log>log_location</Log>
        </Run>
      </Runs>
    </TestCase>

  </TestCases>
</TestSuite>
```

A test suite can contain multiple test case, a test case can contain multiple runs.

Test case contains informations as below:

- **ID**, a global unique GUID for this test case, recommend use a GUID generator to generate an unique GUID for this test case.
- **Name**, tester friendly name to recognize this test case.
- **Command**, command to run the test case, recommend to write test script using xUnit framework.
- **Auto**, for auto case, XSTAF judges the pass/fail by command return value, return 0 is pass, other is fail. For manual case, XSTAF will prompt a dialog to let tester judge the pass/fail after command return.
- **Timeout**, if test script running longer than timeout, XSTAF will terminate the test case, and continue run other test case
- **Description**, test case description, you can fill in all related information useful.

Run contains informations as below:

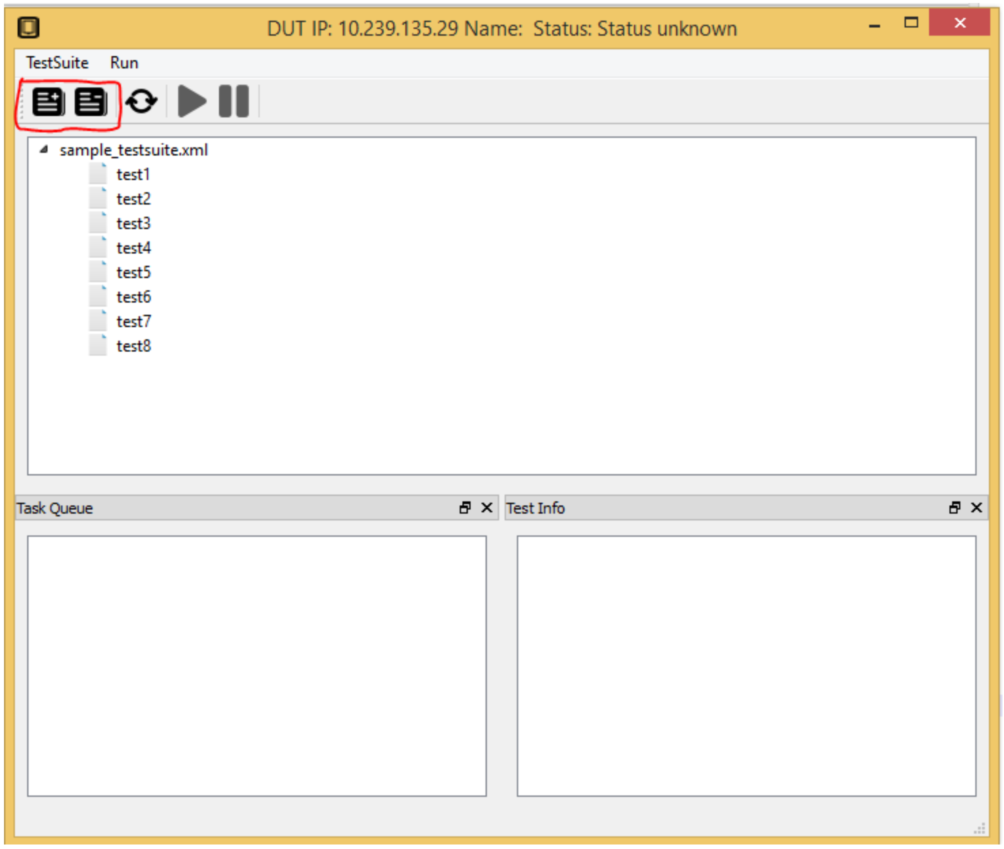
- **Start** run start time
- **End** run end time
- **Result** test result, pass/fail/no run
- **Status** run status information, may contain informations about test issues
- **Log** Log location, include stdout/stderr of test script, and other log files

Note: You can manually write your own test suite.

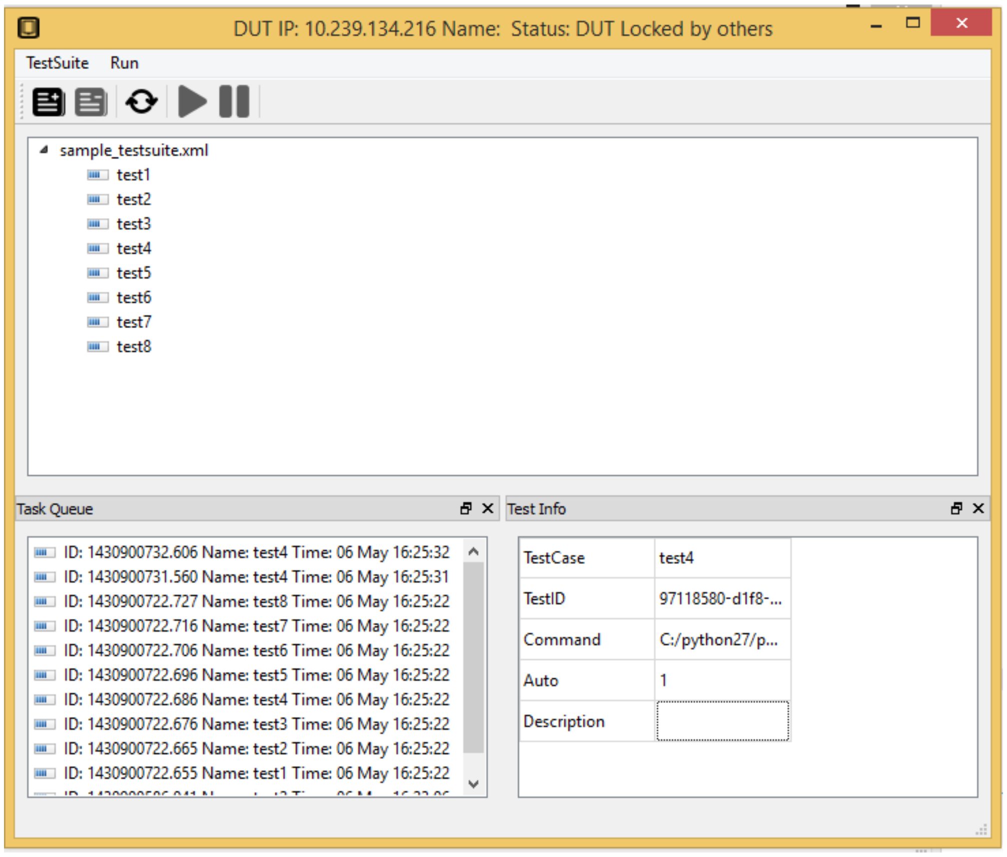
Recommend to write a translate tool, export test cases from test manage system (like testlink, HPQC...) and convert the these exported test cases to XSTAF test suite.

3.2 Test management under DUT view

After you add DUTs into one workspace, you can open DUT view, and add test suites to DUT.



After add test suites to DUT, you can select which tests you want to run, and add them to DUT task queue. XSTAF support add test case to task queue, or add whole test suite to task queue(all test cases in task queue will be added to task queue).

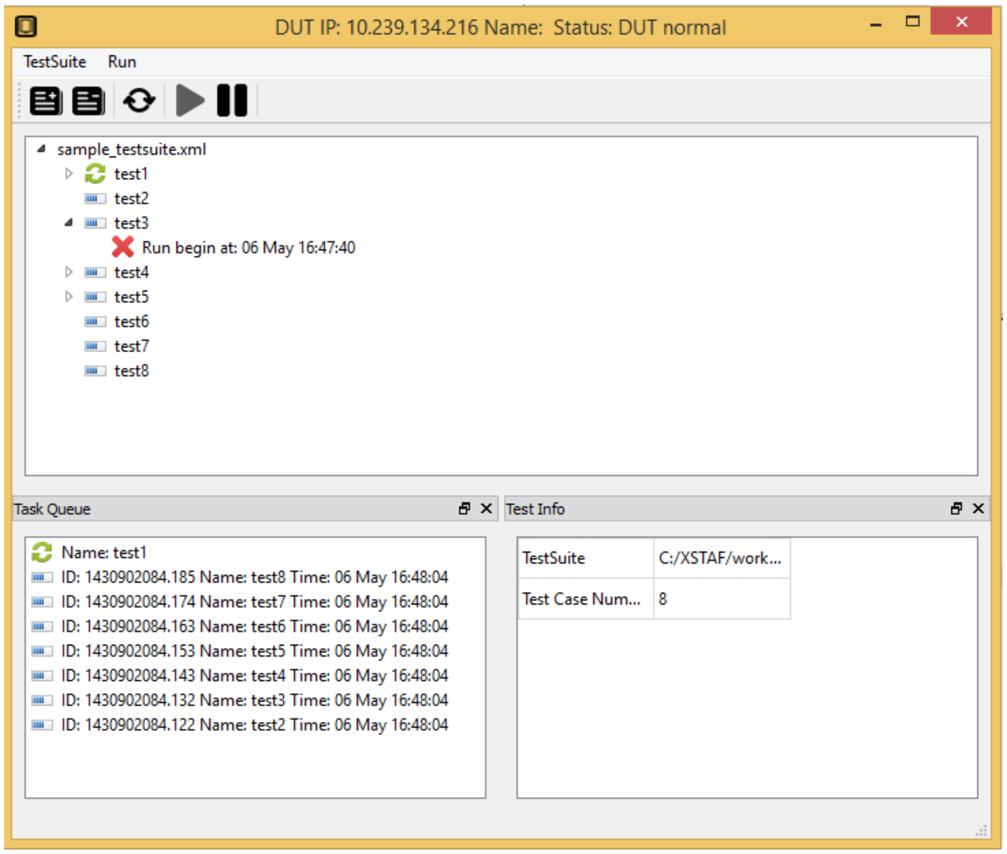


sample_testsuite.xml
test1
test2
test3
test4
test5
test6
test7
test8

```
ID: 1430900732.606 Name: test4 Time: 06 May 16:25:32
ID: 1430900731.560 Name: test4 Time: 06 May 16:25:31
ID: 1430900722.727 Name: test8 Time: 06 May 16:25:22
ID: 1430900722.716 Name: test7 Time: 06 May 16:25:22
ID: 1430900722.706 Name: test6 Time: 06 May 16:25:22
ID: 1430900722.696 Name: test5 Time: 06 May 16:25:22
ID: 1430900722.686 Name: test4 Time: 06 May 16:25:22
ID: 1430900722.676 Name: test3 Time: 06 May 16:25:22
ID: 1430900722.665 Name: test2 Time: 06 May 16:25:22
ID: 1430900722.655 Name: test1 Time: 06 May 16:25:22
```

TestCase	test4
TestID	97118580-d1f8-...
Command	C:/python27/p...
Auto	1
Description	

You can start the task queue if DUT status is normal, test case in task queue will be executed, and a new run will be added to each test case.



sample_testsuite.xml
test1
test2
test3
Run begin at: 06 May 16:47:40
test4
test5
test6
test7
test8

Name: test1
ID: 1430902084.185 Name: test8 Time: 06 May 16:48:04
ID: 1430902084.174 Name: test7 Time: 06 May 16:48:04
ID: 1430902084.163 Name: test6 Time: 06 May 16:48:04
ID: 1430902084.153 Name: test5 Time: 06 May 16:48:04
ID: 1430902084.143 Name: test4 Time: 06 May 16:48:04
ID: 1430902084.132 Name: test3 Time: 06 May 16:48:04
ID: 1430902084.122 Name: test2 Time: 06 May 16:48:04

TestSuite	C:/XSTAF/work...
Test Case Num...	8

CHAPTER 4

XSTAF tool management

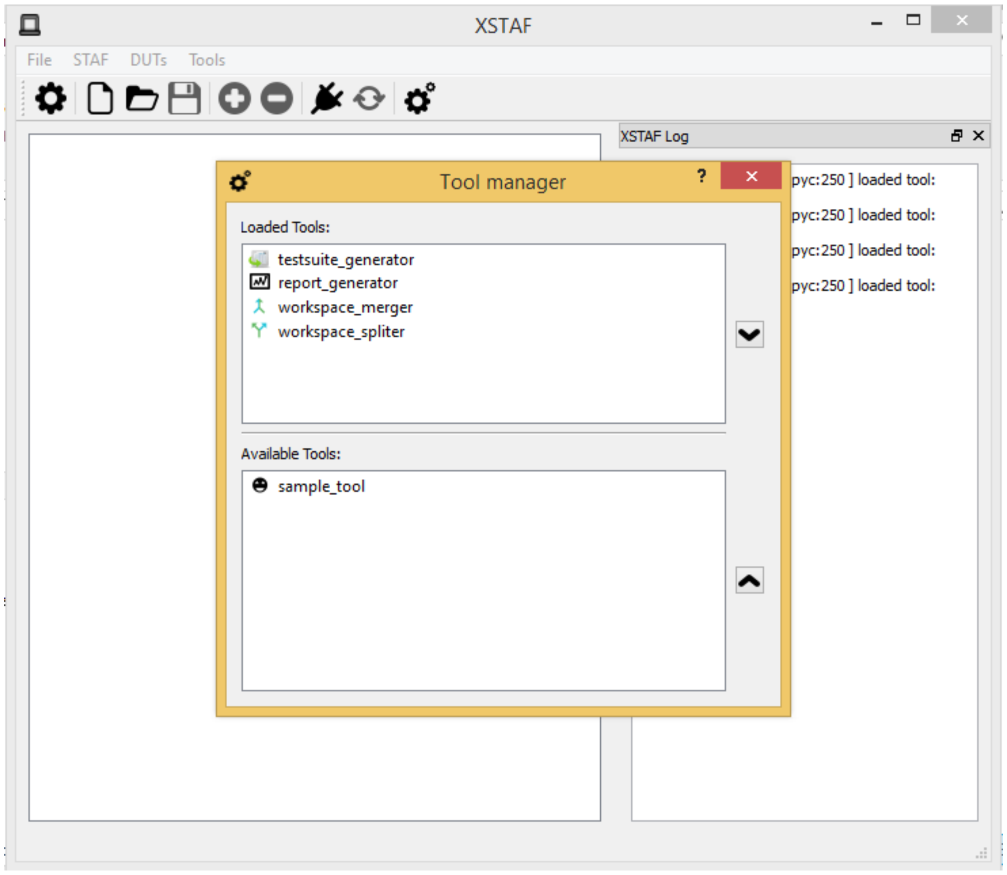
Page Status Development

Last Reviewed

XSTAF provide a plugin mechanism to support extending XSTAF functions.

4.1 Tool manager

Tool manager is response to check the tool directory, and load the tools available. Tool manager supports load the tool dynamically, and provide tools runtime XSTAF window instance object, so tools can interact with XSTAF core.



```
pyc:250 ] loaded tool:
pyc:250 ] loaded tool:
pyc:250 ] loaded tool:
pyc:250 ] loaded tool:
```

4.2 Add custom tools

To write custom tools, you just need implement an standard interface provided by tool manager, and place your tool to `tools` directory.

Below is code from `tools/sample_tool`, it demonstrates how to implement a sample interface:

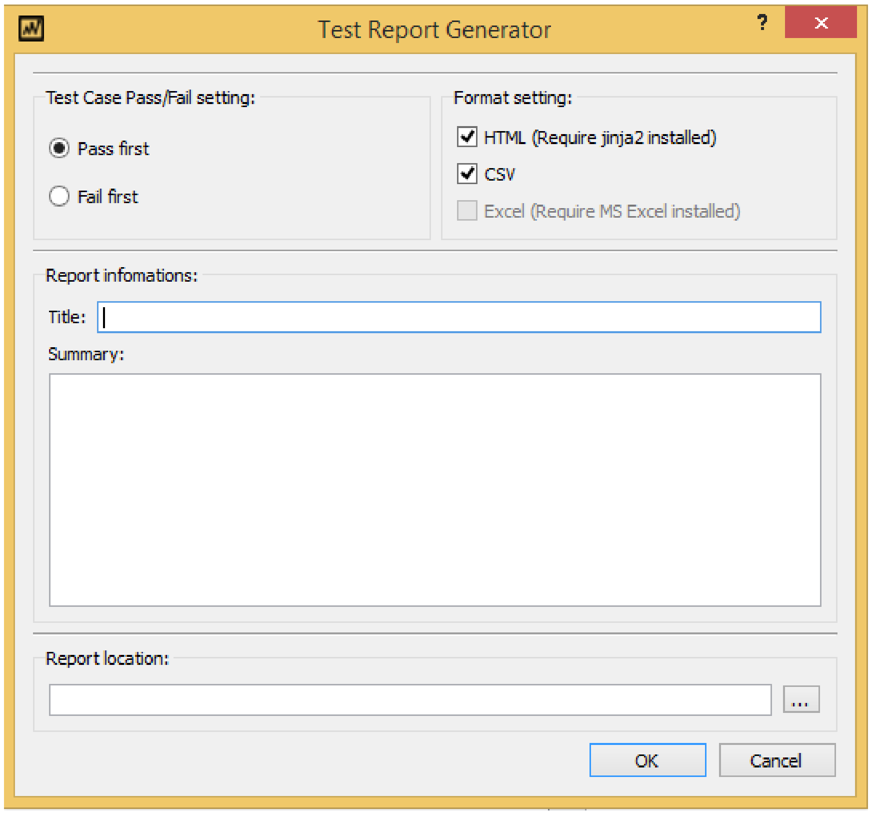
```
import traceback
from XSTAF.core.logger import LOGGER
from PyQt4 import QtCore, QtGui
from ui.ui_sampleDialog import Ui_SampleDialog
import ui.resources_rc

###############################
#interface for tool manager
###############################
class Tool(object):
    _description = "Sample Tool"
    main_window = None

    #this method is for tool manager to pass in the XSTAF main window runtime object
    #so in this tool, we can interact with XSTAF main window
    @classmethod
    def set_main_window(cls, main_window):
        cls.main_window = main_window

    #method for tool manager to get icon of this tool
    @staticmethod
    def icon():
        tool_icon = QtGui.QIcon()
        tool_icon.addPixmap(QtGui.QPixmap(":icons/icons/sample.png"))
        return tool_icon

    #method for tool manager to launch this tool's dialog
    @classmethod
    def launch(cls):
        try:
            LOGGER.info("Launch sample tool")
            tool_dialog = SampleTool(cls.main_window)
            tool_dialog.exec_()
        except:
            LOGGER.error(traceback.format_exc())

    #method for tool manager to get description of this tool
    @classmethod
    def description(cls):
        return cls._description

################################
#dialog class for this tool
################################
class SampleTool(QtGui.QDialog, Ui_SampleDialog):
    def __init__(self, parent):
        QtGui.QDialog.__init__(self)
        self.setupUi(self)
```

4.3 Built-in Tools

XSTAF provide some built-in tools for common met test functions.

4.3.1 report_generator

report generator can generate a static html report for current workspace.



Test Report Generator

Test Case Pass/Fail setting:

[✓ Pass first] Fail first

Format setting:

- [✓] HTML (Require jinja2 installed)
- [✓] CSV
- [ ] Excel (Require MS Excel installed)

Report infomations:

Title:

Summary:

Report location:

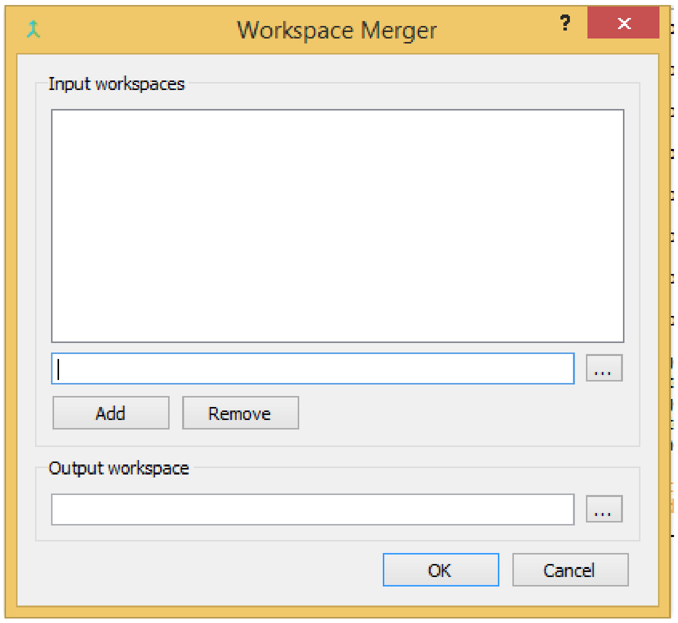
4.3.2 testsuite_generator

test suite generator can generate XSTAF test suite XML file from other format, like a CSV document.

Plan to support formats fro some common used QC system, like testlink...

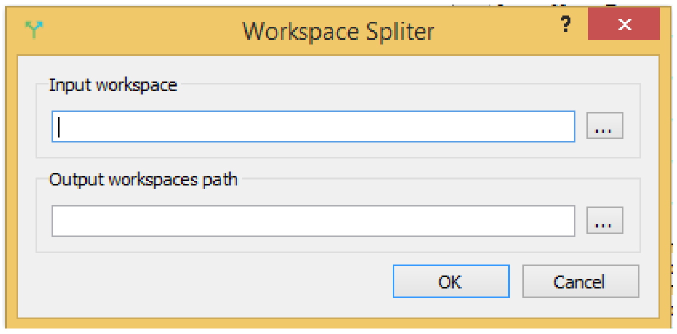
4.3.3 workspace_merger

workspace merger supports merging multiple workspaces to one workspace.



4.3.4 workspace_spliter

workspace merger supports splitting one workspace with multiple DUTs into multiple workspaces with one DUT.



CHAPTER 5

Use XSTAF

Page Status Development

Last Reviewed

5.1 Prepare XSTAF environment

5.1.1 Prepare XSTAF server

1. Install STAF
2. Install XSTAF

- For windows, you can download and use binary app
- For other system, `pip install XSTAF`

5.1.2 Prepare XSTAF DUTs

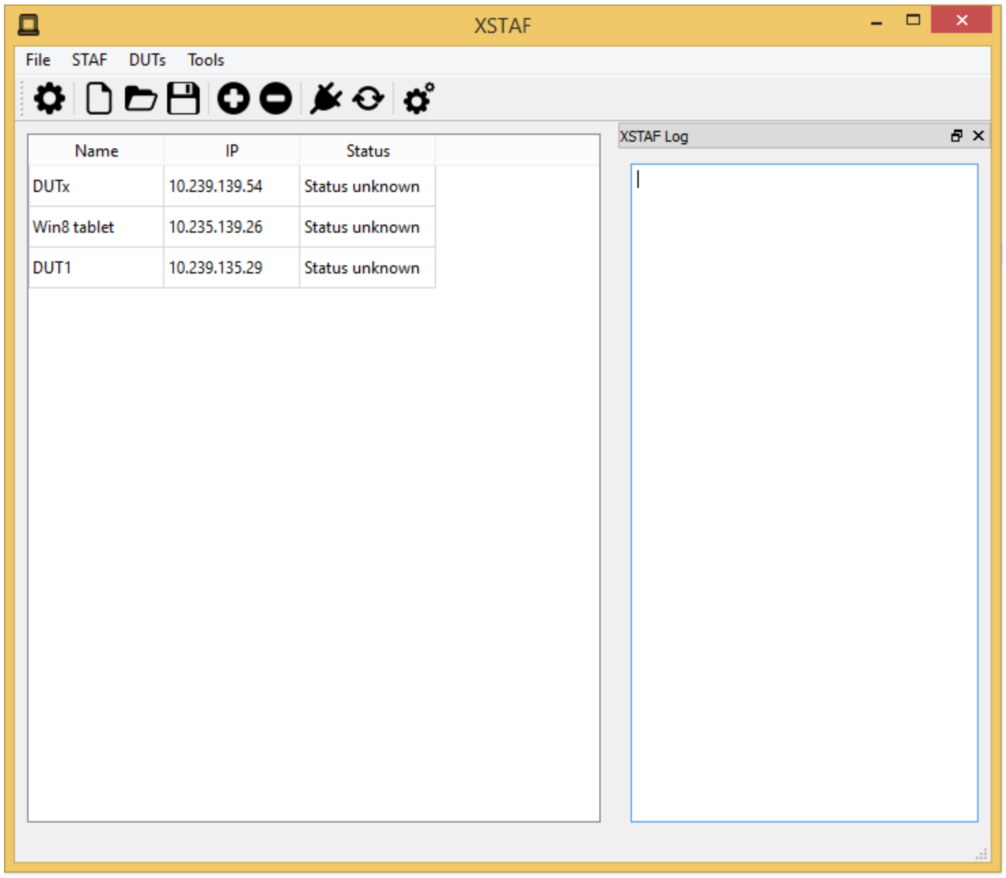
1. Install STAF
2. Set proper trust level, add one line `trust default level 5` in STAF config file `STAF_dir/bin/staf.cfg`

5.2 XSTAF UI preview

5.2.1 Server view

XSTAF has one server view per instance, server view is used to:

- manage workspace, new/load/save workspace
- manage DUTs for each workspace, add/delete DUT
- settings
- manage tools, like report generator, test suite generator
- start STAF, check all DUTs status via STAF

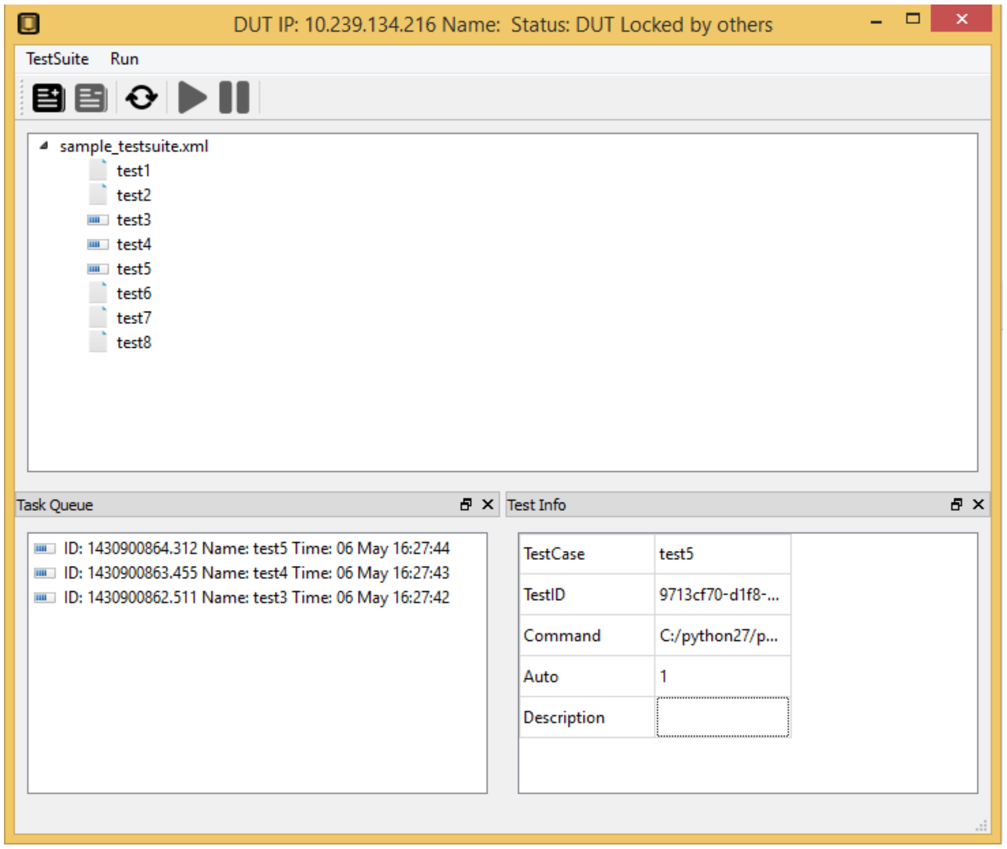


Name	IP	Status
DUTx	10.239.139.54	Status unknown
Win8 tablet	10.235.139.26	Status unknown
DUT1	10.239.135.29	Status unknown

5.2.2 DUT view

XSTAF could have multiple DUT views per instance, DUT view is used to:

- manage test suites for each DUT, add/delete test suite
- run test case, run individual test case/run a whole test suite/rerun fail cases



sample_testsuite.xml
test1
test2
test3
test4
test5
test6
test7
test8

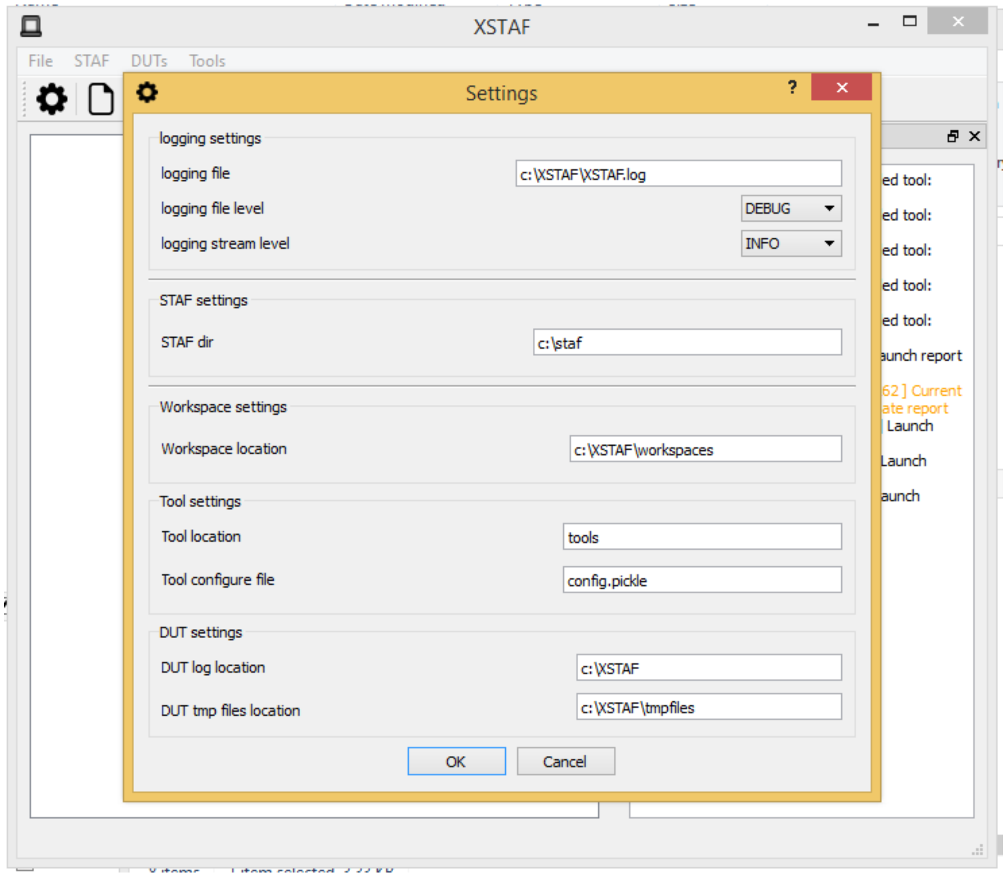
```
ID: 1430900864.312 Name: test5 Time: 06 May 16:27:44
ID: 1430900863.455 Name: test4 Time: 06 May 16:27:43
ID: 1430900862.511 Name: test3 Time: 06 May 16:27:42
```

TestCase	test5
TestID	9713cf70-d1f8-...
Command	C:/python27/p...
Auto	1
Description	

5.3 Use XSTAF step by step

5.3.1 settings

There are some setting you can set for your test environment



logging settings

logging file	c:\XSTAF\XSTAF.log
logging file level	DEBUG
logging stream level	INFO

STAF settings

STAF dir	c:\staf

Workspace settings

Workspace location	c:\XSTAF\workspaces

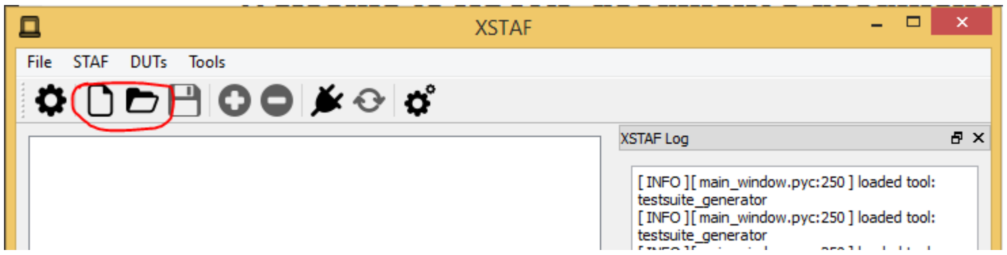
Tool settings

Tool location	tools
Tool configure file	config.pickle

DUT settings

DUT log location	c:\XSTAF
DUT tmp files location	c:\XSTAF\tmpfiles

5.3.2 new or load existing workspace

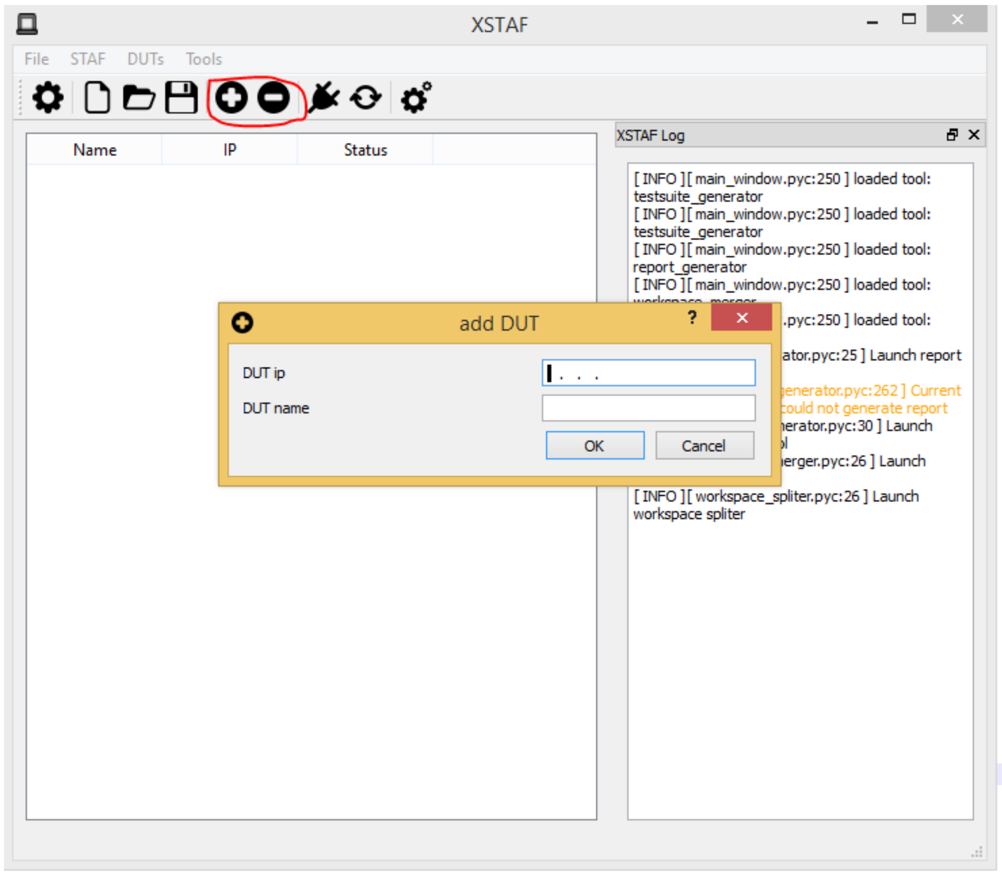


```
[ INFO ][ main_window.pyc:250 ] loaded tool:
testsuite_generator
[ INFO ][ main_window.pyc:250 ] loaded tool:
testsuite_generator
```

5.3.3 add DUTs

After you have new or load a workspace, you can add/delete DUTs into this workspace

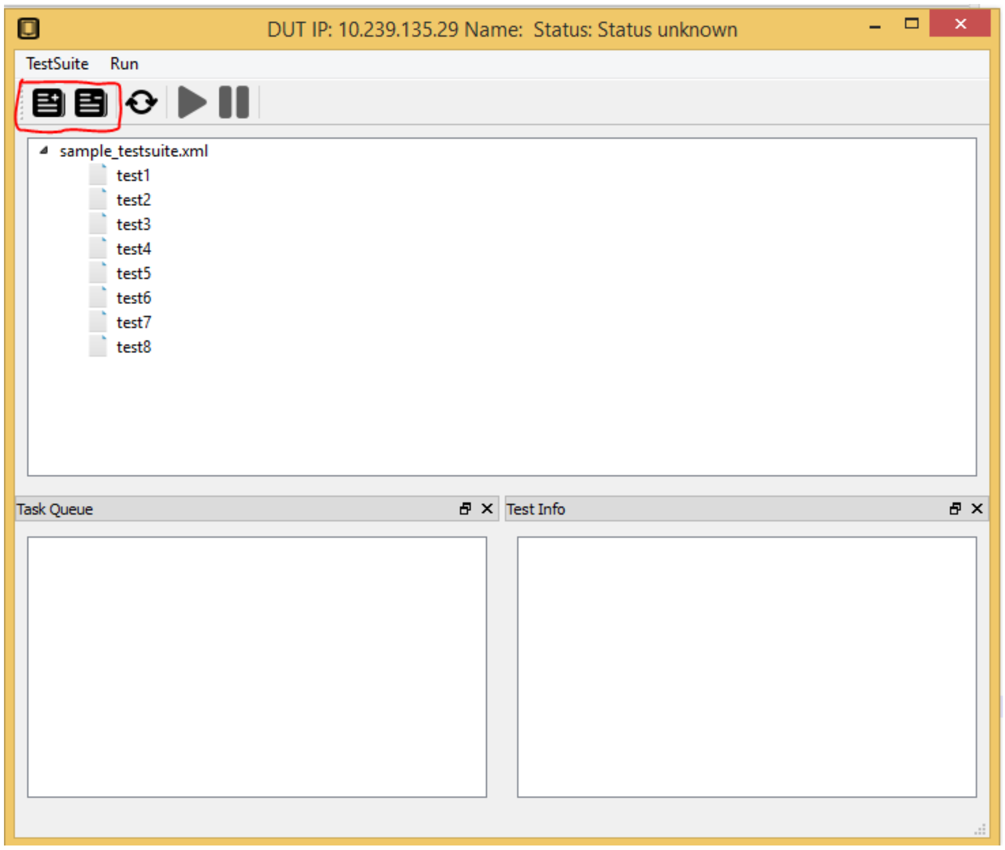
Note: When adding DUT, DUT ip is required, DUT name is optional



```
[ INFO ][ main_window.pyc:250 ] loaded tool:
testsuite_generator
[ INFO ][ main_window.pyc:250 ] loaded tool:
testsuite_generator
[ INFO ][ main_window.pyc:250 ] loaded tool:
report_generator
[ INFO ][ main_window.pyc:250 ] loaded tool:
.pyc:250 ] loaded tool:
ator.pyc:25 ] Launch report
enerator.pyc:262 ] Current
ould not generate report
erator.pyc:30 ] Launch
l
erger.pyc:26 ] Launch
[ INFO ][ workspace_spliter.pyc:26 ] Launch
workspace spliter
```

5.3.4 add test suites for each DUT

Note: Double click/right click DUT raw, can open DUT view. On DUT view you can add/delete test suite.

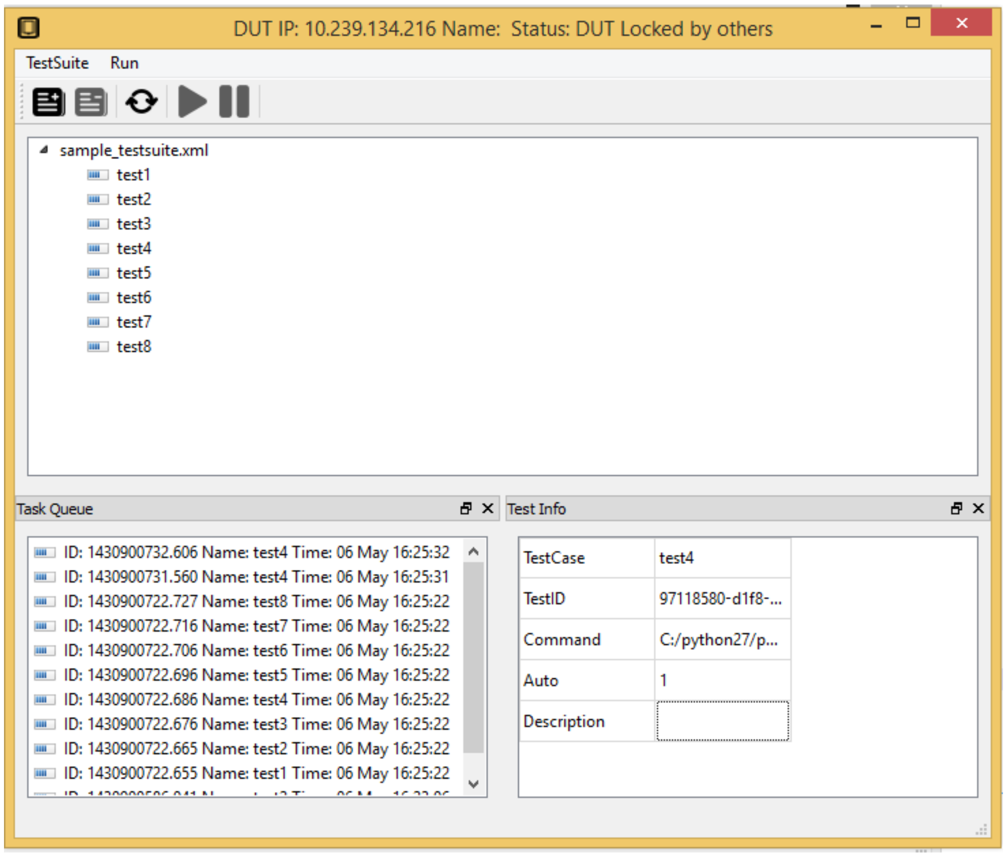


5.3.5 add test case to DUT task queue

Each DUT has a task queue, tasks in task queue will be run on DUT one by one when task queue started.

Note: Right click at the test suite or test case, you can add test suite or test case to task queue

You can add same test suite/test case to task queue multiple times, then these test suite/test case will be run multiple times



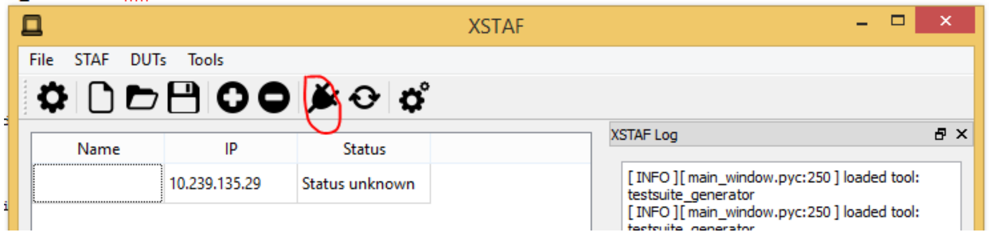
```
ID: 1430900732.606 Name: test4 Time: 06 May 16:25:32
ID: 1430900731.560 Name: test4 Time: 06 May 16:25:31
ID: 1430900722.727 Name: test8 Time: 06 May 16:25:22
ID: 1430900722.716 Name: test7 Time: 06 May 16:25:22
ID: 1430900722.706 Name: test6 Time: 06 May 16:25:22
ID: 1430900722.696 Name: test5 Time: 06 May 16:25:22
ID: 1430900722.686 Name: test4 Time: 06 May 16:25:22
ID: 1430900722.676 Name: test3 Time: 06 May 16:25:22
ID: 1430900722.665 Name: test2 Time: 06 May 16:25:22
ID: 1430900722.655 Name: test1 Time: 06 May 16:25:22
```

TestCase	test4
TestID	97118580-d1f8-...
Command	C:/python27/p...
Auto	1
Description	

5.3.6 start STAF

You can manage DUTs and test suites in workspace without enabling STAF, but if you want to execute test case on DUT, then you need to start STAF

You can start STAF by using STAF launcher, like `startSTAFProc.bat` on windows. Or you can use XSTAF to start STAF



Name	IP	Status
	10.239.135.29	Status unknown

```
[ INFO ][ main_window.pyc:250 ] loaded tool:
testsuite_generator
[ INFO ][ main_window.pyc:250 ] loaded tool:
```

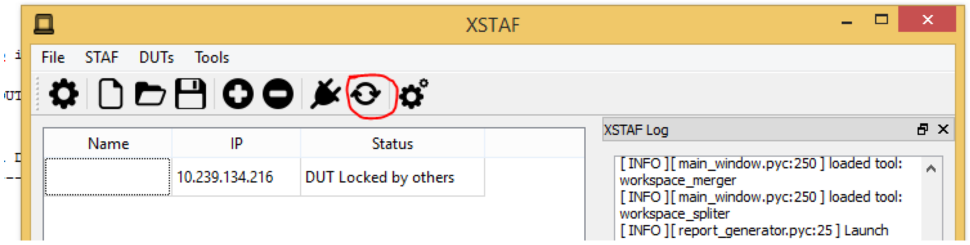
5.3.7 Check DUT status

After start STAF, you can check DUTs status

Note: If you start STAF not via XSTAF, you must check status in server view before run task

Note: Check status take longer time if you have a lot of DUTs in your workspace, to check specified DUT status, you can check status in DUT view

Check status take longer time if the DUT is not detected.

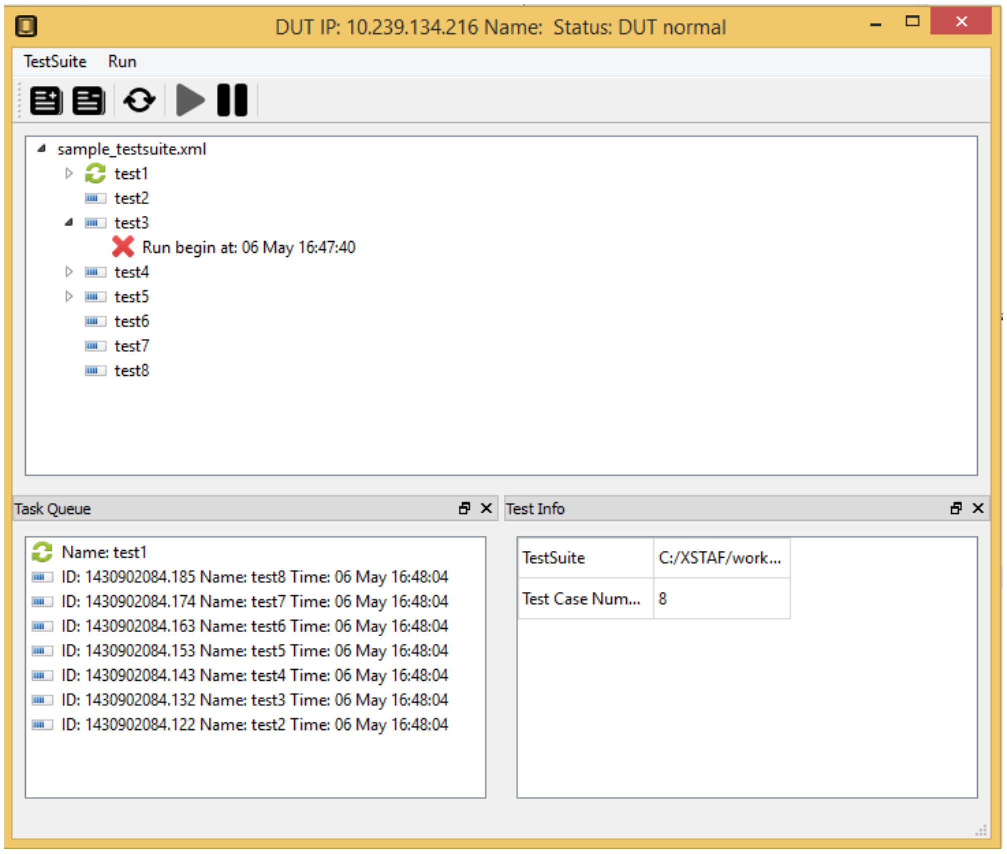


Name	IP	Status
	10.239.134.216	DUT Locked by others

```
[ INFO ] [ main_window.pyc:250 ] loaded tool:
workspace_merger
[ INFO ] [ main_window.pyc:250 ] loaded tool:
workspace_spliter
[ INFO ] [ report_generator.pyc:25 ] Launch
```

5.3.8 start task queue

If you DUT status is normal, then you can start task queue of this DUT, test cases in task queue will run synchronously.



```
sample_testsuite.xml
  test1
  test2
  test3
    Run begin at: 06 May 16:47:40
  test4
  test5
  test6
  test7
  test8
```

```
Name: test1
ID: 1430902084.185 Name: test8 Time: 06 May 16:48:04
ID: 1430902084.174 Name: test7 Time: 06 May 16:48:04
ID: 1430902084.163 Name: test6 Time: 06 May 16:48:04
ID: 1430902084.153 Name: test5 Time: 06 May 16:48:04
ID: 1430902084.143 Name: test4 Time: 06 May 16:48:04
ID: 1430902084.132 Name: test3 Time: 06 May 16:48:04
ID: 1430902084.122 Name: test2 Time: 06 May 16:48:04
```

TestSuite	C:/XSTAF/work...
Test Case Num...	8

5.3.9 generate test report

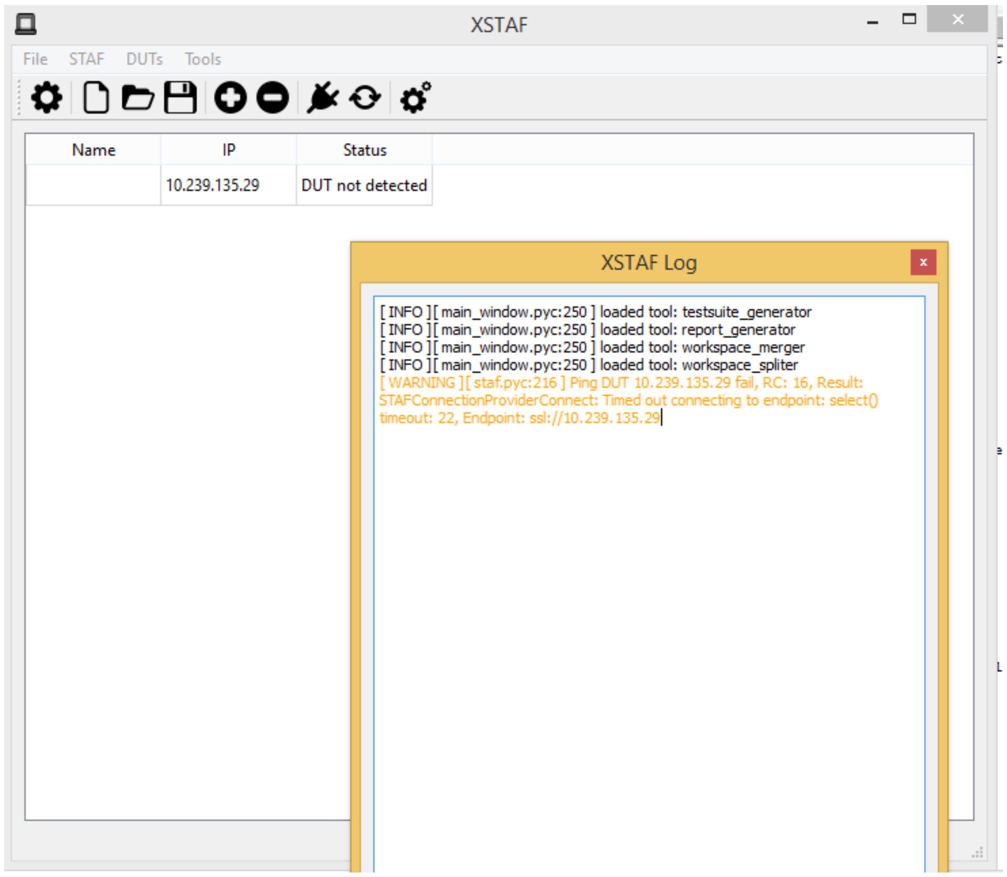
XSTAF has a built-in test report tool, to generate html report for one workspace. You can just use this test report tool, or write a your own customized report tool.

5.4 XSTAF Trouble shooting

XSTAF could have bugs since it's a new tool, not have been used and test thoroughly.

5.4.1 XSTAF logger

XSTAF provide a logger, with it you can see what is happening inside the XSTAF when bug happens. If you are a python developer, you maybe can fix it yourself and push the fix to github. Or you can just report a bug to me.



Name	IP	Status
	10.239.135.29	DUT not detected

```
[ INFO ][ main_window.pyc:250 ] loaded tool: testsuite_generator
[ INFO ][ main_window.pyc:250 ] loaded tool: report_generator
[ INFO ][ main_window.pyc:250 ] loaded tool: workspace_merger
[ INFO ][ main_window.pyc:250 ] loaded tool: workspace_spliter
[ WARNING ][ staf.pyc:216 ] Ping DUT 10.239.135.29 fail, RC: 16, Result:
STAFConnectionProviderConnect: Timed out connecting to endpoint: select()
timeout: 22, Endpoint: ssl://10.239.135.29
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search