**Wifi Interoperability HLD**

**2012-02-08**

# Components

Here are components and block diagram we designed,

(1)Description, which describes the test cases in json formatted configuration files.

(2)Parser, which analyzes the description file, converts it into specific test scenario and then call scenario python script.

(3)Controller, which traversal all cases in directory and schedule Parser to run every case.

(4)Reporter, which report the test result with specific format.



## Description File

### Scenario description file



We intend to adopt json format file describe scenario.

(1)APScenario is base description file for AP, TPLink and Cisco extends it, so Parser script can handle all aps in the same way, similar OO polymorphism.

(2)DUTScanario is base description file for target, Tate and Jem extends it, so Parser script can handle all targets in the same way, similar OO polymorphism.

Here are scenario description tags and explainations,

(1)name, scenario name. It is a unique name and always is file name.

(2)type, defined parser script how to execute scenario,

(a)once, which represents the scenario will be executed only once.

(b)loop, which represents the scenario will be executed repeatedly.

(c)none, which represents the scenario will not be executed.

(3)script, which represents python script and parameter list.

(4)reporter, which represents return string of the script.

(5)description, which describes this scenario.

(6)scenarios, which represents other sub-scenarios.

### Case descriptor

One case consists of scenarios and is organized a json description file. Parser script analysis description file, get scenarios, then execute scenario script.

Here are case description tag and explaination,

(1)name, which is the case name and always is the file name.

(2)type, defined parser script how to execute scenario,

(a)once, which represents the scenario will be executed only once.

(b)loop, which represents the scenario will be executed repeatedly.

(c)none, which represents the scenario will not be executed.

(3)description, which describes this scenario.

(4)scenarios, which represents other sub-scenarios.

## Main Flow

(1)The Runner script traversal case folder.

(2)The Runner script schedule Parser script to parse case description file.

(3)Then scenario script will be executed.

(4)Scenario script is responsible for setting up aps state or target state.

(5)Then execute test action and print test result.

(6)loop to step 2 until all cases are executed.

# File structure

**2.1 Here is main folder, it stores main programme.**

wifi\_interoperability/

**2.2 Here are scenario folders, they stores main scenario description files and python script.**

wifi\_interoperability /scenarios

wifi\_interoperability /scenarios/ap/cisco

wifi\_interoperability /scenarios/ap/dlink

wifi\_interoperability /scenarios/device/tate

wifi\_interoperability /scenarios/device/jem

**2.3 Here are case folders, they stores all cases, for example, cisco\_tate stores cases which consists of tate target and cisco ap.**

wifi\_interoperability /cases

wifi\_interoperability /cases/cisco\_tate

wifi\_interoperability /cases/cisco\_jem

wifi\_interoperability /cases/dlink\_tate

wifi\_interoperability /cases/dlink\_jem

**2.4 Here are utility folders, they stores common helper class or functions.**

wifi\_interoperability /utils

wifi\_interoperability /utils/ap

wifi\_interoperability /utils/target

**2.5 Here is test report folder.**

wifi\_interoperability /report

wifi\_interoperability /report/cisco\_tate

wifi\_interoperability /report/cisco\_jem

wifi\_interoperability /report/dlink\_tate

wifi\_interoperability /report/dlink\_jem

# Example

Here is an example how to integrate it to interoperability framework,

**Instantiate an AP object, Ap = AP(“WRT160N”)**

**Ap.powerOn()**

**Ap.resetToFactoryDefault()**

**Ap.setSSID(“MySSID”)**

**Ap.setSecurityMode(WPA2)**

**Ap.setChannel(11)**

**Ap.setWirelessMode(N-only)**

**Instantiate a DUT object Dut**

**Dut.scan()**

**Dut.setSecurityMode3(WPA2)**

**Dut.connect(“MySSID”)**

**Dut.getGatewayIP(gatewayIp)**

**Dut.ping(gatewayIp)**

**Report status**

**Dut.reboot()**

**Ap.powerOff()**

In this example, we do every thing here with ‘\*.py’ scripts,which make http request or reponse for the settings.

The configure files shows the test with many combinations of the “\*.py” scripts in scripts directory. The Configure files will be as follow:

We test it using tate target and cisco app

**(1) Make case description file, and store it in cases/cisco\_tate folder, named ping\_test.json, here is its content,**

{

“name”:”ping\_test”,

“type”:{“name”:”once”},

“description”:”tate ping cisco ap”,

“scenarios”:

[

{“ap\_init”:{“name”:”ap/cisco/ap\_init.json”,”type”:{“name”:“once”},

”parameter”:[”WRT160N”]}},

{“ap\_setup”:{“name”:” ap/cisco/ap\_setup.json”, “type”:{“name”:”once”}},

“parameter”:[”MySSID”,”WPA2”,”11”,”N-only”]},

{“device\_init”:{“name”:”device/tate/device\_init.json”,“type”:{“name”:”once”}, “parameter”:[””]}},

{“device\_setup”:{“name”:” device/tate/device \_setup.json”, “type”:{“name”:”once”},

“parameter”:[”WPA2”, “MYSSID”, “gatewayIp”]}},

{“test\_action”:{“name”:”device/tate/device\_ping.json”,“type”:{“name”:”once”},

“parameter”:[”gatewayIp”]}},

{“device\_teardown”:{“name”:” device/tate/device\_reboot.json”,

“type”:{“name”:”once”}, “parameter”:[””]}},

{“ap\_teardown”:{“name”:”ap/cisco/ap\_off.json”,“type”:{“name”:”once”},“parameter”:[””]}}]

}

**(2)Make scenario descriptor file and store it in ap/cisco folder. These scenarios will be re-used in other cases. In addition, every scenario has a python script which does actual work, here are their contents,**

**Ap init scenario defined by ap\_init.json**

{

“name”:” ap\_init”,

“type”:{“name”:” once”},

“script”:{“name”:” ap\_init.py”},

“reporter”:[“success”,”fail”],

“description”:”ap initialize”

}

**Ap setup scenario defined by ap\_setup.json**

{

“name”:” ap\_setup”,

“type”:{“name”:” once”},

“script”:{“name”:” ap\_setup.py”},

“reporter”:[“success”,”fail”],

“description”:”setting cisco ap security mode, channel, wireless mode.”

}

**Device init scenario defined by device\_init.json**

{

“name”:” dut\_init”,

“type”:{“name”:” once”},

“script”:{“name”:”device \_init.py”},

“reporter”:[“success”,”fail”],

“description”:”initialize device.”

}

**Device setup scenario defined by device\_setup.json**

{

“name”:” device\_setup”,

“type”:{“name”:”once”},

“script”:{“name”:” device\_setup.py”},

“reporter”:[“success”,”fail”],

“description”:”setting device wireless”

}

**Test action defined by test\_action.json**

{

“name”:” device\_ping”,

“type”:{“name”:”once”},

“script”:{“name”:” device\_ping.py”},

“reporter”:[“success”,”fail”],

“description”:”ping test.”

}

**Device teardown scenario defined by device\_teardown.json**

{

“name”:” device\_teardown”,

“type”:{“name”:” once”},

“script”:{“name”:” device\_teardown.py”},

“reporter”:[“success”,”fail”],

“description”:”device teardown.”

}

**Ap teardown scenario defined by device\_teardown.json**

{

“name”:” device\_teardown”,

“type”:{“name”:”once”},

“script”:{“name”:” ap\_teardown.py”},

“reporter”:[“success”,”fail”],

“description”:”cisco ap teardown”

}

# Pending problems

1. How to deal with parameters for each scenario?
2. How to deal with loop condition, scenarios parameter?
3. How to collect test results?
4. How to control the execute sequence of test case?
5. How to control power?