**Wifi inter-operability**

**HLD**

|  |  |  |  |
| --- | --- | --- | --- |
| DATE | Rev | Author | Summary of changes |
| **2/01/12** | **0.1** | **Lv Kai** | **Init** |
| **2/14/12** | **0.5** | **Lv Kai** | **Modified overview.** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc317106102)

[1.1 Purpose 4](#_Toc317106103)

[1.2 Overview 4](#_Toc317106104)

[2 Description Module 4](#_Toc317106105)

[2.1 Case description 4](#_Toc317106106)

[2.2 Scenario description 5](#_Toc317106107)

[3 Parser module 5](#_Toc317106108)

[4 Controller module 5](#_Toc317106109)

[5 AP module 5](#_Toc317106110)

[5.1 Configure AP 5](#_Toc317106111)

[5.2 Turn ON/OFF AP 6](#_Toc317106112)

[6 Main Flow 6](#_Toc317106113)

[7 Organize files 6](#_Toc317106114)

[8 Example 7](#_Toc317106115)

# Introduction

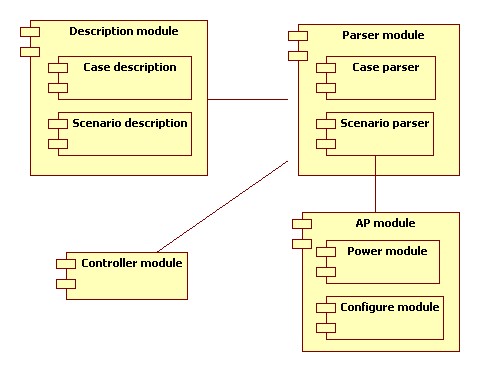
## Purpose

Verify Wifi inter-operability by switching different APs. This requires controlling different vendor’s AP such as Cisco, DLink, and controlling different target, such as Tate, Jem.

## Overview

This design document describes auto test framework, how to control different vendor’s AP, and how to control different target. Here are block diagram and divided different modules,

* Description module, It is a logical module, which describes the test case and scenario in JSON formatted files.
* Parser module, which analyzes the description files, converts them into specific test scripts and then calls these scripts.
* Controller module, which traversals all cases in organized directory and schedule Parser to run every case.
* AP module, which is responsible for turning ON/OFF AP and setting AP status.



# Description Module

## Case description

One case consists of many scenarios and is organized a JSON file. Parser module analysis it, get scenario description, and then execute python script which defined scenario description.

Here are case description tag and its explanation,

* Name, which is the case name and always is the same as file name.
* Type, defined parser script how to execute scenario,
  + Seq, which means sequent, represents the scenario will be executed sequentially.
  + Loop, represents the scenario will be executed repeatedly.
* Description, which describes this case usage.
* Scenarios, which constitutes case.

## Scenario description

Scenario is designed re-useable for cases, so cases are organized in different scenarios. There are two kinds of scenarios as following,

* APScenario is a logical description. It includes AP common description and is base description for AP, Cisco and DLink extend it, so Parser module can handle all APs in the same way, similar OO polymorphism.
* DUTScanario is the same as APScenario. It is a logical description also. Tate and Jem extend it.

Here are scenario description tags and explanations,

* Name, scenario name. It is a unique and always is the same as file name.
* Type, defined parser script how to execute scenario,
  + Seq, which represents that the scenario will be executed sequentially.
  + Loop, which represents that the scenario will be executed repeatedly.
* Script, which represents python script and parameter list. The python script does actually work, for example, setting AP.
* Description, which describes this scenario.
* Scenarios, which represent other sub-scenarios.

# Parser module

This module consists of python scripts. There are CaseParser and ScenarioParser. One parses case description and the other parses scenario description. ScenarioParser will call python defined in scenario description.

# Controller module

This module traversals all cases in organized directory and schedule Parser to run every case.

# AP module

This module control different vendor’s AP.

## Configure AP

Now, most APs offer URL for user setting in Web Browser, we can configure AP as following,

1. Create http connection using python httplib.
2. Get AP response and prepare for next request.
3. Request AP URL again.

## Turn ON/OFF AP

There will a power management module which controls how to turn on/off AP.

It is under designing.

# Main Flow

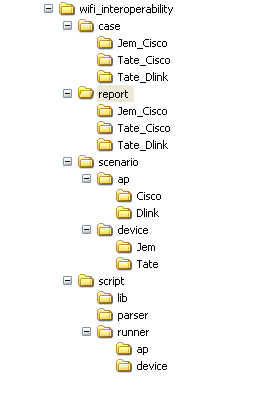
1. Controller module traversal all case folder.
2. Controller module schedule Parser module to parse case description file.
3. Then scenario script will be executed.
4. Scenario script is responsible for setting up AP state or target state.
5. Then execute test action and print test result.
6. Loop to step 2 until all cases are executed.

# Organize files

We organized description and scripts in different folder. Here are out directory tree,

* Case folder, it stores case description. There are sub folder named by joining AP and DUT, for example, Cisco\_Tate store Cisco AP and Tate target cases.
* Scenario folder, it stores scenario description. There are AP and device sub folder. AP folder has sub folder, named by different vendor’s product.
* Script folder, it stores python scripts. Parser sub folder, it stores Parser module scripts. Runner sub folder, it stores scenario scripts. Lib sub folder, it stores common scripts.
* Report folder, it stores test case report.

Here are snapshot



# Example

Here is an example how to integrate it to interoperability framework. Test case like following,

|  |
| --- |
| 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() |

If we want to integrate it to auto test framework with Tate target and Cisco router, do following,

1. Make a test\_ping.json description file in case/Cisco\_Tate folder, here is its content,

{

“name”:”test\_ping”,

“type”:{“name”:”seq”},

“description”:” Test Tate ping Cisco router”,

“scenarios”:

[

{“ap\_init”:{“name”:”ap/cisco/ap\_init.json”,”type”:{“name”:“seq”}, ”parameter”:[”WRT160N”]}},

{“ap\_setup”:{“name”:”ap/cisco/ap\_setup.json”,“type”:{“name”:”seq”}},“parameter”:[”MySSID”,”WPA2”,”11”,”N-only”]},

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

{“device\_setup”:{“name”:”device/tate/device\_setup.json”,“type”:{“name”:”seq”}, “parameter”:[”WPA2”, “MYSSID”, “gatewayIp”]}},

{“test\_action”:{“name”:”device/tate/device\_ping.json”,“type”:{“name”:”seq”},“parameter”:[”gatewayIp”]}},

{“device\_cleanup”:{“name”:”device/tate/device\_reboot.json”,“type”:{“name”:”seq”},“parameter”:[””]}},

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

]

}

1. Make several AP scenario descriptor file and in ap/cisco folder.

ap\_init.json

{

“name”:” ap\_init”,

“type”:{“name”:”seq”},

“script”:{“name”:” ap\_init.py”},

“description”:”Initialize Cisco AP”

}

ap\_setup.json

{

“name”:” ap\_setup”,

“type”:{“name”:”seq”},

“script”:{“name”:” ap\_setup.py”},

“description”:”Set Cisco AP security mode, channel, wireless mode.”

}

1. Make several DUT scenario descriptor file and in device/tate folder. Also, need to create python scripts in responding script folder.

device\_init.json

{

“name”:” dut\_init”,

“type”:{“name”:” seq”},

“script”:{“name”:”device \_init.py”},

“description”:”Initialize Tate device”

}

device\_setup.json

{

“name”:” device\_setup”,

“type”:{“name”:”seq”},

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

“description”:”Setting Tate device wireless”

}

1. Make test scenario descriptor file and in device/tate folder. Also need to create python script in responding folder.

test\_action.json

{

“name”:” device\_ping”,

“type”:{“name”:”seq”},

“script”:{“name”:” device\_ping.py”},

“description”:”Test ping”

}

1. Make clean\_up scenario descriptor file and python scripts in responding script folder.

device\_cleanup.json

{

“name”:” device\_cleanup”,

“type”:{“name”:”seq”},

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

“description”:”device cleanup”

}

ap\_cleanup.json

{

“name”:” device\_cleanup”,

“type”:{“name”:”seq”},

“script”:{“name”:” ap\_cleanup.py”},

“description”:”Cisco AP cleanup”

}