**Test frame work**

**2012-02-08**

目录

[1 Components 3](#_Toc316908180)

[1.1 Descriptor 4](#_Toc316908181)

[1.1.1 Scenario descriptor 4](#_Toc316908182)

[1.1.2 Case descriptor 6](#_Toc316908183)

[1.2 Runner & Parser & Reporter 7](#_Toc316908184)

[2 File structure 8](#_Toc316908185)

[2.1 scripts 9](#_Toc316908186)

[2.2 cases 10](#_Toc316908187)

[2.3 scenarios 11](#_Toc316908188)

[3 Configure rules 12](#_Toc316908189)

[3.1 Usage 12](#_Toc316908190)

[3.2 Rules 12](#_Toc316908191)

[3.2.1 Case configure files 12](#_Toc316908192)

[3.2.2 Scenario configure files 14](#_Toc316908193)

[4 Example 16](#_Toc316908194)

[5 Others 18](#_Toc316908195)

# Components

There are four components:

Descripter: describe the test cases in configure files.

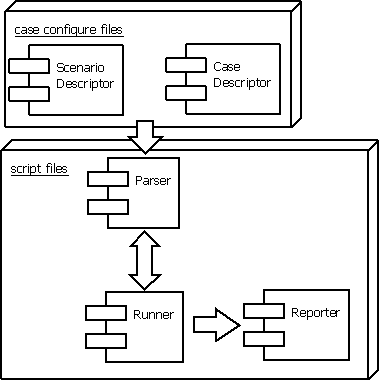
Parser: analyze the configure files and convert them into specific steps.

Runner: launch the parser and run the tools for test cases.

Reporter: report the test result with specific format.

Their overview relationship when excute one test case is shown in the following figure:

Figure 1‑1 components

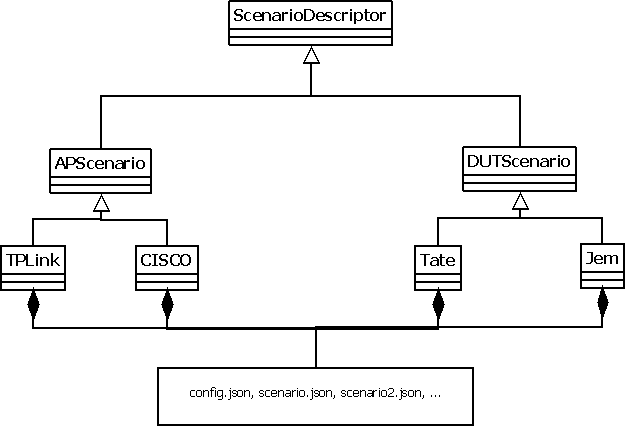


## Descriptor

There are two kinds of descriptors, so corresponding two kinds of configures files. Which are shown as following:

### Scenario descriptor

Figure 1‑2 structure of ScenarioDescriptor



Where, the config.json file describes the AP general setting (such as usename, ip), other \*.json describes set step, the \*.json file will specifics “\*.py” scripts which excute the actual setting.

By now, each scenario is described by a “\*.json” configure file in “scenarios” directory. Assume we have a scenario configure file, it contains the following field:

1. name

The name of the scenario which is represented by this configuration file. This is a unique name and it is the same as the file name often.

1. type

The default execution characteristics of this scenario. The type field contains the following value by now:

1. seq

Execute the scripts and imported scenarios in the sequence that they presented in this scenario record.

1. loop

Execute this scenario repeatedly.

1. none

Don’t execute this scenario.

1. script

Contains scripts in the “scripts” directory which is executed by this scenario (include the corresponding script arguments).

1. reporter

All of the possible result messagea of this scenarios, this is user defined string list, and then get the correct sring by the list index which return from scripts, we can also specify our own log out script here.

1. description

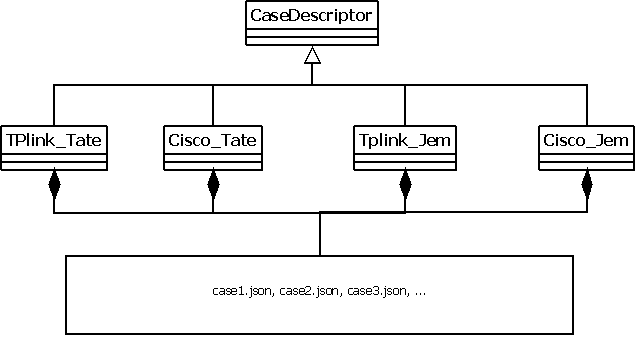
Description of this scenario.

1. scenarios

Contains other sub-scenarios in the “scenarios” directory which may be executed in this scenario.

### Case descriptor

Figure ‑ structure of CaseDescriptor



Where the \*.json specifies which scenarios in ScenarioDescriptor should be include in the corresponding test case.

By now, each test case is described by a “\*.json” configure file in “cases” directory. Assume we have a test case configure file, it contains the following field:

1. name

The name of the test case which is represented by this configuration file. This is a unique name represent the test case object, and it is the same as the file name often.

1. type

The default execution characteristics of this test case. The type field contains the following value by now:

1. seq

Execute the scenarios in the sequence that they presented in this test case record.

1. loop

Execute the whole test case in “seq” way repeatedly.

1. none

Don’t execute this test case.

1. description

Description of this test case.

1. “Scenarios”

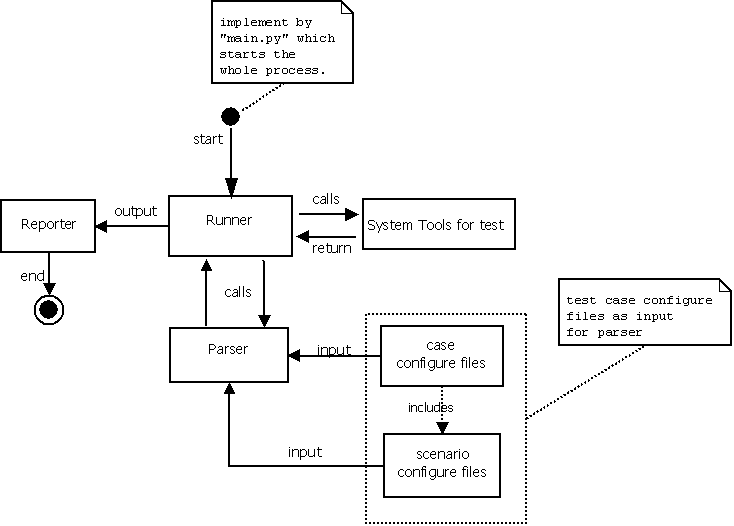
Contains the scenarios this test case needs for, in the “scenarios” directory, this field can be repeated and have different custom names in one case,(will be shown later) .

## Runner & Parser & Reporter

The Runner starts the test process, then it launches Parser to analyze the Descripter, then output the test result through Reporter.

The flow shown as the following:

Figure ‑ process

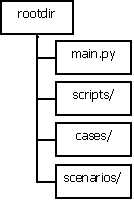


Here all the test cases are described by case and scenario configure files, and they are analyzed by Parser, excuted by Runner, then output to Reporter.

# File structure

The general directory structure by now is as follow:

Figure ‑ directory structure



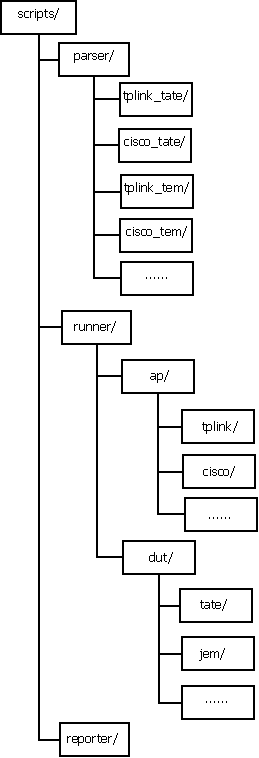
Where “main.py” is the start point of the whole test process and it is actually part of Runner in “Figure 1‑1 components”;

“scripts/”contains all the Parser scripts, Runner scripts and all the Reporter scripts.

“cases” and “scenarios” contains all the configure files, which describe the cases and the scenarios of each case, they work as Descripter.

## scripts

Figure ‑ scripts directory



As describe above,“parser” contains all the Parser scripts which used to analyze the corresponding configure files in the “cases” and “scenarios” directory;

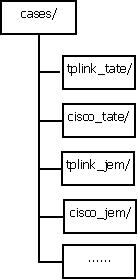
“runner” contains all the Runner scripts which used to do the real excute work;

“reporter” contains all the Reporter scripts which can be called at any time by Runner scripts to generate formatted output.

Note:in the above, tplink and cisco is ap’s type, tate and jem is dut’s type.

## cases

Figure ‑ cases directory

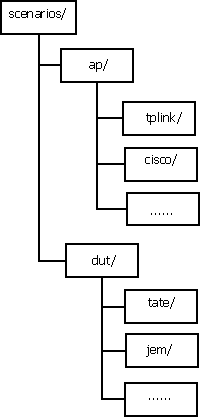


The “cases” directory contains all the test case describe files in different environment. Each environment is represented by a separate directory, which contains test case configure files under it.

The directory name represents the environment, for example the ap type is tplink and dut type is tate, so this environment’s configure files are in “tplink\_tate” directory.

## scenarios

Figure ‑ scenarios directory



The “scenarios” directory contains all the scenarios configure files under different ap and dut. Which will be referred to by test case configure files in “cases” directory. Note that there is a configure.json file in each subdirectory used for general setting for this environment.

# Configure rules

## Usage

Here will describe the configure rules in detail. At first, we should know a usecase to help understanding.

When we want to add a test case, we should do the following:

Step1. Create a test case configure file in specific subdirectory in “cases” directory, if the test case depends a new scenario, go step 2.

Step2. Create a new scenario configure file in specific subdirectory in “scenarios” directory, if the scenario needs to call a new script file, go step3.

Step3. Create a new script needed in specific subdirectory in “scripts” directory.

## Rules

We now talk about the detail rules of the configure files through a sample configure. At last, we will give an relatively complete example.

The files mentioned here contains:

case0.json, case1.json, case2.json: the test case configure files in “cases” directory.

scenario.json, scenario1json: the scenario configure files in “scenarios” directory.

main.py: the script to start the whole test.

script1.py: script files in “scripts” directory to implement the scenario in the test case .

reporter1.py:reporter script files define by user to output their own log.

We focus on the configure files in “cases” and “scenarios” directories.

### Case configure files

1. the file content is as the following

#case0.json

{“name”:case0,

“type”:{“name”:”none”},

“description”:”an example of excute nothing.”,

“scenarios1”:{“name”:”none”,”type”:{“name”:”none”},”parameter”:””},

“scenarios2”:{“name”:”none”, “type”:{“name”:”none”}, “parameter”:””}

}

#case1.json

　　{“name”:”case1”,

“type”:{“name”:”seq”},

“description”:”an example of excute scenarios by sequence.”,

“scenarios1”:{“name”:”scenario0.json”, “type”:{“name”:”seq”}, “parameter”:””},

“scenarios2”:{“name”:”scenario1.json”, “type”:{“name”:”seq”}, “parameter”:””},

}

#case2.json

{“name”:”case2”,

“type”:{”name”:”loop”, “condition”:”num<=5”},

“description”:” an example of execute five times("loop":"none" means forever).”,

“scenarios1”:{“name”:”scenario0.json”, “type”:{“name”:”seq”}, “parameter”:””},

“scenarios2”:{“name”:”scenario1.json”, “type”:{“name”:”seq”}, “parameter”:””},

}

1. detail
2. name

Stands for the test case name, cann’t be none, because “none” may be used as a keyword after.

For example:

　　 "name":”case0”

Here, means the case name is “case0”.

1. type

the way this test case is excuted.Its value is a form of “<name,argument>”, where:

name: the type name.

argument:the excute way parameter passed to the name. for “name”=”loop”, it means loop condition, other type now have no meaning.

For example:

“type”:{”name”:”loop”, “condition”:”num<=5”},

Means will excute the case 5 times.

1. Description

Extra information for this case.

1. scenarios

The scenarios which will be excuted by the case. It can be present many times with different key name in one case, and its value is a form of “<name,type,parameters>”, where:

name:means the scenario name, name=”none” means null scenario.

type:the same as “type” field of this test case before.

parameter:means the parameter passed to the scenario.

For example:

“scenarios1”:{“name”:”scenario0.json”, “type”:{“name”:”seq”}, “parameter”:””},

Here means will excute “scenario1.json” by the type of “seq”, and with out parameter.

The key “scenarios1” name is user-defined.

The things we should note here is: when excute a case, at first, determine the excute way for this case by type field; then, determine the excute way for the separate scenarios.

Note: “scenarios” is default scenario root directory.

### Scenario configure files

1. the file content is as the following

#scenario0.conf

{“name”:”scenario0”,

“type”:{“name”:”none”,”parameter”:””},

“script”:{“name”:”script1.py”},

“reporter”:””,

“description”:”an example of excute nothing”

“scenario0”:{“name”:”none”}

#scenario1.conf

{“name”:”scenario1”,

“type”:{“name”:”seq”,”parameter”:””},

“script”:{“name”:“script1.py”},

"reporter":{“name”:”reporter1.py”, “return”:[”ok”,”fail1”, ”fail2”]}

"description": "an example of execute script1.py and other scenario."},

“scenario0”:{“name”:”scenario0.json”,”type”:{“name”:”seq”}, “parameter”:””}

}

1. detail
2. name

Stands for the scenario name, cann’t be none, because “none” may be used as a keyword after.

For example:

　　 “name”:”scenario0”,

Here, means the scenario name is “scenario0”.

1. type

It has the same meaning as in case configure files.

The things we should note here is: when excute a scenario, at first, determine the excute way for this scenario by type field; then, determine the excute way for the separate sub-scenarios.

1. scripts

The scripts which will be excuted by the scenario. By now it support one script each scenario.

For example:

“script”:{“name”:“script1.py”},

Here, it means the path name of script file to excute is “script1.py”.

Note: “scripts” is default script root directory.

1. reporter

The messages reflect the result after excute.This is a string list, which defined by user according to the scripts they write. As we will get the message after excute the scenario by the index returned from corresponding script.

For example:

"reporter":{“name”:”reporter/reporter1.py”, “return”:[”ok”,”fail1”, ”fail2”]}

Means the possible message report after excuting this scenario is:”ok”, “fail1”, “fail2”, and the corresponding return value of script for message are:”0”,”1”,”2”, the script to process the log is “reporter1.py” as there has been specified by “name”.

Note:we can specified our own reporter script here.

Here, we put the reporter1.py into “scripts/reporter” directory.

1. Description

Extra information for this scenario.

1. scenarios

The same as in case configure file.

# Example

Assume we have the following example:

**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:

1. Case configure

#cisco\_tate/sample.json

{“name”:”sample”,

“type”:{“name”:”seq”},

“description”:”sample”,

“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”]},

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

“dut\_setup”:{“name”:” dut/tate/dut\_setup.json”, “type”:{“name”:”seq”},

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

“dut\_ping”:{“name”:”dut/tate/dut\_ping.json”,“type”:{“name”:”seq”},

“parameter”:[”gatewayIp”]},

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

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

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

}

1. Scenario configure

#ap/cisco/ap\_init.json

{“name”:” ap\_init”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/ap/cisco/ap\_init.py”},

“reporter”:[“success”,”fail”],

“description”:”ap initialize”

“scenario0”:{“name”:”none”}

}

#ap/cisco/ap\_setup.json

{“name”:” ap\_setup”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/ap/cisco/ap\_setup.py”},

“reporter”:[“success”,”fail”],

“description”:”set up ap’s sid, security mode, channel, wireless mode.”

“scenario0”:{“name”:”none”}

}

#dut/tate/dut\_init.json

{“name”:” dut\_init”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/dut/tate/dut\_scan.py”},

“reporter”:[“success”,”fail”],

“description”:”initialize dut.”

“scenario0”:{“name”:”none”}

}

#dut/tate/dut\_setup.json

{“name”:” dut\_setup”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/dut/tate/dut\_setup.py”},

“reporter”:[“success”,”fail”],

“description”:”setup dut mode, connect sid, and gateway IP.”

“scenario0”:{“name”:”none”}

}

#dut/tate/dut\_ping.json

{“name”:” dut\_ping”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/dut/tate/dut\_ping.py”},

“reporter”:[“success”,”fail”],

“description”:”ping test.”

“scenario0”:{“name”:”none”}

}

#dut/tate/dut\_teardown.json

{“name”:” dut\_teardown”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/dut/tate/dut\_reboot.py”},

“reporter”:[“success”,”fail”],

“description”:”dut teardown.”

“scenario0”:{“name”:”none”}

}

#ap/cisco/ap\_teardown.json

{“name”:” ap\_teardown”,

“type”:{“name”:” seq”},

“script”:{“name”:” runner/ap/cisco/ap\_powerOff.py”},

“reporter”:[“success”,”fail”],

“description”:”set up ap’s sid, security mode, channel, wireless mode.”

“scenario0”:{“name”:”none”}

}

# Others

1. How to deal with parameters for each scenario.
2. How to import the configure files.
3. How to deal with loop condition, scenarios parameter.
4. How to report result?
5. How to control the sequence of test case?(file name).