
Xtesting Documentation

Release master

#opnfv-functest (chat.freenode.net)

Sep 23, 2022

Contents

1	Technical guidelines	3
2	Try it!	5
3	Contents:	7
3.1	xtesting	7
4	Indices and tables	17
	Python Module Index	19
	Index	21

Xtesting have leveraged on Functest efforts to provide a reference testing framework:

- [Requirements Management](#)
- [Docker Slicing](#)
- [Functest Framework](#)

Xtesting aims at allowing a smooth integration of new Functest Kubernetes testcases.

But, more generally, it eases building any CI/CD toolchain for other domains than testing Virtualized Infrastructure Managers (VIM) such as [OpenStack](#).

It now offers a possible reuse of our framework in other OpenSource projects such as ONAP: [integration_demo_E2E_chain.pdf](#)

CHAPTER 1

Technical guidelines

- to support both python2 and python3
- to be fully covered by unit tests
- to be well rated by pylint (only local exceptions are accepted on purpose)
- to be released as a [python package](#) and then to be unlinked to OPNFV Milestones
- to provide [Docker containers](#) and manifests for both architectures supported by OPNFV: amd64 and arm64
- to publish the API documentation online

CHAPTER 2

Try it!

- run xtesting container:

```
$ sudo docker run opnfv/xtesting
```

- run xtesting via package (python2):

```
$ virtualenv xtesting-py2
$ . xtesting-py2/bin/activate
$ pip install xtesting
$ sudo xtesting-py2/bin/run_tests -t all
$ deactivate
```

- run xtesting via package (python3):

```
$ virtualenv xtesting-py3 -p python3
$ . xtesting-py3/bin/activate
$ pip install xtesting
$ sudo xtesting-py3/bin/run_tests -t all
$ deactivate
```


3.1 xtesting

3.1.1 xtesting package

Subpackages

xtesting.ci package

Submodules

xtesting.ci.run_tests module

The entry of running tests: 1) Parses xtesting/ci/testcases.yaml to check which testcase(s) to be run 2) Execute the common operations on every testcase (run, push results to db...) 3) Return the right status code

exception xtesting.ci.run_tests.**BlockingTestFailed**

Bases: Exception

Exception when the blocking test fails

class xtesting.ci.run_tests.**Result**

Bases: enum.Enum

The overall result in enumerated type

EX_ERROR = -1

EX_OK = 0

class xtesting.ci.run_tests.**RunTestsParser**

Bases: object

Parser to run tests

parse_args (*argv=None*)

Parse arguments.

It can call `sys.exit` if arguments are incorrect.

Returns: the arguments from cmdline

class `xtesting.ci.run_tests.Runner`

Bases: `object`

Runner class

static `get_dict_by_test` (*testname*)

static `get_run_dict` (*testname*)

Obtain the 'run' block of the testcase from `testcases.yaml`

main (***kwargs*)

Entry point of class `Runner`

run_all ()

Run all available testcases

run_test (*test*)

Run one test case

run_tier (*tier*)

Run one tier

static `source_envfile` (*rc_file='/var/lib/xtesting/conf/env_file'*)

Source the env file passed as arg

summary (*tier=None*)

To generate xtesting report showing the overall results

`xtesting.ci.run_tests.main` ()

Entry point

`xtesting.ci.tier_builder` module

TierBuilder class to parse testcases config file

class `xtesting.ci.tier_builder.TierBuilder` (*testcases_file*)

Bases: `object`

generate_tiers ()

get_test (*test_name*)

get_tests (*tier_name*)

get_tier (*tier_name*)

get_tier_name (*test_name*)

get_tier_names ()

get_tiers ()

read_test_yaml ()

xtesting.ci.tier_handler module

Tier and TestCase classes to wrap the testcases config file

```
class xtesting.ci.tier_handler.TestCase (name, enabled, skipped, criteria, blocking, description="", project="")
```

Bases: object

get_criteria ()

get_name ()

get_project ()

is_blocking ()

is_enabled ()

is_skipped ()

```
class xtesting.ci.tier_handler.Tier (name, description="")
```

Bases: object

add_test (*testcase*)

get_name ()

get_skipped_test ()

get_test (*test_name*)

get_test_names ()

get_tests ()

is_test (*test_name*)

skip_test (*testcase*)

xtesting.core package

Submodules

xtesting.core.behaveframework module

Define classes required to run any Behave test suites.

```
class xtesting.core.behaveframework.BehaveFramework (**kwargs)
```

Bases: *xtesting.core.testcase.TestCase*

BehaveFramework runner.

parse_results ()

Parse output.json and get the details in it.

run (***kwargs*)

Run the BehaveFramework feature files

Here are the steps:

- create the output directories if required,
- run behave features with parameters

- get the results in output.json,

Args: kwargs: Arbitrary keyword arguments.

Returns: EX_OK if all suites ran well. EX_RUN_ERROR otherwise.

xtesting.core.feature module

Define the parent classes of all Xtesting Features.

Feature is considered as TestCase offered by Third-party. It offers helpers to run any python method or any bash command.

```
class xtesting.core.feature.BashFeature (**kwargs)
```

Bases: *xtesting.core.feature.Feature*

Class designed to run any bash command.

```
execute (**kwargs)
```

Execute the cmd passed as arg

Args: kwargs: Arbitrary keyword arguments.

Returns: process return code if no exception, -1 otherwise.

```
class xtesting.core.feature.Feature (**kwargs)
```

Bases: *xtesting.core.testcase.TestCase*

Base model for single feature.

```
execute (**kwargs)
```

Execute the Python method.

The subclasses must override the default implementation which is false on purpose.

The new implementation must return 0 if success or anything else if failure.

Args: kwargs: Arbitrary keyword arguments.

```
run (**kwargs)
```

Run the feature.

It allows executing any Python method by calling execute().

It sets the following attributes required to push the results to DB:

- result,
- start_time,
- stop_time.

It doesn't fulfill details when pushing the results to the DB.

Args: kwargs: Arbitrary keyword arguments.

Returns: TestCase.EX_OK if execute() returns 0, TestCase.EX_RUN_ERROR otherwise.

xtesting.core.robotframework module

Define classes required to run any Robot suites.

```

class xtesting.core.robotframework.ResultVisitor
    Bases: robot.result.visitor.ResultVisitor

    Visitor to get result details.

    get_data ()
        Get the details of the result.

    visit_test (test)
        Implements traversing through tests.

        Can be overridden to allow modifying the passed in test without calling start_test() or
        end_test() nor visiting the body of the test.

class xtesting.core.robotframework.RobotFramework (**kwargs)
    Bases: xtesting.core.testcase.TestCase

    RobotFramework runner.

    generate_report ()
        Generate html and xunit outputs

    parse_results ()
        Parse output.xml and get the details in it.

    run (**kwargs)
        Run the RobotFramework suites

    Here are the steps:

        • create the output directories if required,
        • get the results in output.xml,
        • delete temporary files.

    Args: kwargs: Arbitrary keyword arguments.

    Returns: EX_OK if all suites ran well. EX_RUN_ERROR otherwise.

```

xtesting.core.testcase module

Define the parent class of all Xtesting TestCases.

```

class xtesting.core.testcase.TestCase (**kwargs)
    Bases: object

    Base model for single test case.

    EX_OK = 0
        everything is OK

    EX_PUBLISH_ARTIFACTS_ERROR = 66
        publish_artifacts() failed

    EX_PUSH_TO_DB_ERROR = 69
        push_to_db() failed

    EX_RUN_ERROR = 70
        run() failed

    EX_TESTCASE_FAILED = 68
        results are false

```

EX_TESTCASE_SKIPPED = 67

requirements are unmet

check_requirements ()

Check the requirements of the test case.

It can be overridden on purpose.

clean ()

Clean the resources.

It can be overridden if resources must be deleted after running the test case.

dir_results = '/var/lib/xtesting/results'

get_duration ()

Return the duration of the test case.

Returns: duration if start_time and stop_time are set “XX:XX” otherwise.

headers = {'Content-Type': 'application/json'}

is_successful ()

Interpret the result of the test case.

It allows getting the result of TestCase. It completes run() which only returns the execution status.

It can be overridden if checking result is not suitable.

Returns: TestCase.EX_OK if result is ‘PASS’. TestCase.EX_TESTCASE_SKIPPED if test case is skipped. TestCase.EX_TESTCASE_FAILED otherwise.

publish_artifacts ()

Push the artifacts to the S3 repository.

It allows publishing the artifacts.

It could be overridden if the common implementation is not suitable.

The credentials must be configured before publishing the artifacts:

- fill ~/.aws/credentials or ~/.boto,
- set AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY in env.

The next vars must be set in env:

- S3_ENDPOINT_URL (<http://127.0.0.1:9000>),
- S3_DST_URL (s3://xtesting/prefix),
- HTTP_DST_URL (<http://127.0.0.1/prefix>).

Returns: TestCase.EX_OK if artifacts were published to repository. TestCase.EX_PUBLISH_ARTIFACTS_ERROR otherwise.

push_to_db ()

Push the results of the test case to the DB.

It allows publishing the results and checking the status.

It could be overridden if the common implementation is not suitable.

The following attributes must be set before pushing the results to DB:

- project_name,

- `case_name`,
- `result`,
- `start_time`,
- `stop_time`.

The next vars must be set in env:

- `TEST_DB_URL`,
- `INSTALLER_TYPE`,
- `DEPLOY_SCENARIO`,
- `NODE_NAME`,
- `BUILD_TAG`.

Returns: `TestCase.EX_OK` if results were pushed to DB. `TestCase.EX_PUSH_TO_DB_ERROR` otherwise.

run (***kwargs*)

Run the test case.

It allows running `TestCase` and getting its execution status.

The subclasses must override the default implementation which is false on purpose.

The new implementation must set the following attributes to push the results to DB:

- `result`,
- `start_time`,
- `stop_time`.

Args: `kwargs`: Arbitrary keyword arguments.

xtesting.core.unit module

Define the parent class to run `unittest.TestSuite` as `TestCase`.

```
class xtesting.core.unit.Suite (**kwargs)
    Bases: xtesting.core.testcase.TestCase
```

Base model for running `unittest.TestSuite`.

```
generate_html (stream)
    Generate html report from subunit stream
```

Raises: `CalledProcessError`

```
classmethod generate_stats (stream)
    Generate stats from subunit stream
```

Raises: `Exception`

```
generate_xunit (stream)
    Generate junit report from subunit stream
```

Raises: `Exception`

run (***kwargs*)

Run the test suite.

It allows running any `unittest.TestSuite` and getting its execution status.

By default, it runs the suite defined as instance attribute. It can be overridden by passing name as arg. It must conform with `TestLoader.loadTestsFromName()`.

It sets the following attributes required to push the results to DB:

- `result`,
- `start_time`,
- `stop_time`,
- `details`.

Args: `kwargs`: Arbitrary keyword arguments.

Return: `TestCase.EX_OK` if any `TestSuite` has been run `TestCase.EX_RUN_ERROR` otherwise.

xtesting.core.vnf module

Define the parent class of all VNF TestCases.

exception `xtesting.core.vnf.OrchestratorDeploymentException`

Bases: `Exception`

Raise when orchestrator cannot be deployed.

exception `xtesting.core.vnf.VnfDeploymentException`

Bases: `Exception`

Raise when VNF cannot be deployed.

class `xtesting.core.vnf.VnfOnBoarding` (***kwargs*)

Bases: `xtesting.core.testcase.TestCase`

Base model for VNF test cases.

clean ()

Clean VNF test case.

It is up to the test providers to delete resources used for the tests. By default we clean:

- the user,
- the tenant

deploy_orchestrator ()

Deploy an orchestrator (optional).

If this method is overridden then raise `orchestratorDeploymentException` if error during orchestrator deployment

deploy_vnf ()

Deploy the VNF

This function **MUST** be implemented by vnf test cases. The details section **MAY** be updated in the vnf test cases.

The deployment can be executed via a specific orchestrator or using build-in orchestrators such as heat, OpenBaton, cloudify, juju, onap, ...

Returns: True if the VNF is properly deployed False if the VNF is not deployed

Raise VnfDeploymentException if error during VNF deployment

prepare ()

Prepare the environment for VNF testing:

Returns base.TestCase.EX_OK if preparation is successful

Raise VnfPreparationException in case of problem

run (kwargs)**

Run of the VNF test case:

- Deploy an orchestrator if needed (e.g. heat, cloudify, ONAP,...),
- Deploy the VNF,
- Perform tests on the VNF

A VNF test case is successful when the 3 steps are PASS If one of the step is FAIL, the test case is FAIL

Returns: TestCase.EX_OK if result is 'PASS'. TestCase.EX_TESTCASE_FAILED otherwise.

test_vnf ()

Test the VNF

This function MUST be implemented by vnf test cases. The details section MAY be updated in the vnf test cases.

Once a VNF is deployed, it is assumed that specific test suite can be run to validate the VNF. Please note that the same test suite can be used on several test case (e.g. clearwater test suite can be used whatever the orchestrator used for the deployment)

Returns: True if VNF tests are PASS False if test suite is FAIL

Raise VnfTestException if error during VNF test

exception xtesting.core.vnf.VnfPreparationException

Bases: Exception

Raise when VNF preparation cannot be executed.

exception xtesting.core.vnf.VnfTestException

Bases: Exception

Raise when VNF cannot be tested.

xtesting.utils package

Submodules

xtesting.utils.constants module

xtesting.utils.decorators module

xtesting.utils.decorators.**can_dump_request_to_file** (*method*)

xtesting.utils.env module

xtesting.utils.env.**get** (*env_var*)

xtesting.utils.env.**string**()

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

X

xtesting, 7
xtesting.ci, 7
xtesting.ci.run_tests, 7
xtesting.ci.tier_builder, 8
xtesting.ci.tier_handler, 9
xtesting.core, 9
xtesting.core.behaveframework, 9
xtesting.core.feature, 10
xtesting.core.robotframework, 10
xtesting.core.testcase, 11
xtesting.core.unit, 13
xtesting.core.vnf, 14
xtesting.utils, 15
xtesting.utils.constants, 15
xtesting.utils.decorators, 15
xtesting.utils.env, 16

A

`add_test()` (*xtesting.ci.tier_handler.Tier* method), 9

B

`BashFeature` (class in *xtesting.core.feature*), 10

`BehaveFramework` (class in *xtesting.core.behaveframework*), 9

`BlockingTestFailed`, 7

C

`can_dump_request_to_file()` (in module *xtesting.utils.decorators*), 15

`check_requirements()` (*xtesting.core.testcase.TestCase* method), 12

`clean()` (*xtesting.core.testcase.TestCase* method), 12

`clean()` (*xtesting.core.vnf.VnfOnBoarding* method), 14

D

`deploy_orchestrator()` (*xtesting.core.vnf.VnfOnBoarding* method), 14

`deploy_vnf()` (*xtesting.core.vnf.VnfOnBoarding* method), 14

`dir_results` (*xtesting.core.testcase.TestCase* attribute), 12

E

`EX_ERROR` (*xtesting.ci.run_tests.Result* attribute), 7

`EX_OK` (*xtesting.ci.run_tests.Result* attribute), 7

`EX_OK` (*xtesting.core.testcase.TestCase* attribute), 11

`EX_PUBLISH_ARTIFACTS_ERROR` (*xtesting.core.testcase.TestCase* attribute), 11

`EX_PUSH_TO_DB_ERROR` (*xtesting.core.testcase.TestCase* attribute), 11

`EX_RUN_ERROR` (*xtesting.core.testcase.TestCase* attribute), 11

`EX_TESTCASE_FAILED` (*xtesting.core.testcase.TestCase* attribute), 11

`EX_TESTCASE_SKIPPED` (*xtesting.core.testcase.TestCase* attribute), 11

`execute()` (*xtesting.core.feature.BashFeature* method), 10

`execute()` (*xtesting.core.feature.Feature* method), 10

F

`Feature` (class in *xtesting.core.feature*), 10

G

`generate_html()` (*xtesting.core.unit.Suite* method), 13

`generate_report()` (*xtesting.core.robotframework.RobotFramework* method), 11

`generate_stats()` (*xtesting.core.unit.Suite* class method), 13

`generate_tiers()` (*xtesting.ci.tier_builder.TierBuilder* method), 8

`generate_xunit()` (*xtesting.core.unit.Suite* method), 13

`get()` (in module *xtesting.utils.env*), 16

`get_criteria()` (*xtesting.ci.tier_handler.TestCase* method), 9

`get_data()` (*xtesting.core.robotframework.ResultVisitor* method), 11

`get_dict_by_test()` (*xtesting.ci.run_tests.Runner* static method), 8

`get_duration()` (*xtesting.core.testcase.TestCase* method), 12

`get_name()` (*xtesting.ci.tier_handler.TestCase* method), 9

`get_name()` (*xtesting.ci.tier_handler.Tier* method), 9

`get_project()` (*xtesting.ci.tier_handler.TestCase* method), 9

`get_run_dict()` (*xtesting.ci.run_tests.Runner* static method), 8

`get_skipped_test()` (*xtesting.ci.tier_handler.Tier* method), 9

`get_test()` (*xtesting.ci.tier_builder.TierBuilder* method), 8

get_test() (*xtesting.ci.tier_handler.Tier method*), 9
 get_test_names() (*xtesting.ci.tier_handler.Tier method*), 9
 get_tests() (*xtesting.ci.tier_builder.TierBuilder method*), 8
 get_tests() (*xtesting.ci.tier_handler.Tier method*), 9
 get_tier() (*xtesting.ci.tier_builder.TierBuilder method*), 8
 get_tier_name() (*xtesting.ci.tier_builder.TierBuilder method*), 8
 get_tier_names() (*xtesting.ci.tier_builder.TierBuilder method*), 8
 get_tiers() (*xtesting.ci.tier_builder.TierBuilder method*), 8

H

headers (*xtesting.core.testcase.TestCase attribute*), 12

I

is_blocking() (*xtesting.ci.tier_handler.TestCase method*), 9
 is_enabled() (*xtesting.ci.tier_handler.TestCase method*), 9
 is_skipped() (*xtesting.ci.tier_handler.TestCase method*), 9
 is_successful() (*xtesting.core.testcase.TestCase method*), 12
 is_test() (*xtesting.ci.tier_handler.Tier method*), 9

M

main() (*in module xtesting.ci.run_tests*), 8
 main() (*xtesting.ci.run_tests.Runner method*), 8

O

OrchestratorDeploymentException, 14

P

parse_args() (*xtesting.ci.run_tests.RunTestsParser method*), 7
 parse_results() (*xtesting.core.behaveframework.BehaveFramework method*), 9
 parse_results() (*xtesting.core.robotframework.RobotFramework method*), 11
 prepare() (*xtesting.core.vnf.VnfOnBoarding method*), 15
 publish_artifacts() (*xtesting.core.testcase.TestCase method*), 12
 push_to_db() (*xtesting.core.testcase.TestCase method*), 12

R

read_test_yaml() (*xtesting.ci.tier_builder.TierBuilder method*), 8
 Result (*class in xtesting.ci.run_tests*), 7
 ResultVisitor (*class in xtesting.core.robotframework*), 10
 RobotFramework (*class in xtesting.core.robotframework*), 11
 run() (*xtesting.core.behaveframework.BehaveFramework method*), 9
 run() (*xtesting.core.feature.Feature method*), 10
 run() (*xtesting.core.robotframework.RobotFramework method*), 11
 run() (*xtesting.core.testcase.TestCase method*), 13
 run() (*xtesting.core.unit.Suite method*), 13
 run() (*xtesting.core.vnf.VnfOnBoarding method*), 15
 run_all() (*xtesting.ci.run_tests.Runner method*), 8
 run_test() (*xtesting.ci.run_tests.Runner method*), 8
 run_tier() (*xtesting.ci.run_tests.Runner method*), 8
 Runner (*class in xtesting.ci.run_tests*), 8
 RunTestsParser (*class in xtesting.ci.run_tests*), 7

S

skip_test() (*xtesting.ci.tier_handler.Tier method*), 9
 source_envfile() (*xtesting.ci.run_tests.Runner static method*), 8
 string() (*in module xtesting.utils.env*), 16
 Suite (*class in xtesting.core.unit*), 13
 summary() (*xtesting.ci.run_tests.Runner method*), 8

T

test_vnf() (*xtesting.core.vnf.VnfOnBoarding method*), 15
 TestCase (*class in xtesting.ci.tier_handler*), 9
 TestCase (*class in xtesting.core.testcase*), 11
 Tier (*class in xtesting.ci.tier_handler*), 9
 TierBuilder (*class in xtesting.ci.tier_builder*), 8

V

visit_test() (*xtesting.core.robotframework.ResultVisitor method*), 11
 VnfDeploymentException, 14
 VnfOnBoarding (*class in xtesting.core.vnf*), 14
 VnfPreparationException, 15
 VnfTestException, 15

X

xtesting (*module*), 7
 xtesting.ci (*module*), 7
 xtesting.ci.run_tests (*module*), 7
 xtesting.ci.tier_builder (*module*), 8
 xtesting.ci.tier_handler (*module*), 9

xtesting.core (*module*), 9
xtesting.core.behaveframework (*module*), 9
xtesting.core.feature (*module*), 10
xtesting.core.robotframework (*module*), 10
xtesting.core.testcase (*module*), 11
xtesting.core.unit (*module*), 13
xtesting.core.vnf (*module*), 14
xtesting.utils (*module*), 15
xtesting.utils.constants (*module*), 15
xtesting.utils.decorators (*module*), 15
xtesting.utils.env (*module*), 16