

---

# Xtesting Documentation

*Release master*

**#opnfv-functest (chat.freenode.net)**

**Sep 15, 2018**



---

## Contents

---

<b>1</b>	<b>Technical guidelines</b>	<b>3</b>
<b>2</b>	<b>Try it!</b>	<b>5</b>
<b>3</b>	<b>Contents:</b>	<b>7</b>
3.1	xtesting . . . . .	7
<b>4</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



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](#)



---

## 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: `exceptions.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.Dependency (installer='.*', scenario='.*')
    Bases: object

    get_installer()

    get_scenario()

class xtesting.ci.tier_handler.TestCase (name, enabled, skipped, dependency, criteria,
                                         blocking, description="", project="")
    Bases: object

    get_criteria()

    get_name()

    get_project()

    is_blocking()

    is_compatible (ci_installer, ci_scenario)

    is_enabled()

    is_skipped()

class xtesting.ci.tier_handler.Tier (name, order, ci_loop, description="")
    Bases: object

    add_test (testcase)

    get_ci_loop()

    get_name()

    get_order()

    get_skipped_test()

    get_test (test_name)

    get_test_names()

    get_tests()

    is_test (test_name)

    skip_test (testcase)

xtesting.ci.tier_handler.split_text (text, max_len)
```

## xtesting.core package

### Submodules

## 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: 0 if cmd returns 0, -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.

        Returns: -1.

    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 the test and its keywords.

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

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

    RobotFramework runner.

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

    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_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.

    get_duration ()
        Return the duration of the test case.
```

**Returns:** duration if start\_time and stop\_time are set “XX:XX” otherwise.

**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.

**push\_to\_db** (\*\*kwargs)

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.

**Returns:** TestCase.EX\_RUN\_ERROR.



## 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.

```
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.

**Returns:** 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: exceptions.Exception
```

Raise when orchestrator cannot be deployed.

```
exception xtesting.core.vnf.VnfDeploymentException
    Bases: exceptions.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 successfull

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 successfull 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: exceptions.Exception

Raise when VNF preparation cannot be executed.

**exception xtesting.core.vnf.VnfTestException**

Bases: exceptions.Exception

Raise when VNF cannot be tested.

## xtesting.energy package

### Submodules

#### xtesting.energy.energy module

This module manages calls to Energy recording API.

```
class xtesting.energy.energy.EnergyRecorder
    Bases: object

    Manage Energy recording session.

    CONNECTION_TIMEOUT = 4

    INITIAL_STEP = 'running'

    energy_recorder_api = None

    static get_current_scenario()
        Get current running scenario (if any, None else).

    static load_config()
        Load connectivity settings from yaml.

        Load connectivity settings to Energy recording API

    logger = <logging.Logger object>

    static set_step(step)
        Notify energy recording service of current step of the testcase.

    static start(scenario)
        Start a recording session for scenario.

        param scenario: Starting scenario :type scenario: string

    static stop()
        Stop current recording session.

    static submit_scenario(scenario, step)
        Submit a complet scenario definition to Energy recorder API.

        param scenario: Scenario name :type scenario: string param step: Step name :type step: string

xtesting.energy.energy.enable_recording(method)
    Record energy during method execution.

    Decorator to record energy during “method” exection.

    param method: Method to surround with start and stop :type method: function
```

---

**Note:** “method” should belong to a class having a “case\_name” attribute

---

```
xtesting.energy.energy.finish_session(current_scenario)
    Finish a recording session.
```

## 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.feature`, [9](#)
- `xtesting.core.robotframework`, [10](#)
- `xtesting.core.testcase`, [11](#)
- `xtesting.core.unit`, [13](#)
- `xtesting.core.vnf`, [13](#)
- `xtesting.energy`, [15](#)
- `xtesting.energy.energy`, [15](#)
- `xtesting.utils`, [16](#)
- `xtesting.utils.constants`, [16](#)
- `xtesting.utils.decorators`, [16](#)
- `xtesting.utils.env`, [16](#)





## A

add\_test() (xtesting.ci.tier\_handler.Tier method), 9

## B

BashFeature (class in xtesting.core.feature), 9

BlockingTestFailed, 7

## C

can\_dump\_request\_to\_file() (in module xtesting.utils.decorators), 16

check\_requirements() (xtesting.core.testcase.TestCase method), 11

clean() (xtesting.core.testcase.TestCase method), 11

clean() (xtesting.core.vnf.VnfOnBoarding method), 13

CONNECTION\_TIMEOUT (xtesting.energy.energy.EnergyRecorder attribute), 15

## D

Dependency (class in xtesting.ci.tier\_handler), 9

deploy\_orchestrator() (xtesting.core.vnf.VnfOnBoarding method), 13

deploy\_vnf() (xtesting.core.vnf.VnfOnBoarding method), 13

dir\_results (xtesting.core.robotframework.RobotFramework attribute), 11

## E

enable\_recording() (in module xtesting.energy.energy), 15

energy\_recorder\_api (xtesting.energy.energy.EnergyRecorder attribute), 15

EnergyRecorder (class in xtesting.energy.energy), 15

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\_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

finish\_session() (in module xtesting.energy.energy), 15

## G

generate\_report() (xtesting.core.robotframework.RobotFramework method), 11

generate\_tiers() (xtesting.ci.tier\_builder.TierBuilder method), 8

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

get\_ci\_loop() (xtesting.ci.tier\_handler.Tier method), 9

get\_criteria() (xtesting.ci.tier\_handler.TestCase method), 9

get\_current\_scenario() (xtesting.energy.energy.EnergyRecorder static method), 15

get\_data() (xtesting.core.robotframework.ResultVisitor method), 10

get\_dict\_by\_test() (xtesting.ci.run\_tests.Runner static method), 8

get\_duration() (xtesting.core.testcase.TestCase method), 11

get\_installer() (xtesting.ci.tier\_handler.Dependency method), 9

get\_name() (xtesting.ci.tier\_handler.TestCase method), 9

get\_name() (xtesting.ci.tier\_handler.Tier method), 9

get\_order() (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\\_scenario\(\)](#) (xtesting.ci.tier\_handler.Dependency method), 9  
[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

## I

[INITIAL\\_STEP](#) (xtesting.energy.energy.EnergyRecorder attribute), 15  
[is\\_blocking\(\)](#) (xtesting.ci.tier\_handler.TestCase method), 9  
[is\\_compatible\(\)](#) (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

## L

[load\\_config\(\)](#) (xtesting.energy.energy.EnergyRecorder static method), 15  
[logger](#) (xtesting.energy.energy.EnergyRecorder attribute), 15

## M

[main\(\)](#) (in module xtesting.ci.run\_tests), 8  
[main\(\)](#) (xtesting.ci.run\_tests.Runner method), 8

## O

[OrchestratorDeploymentException](#), 13

## P

[parse\\_args\(\)](#) (xtesting.ci.run\_tests.RunTestsParser method), 7  
[parse\\_results\(\)](#) (xtesting.core.robotframework.RobotFramework method), 11  
[prepare\(\)](#) (xtesting.core.vnf.VnfOnBoarding method), 14  
[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), 10  
[run\(\)](#) (xtesting.core.feature.Feature method), 10  
[run\(\)](#) (xtesting.core.robotframework.RobotFramework method), 11  
[run\(\)](#) (xtesting.core.testcase.TestCase method), 12  
[run\(\)](#) (xtesting.core.unit.Suite method), 13  
[run\(\)](#) (xtesting.core.vnf.VnfOnBoarding method), 14  
[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

[set\\_step\(\)](#) (xtesting.energy.energy.EnergyRecorder static method), 15  
[skip\\_test\(\)](#) (xtesting.ci.tier\_handler.Tier method), 9  
[source\\_envfile\(\)](#) (xtesting.ci.run\_tests.Runner static method), 8  
[split\\_text\(\)](#) (in module xtesting.ci.tier\_handler), 9  
[start\(\)](#) (xtesting.energy.energy.EnergyRecorder static method), 15  
[stop\(\)](#) (xtesting.energy.energy.EnergyRecorder static method), 15  
[string\(\)](#) (in module xtesting.utils.env), 16  
[submit\\_scenario\(\)](#) (xtesting.energy.energy.EnergyRecorder static method), 15  
[Suite](#) (class in xtesting.core.unit), 13  
[summary\(\)](#) (xtesting.ci.run\_tests.Runner method), 8

## T

[test\\_vnf\(\)](#) (xtesting.core.vnf.VnfOnBoarding method), 14  
[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), 10  
[VnfDeploymentException](#), 13  
[VnfOnBoarding](#) (class in xtesting.core.vnf), 13  
[VnfPreparationException](#), 14  
[VnfTestException](#), 14

## 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.feature (module), [9](#)
- xtesting.core.robotframework (module), [10](#)
- xtesting.core.testcase (module), [11](#)
- xtesting.core.unit (module), [13](#)
- xtesting.core.vnf (module), [13](#)
- xtesting.energy (module), [15](#)
- xtesting.energy.energy (module), [15](#)
- xtesting.utils (module), [16](#)
- xtesting.utils.constants (module), [16](#)
- xtesting.utils.decorators (module), [16](#)
- xtesting.utils.env (module), [16](#)