zaza Documentation

Canonical Ltd

Nov 25, 2021

CONTENTS

Contents:

CHAPTER

ONE

RUNNING CHARM TESTS

The end-to-end tests of a charm are divided into distinct phases. Each phase can be run in isolation and tests shared between charms.

1.1 Running a suite of deployments and tests

functest-run-suite will read the charms tests.yaml and execute the deployments and tests outlined there. However, each phase can be run independently.

1.2 Charm Test Phases

Charms should ship with bundles that deploy the charm with different application versions, topologies or config options. functest-run-suite will run through each phase listed below in order for each bundle that is to be tested.

1.2.1 0) Environment Variables

Optionally setting the **MODEL_SETTINGS** environment variable allows model settings to be applied to the models created by zaza to run tests in. The settings will be applied on top of those set **charm_lifecycle.prepare.MODEL_DEFAULTS** so it can be used to override any default setting.

MODEL_SETTINGS should be a list of key/value pairs delimited by semicolon e.g.:

```
export MODEL_SETTINGS="no-proxy=jujucharms.com"
export MODEL_CONSTRAINTS="virt-type=kvm"
```

In addition to overriding Zaza's configuration via environment variables, some configuration can be done via a .zaza.yaml file in your home directory, eg.:

```
---
model_settings:
    default-series: xenial
    image-stream: daily
    test-mode: true
    transmit-vendor-metrics: false
    enable-os-upgrade: false
    automatically-retry-hooks: false
    use-default-secgroup: true
model_constraints:
```

(continues on next page)

(continued from previous page)

```
    mem: '4g'
secrets:
    setting1: value1
runtime_config: {}
```

In addition to the config shown above, some options can only be added via a .zaza.yaml file, eg.:

```
---
region: my-region-name
```

The above configuration is required to run Zaza on a multi cloud / region Juju controller.

1.2.2 1) Prepare

Prepare the environment ready for a deployment. At a minimum create a model to run the deployment in.

To run manually:

```
$ functest-prepare --help
usage: functest-prepare [-h] -m MODEL_NAME [--log LOGLEVEL]

optional arguments:
  -h, --help            show this help message and exit
  -m MODEL_NAME, --model-name MODEL_NAME
                        Name of new model
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.2.3 2) Before Deploy

Perform pre-deployment tasks, for example: setup a default model configuration that is necessary to deploy a bundle.

To run manually:

```
$ functest-before-deploy --help
usage: functest-before-deploy [-h] [-c BEFOREFUNCS [BEFOREFUNCS ...]] [--log LOGLEVEL]

optional arguments:
  -h, --help
                        show this help message and exit
  -c BEFOREFUNCS, --beforefuncs BEFOREFUNCS
                        Space separated list of config functions
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.2.4 3) Deploy

Deploy the target bundle and wait for it to complete. **functest-run-suite** will look at the list of bundles in the tests.yaml in the charm to determine the bundle.

In addition to the specified bundle the overlay template directory will be searched for a corresponding template (<bundle_name>.j2). If one is found then the overlay will be rendered using environment variables a specific set of environment variables as context. Currently these are:

- FIP_RANGE
- GATEWAY
- NAME_SERVER
- NET_ID
- OS_*
- VIP_RANGE

The rendered overlay will be used on top of the specified bundle at deploy time.

To run manually:

```
$ functest-deploy --help
usage: functest-deploy [-h] -m MODEL -b BUNDLE [--no-wait] [--log LOGLEVEL]

optional arguments:
  -h, --help            show this help message and exit
  -m MODEL, --model MODEL
                        Model to deploy to
  -b BUNDLE, --bundle BUNDLE
                        Bundle name (excluding file ext)
  --no-wait             Do not wait for deployment to settle
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.2.5 4) Configure

Post-deployment configuration, for example create network, tenant, image, etc. Any necessary post-deploy actions go here. **functest-run-suite** will look for a list of functions that should be run in tests.yaml and execute each in turn.

To run manually:

```
$ functest-configure --help
usage: functest-configure [-h] [-c CONFIGFUNCS [CONFIGFUNCS ...]] [--log LOGLEVEL]

optional arguments:
  -h, --help
                        show this help message and exit
  -c CONFIGFUNCS, --configfuncs CONFIGFUNCS
                        Space separated list of config functions
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.2.6 5) Test

Run tests. These maybe tests in zaza or a wrapper around another testing framework like rally or tempest. **functest-run-suite** will look for a list of test classes that should be run in tests.yaml and execute each in turn.

To run manually:

```
$ functest-test --help
usage: functest-test [-h] [-t TESTS [TESTS ...]] [--log LOGLEVEL]

optional arguments:
  -h, --help            show this help message and exit
  -t TESTS, --tests TESTS
                        Space separated list of test classes
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.2.7 6) Collect

Collect artifacts useful for debugging any failures or useful for trend analysis like deprecation warning or deployment time.

1.2.8 7) Destroy

Destroy the model:

```
$ functest-destroy --help
usage: functest-destroy [-h] -m MODEL_NAME [--log LOGLEVEL]

optional arguments:
  -h, --help            show this help message and exit
  -m MODEL_NAME, --model-name MODEL_NAME
                        Name of model to remove
  --log LOGLEVEL        Loglevel [DEBUG|INFO|WARN|ERROR|CRITICAL]
```

1.3 Example

First, grab the charm in question from the charm store:

```
charm pull cs:~openstack-charmers-next/vault
cd vault
```

1.3.1 Run tests via tox

To run all the charms functional tests:

```
tox -e func
```

or just the smoke test:

```
tox -e func-smoke
```

1.3.2 Run tests directly with functest commands

Setup the virtualenv needs to be created:

```
tox -e func-noop
source .tox/func-noop/bin/activate
```

All the phases can be run with a single command for a specific bundle:

```
$ functest-run-suite -b xenial-mysql
```

OR each phase can be run by hand,

Prepare phase:

```
$ functest-prepare -m testmodel
```

Pick a specific bundle to test:

```
$ functest-deploy -m testmodel -b tests/bundles/xenial-mysql.yaml
```

Run the configure script to prepare the environment for running tests:

```
$ functest-configure -m testmodel
```

Run test:

```
$ functest-test -m testmodel
```

Destroy the environment:

```
$ functest-destroy -m testmodel
```

1.3.3 Run tests directly using model maps

Steps defined in the tests.yaml will often be related to a model alias. For example:

```
charm_name: ubuntu
tests:
 - bionic_model:
   - zaza.charm_tests.noop.tests.NoopTestBionic
 - xenial_model:
   - zaza.charm_tests.noop.tests.NoopTestXenial
```

(continues on next page)

(continued from previous page)

```
 - zaza.charm_tests.noop.tests.NoopTest
configure:
 - bionic_model:
   - zaza.charm_tests.noop.setup.basic_setup_bionic
 - xenial_model:
   - zaza.charm_tests.noop.setup.basic_setup_xenial
 - zaza.charm_tests.noop.setup.basic_setup
```

In the above setup two model aliases are in use: 'bionic_model' and 'xenial_model'. To map an existing model to a model alias (and thereby run the specific step for the alias) pass in the map to the model command. This is done by specifying 'alias:existing_model_name'. For example if there is an existing model called 'bio' then to associate that with 'bionic_model' alias run:

```
$ functest-configure -m bionic_model:bio
$ functest-test -m bionic_model:bio
```

Multiple model aliases can also be passed. To run the tests associated with both aliases:

```
$ functest-configure -m bionic_model:bio -m xenial_model:xen
$ functest-test -m bionic_model:bio -m xenial_model:xen
```

CHAPTER

TWO

ENABLING ZAZA TESTS IN A CHARM

2.1 Update requirements and tox

Add zaza in the charms test-requirements.txt:

```
git+https://github.com/openstack-charmers/zaza.git#egg=zaza
```

Add targets to tox.ini should include a target like:

```
[testenv:func]
basepython = python3
commands =
    functest-run-suite --keep-model

[testenv:func-smoke]
basepython = python3
commands =
    functest-run-suite --keep-model --smoke
```

2.2 Add Bundles

The bundles live in tests/bundles of the built charm, eg:

```
tests/bundles/xenial.yaml
tests/bundles/xenial-ha.yaml
tests/bundles/bionic.yaml
```

The bundle may include overlay templates which are, currently, populated from environment variables. For example the xenial-ha template needs a VIP but the VIP will depend on the setup of the juju provider so will be different between test environments. To accommodate this an overlay is added:

```
tests/bundles/overlays/xenial-ha.yaml.j2
```

The overlay is in jinja2 format and the variables correspond to environment variables:

```
applications:
  vault:
    options:
        vip: '{{ OS_VIP00 }}'
```

It is also possible to provide overlay templates tailored for specific juju provider types, this can be useful to do any provider specific morphing of a bundle. To use this feature use the following directory layout:

```
tests/bundles/overlays/xenial.yaml.j2
tests/bundles/overlays/lxd/xenial.yaml.j2
```

With the above directory layout the overlay template in the lxd sub-directory will be used when tests are executed with juju on a LXD provider and the overlay template in the top level directory will be used for any other provider types.

Bundle templates can be placed in the *tests/bundles* directory. It is usually preferable to use overlay templates rather than bundle templates. Overlays can be used to neatly capture settings that a user might want to change on a per-cloud basis. However, if a bundle template is required place it in tests/bundles with a *j2* extension.

2.3 Add tests.yaml

A tests/tests.yaml file that describes the bundles to be run and the tests:

```
charm_name: vault
tests:
  - zaza.charm_tests.vault.VaultTest
configure:
  - zaza.charm_tests.vault.setup.basic_setup
gate_bundles:
  - base-xenial
  - base-bionic
dev_bundles:
  - base-xenial-ha
smoke_bundles:
  - base-bionic
```

When deploying zaza will wait for the deployment to settle and for the charms to display a workload status which indicates that they are ready. Sometimes one or more of the applications being deployed may have a non-standard workload status target state or message. To inform the deployment step what to wait for an optional target_deploy_status stanza can be added:

```
target_deploy_status:
  vault:
    workload-status: blocked
    workload-status-message: Vault needs to be initialized
  ntp:
    workload-status-message: Go for it
```

2.4 Adding tests to zaza

The setup and tests for a charm should live in zaza, this enables the code to be shared between multiple charms. To add support for a new charm create a directory, named after the charm, inside **zaza/charm_tests**. Within the new directory define the tests in **tests.py** and any setup code in **setup.py** This code can then be referenced in the charms **tests.yaml**

e.g. to add support for a new congress charm create a new directory in zaza:

```
mkdir zaza/charm_tests/congress
```

Add setup code into setup.py:

```
$ cat zaza/charm_tests/congress/setup.py
def basic_setup():
    congress_client(run_special_setup)
```

Add test code into tests.py:

```
class CongressTest(unittest.TestCase):

    def test_policy_create(self):
        policy = congress.create_policy()
        self.assertTrue(policy)
```

These now need to be referenced in the congress charms tests.yaml. Additional setup is needed to run a useful congress tests, so congress' tests.yaml might look like:

```
charm_name: congress
configure:
  - zaza.charm_tests.nova.setup.flavor_setup
  - zaza.charm_tests.nova.setup.image_setup
  - zaza.charm_tests.neutron.setup.create_tenant_networks
  - zaza.charm_tests.neutron.setup.create_ext_networks
  - zaza.charm_tests.congress.setup.basic_setup
tests:
  - zaza.charm_tests.keystone.KeystoneBasicTest
  - zaza.charm_tests.congress.CongressTest
gate_bundles:
  - base-xenial
  - base-bionic
dev_bundles:
  - base-xenial-ha
```

2.5 Deploying bundles using –force

In order to allow early testing of new Ubuntu series the *juju deploy* command has a *–force* option. This allows deployment of charms that don't specify the series being used, or for a new series that juju doesn't support yet.

Force deploying can be achieved in two ways, either on the command line, or via an *tests_options.force_deploy* entry in the *tests.yaml* file.

For the command line a *–force* param is provided:

> functest-run-suite –keep-model –dev –force functest-deploy <other options> –force

In the *tests.yaml* the option is added as a list item:

> charm_name: keystone smoke_bundles: - focal-ussuri
>
> …
>
> **tests_options:**
>
> **force_deploy:**
>
> - focal-ussuri

In the above case, focal-ussuri will be deployed using the –force parameter. i.e. the *tests_options.force_deploy['focal-ussuri']* option applies to the *focal-ussuri* bundle whether it appears in any of the bundle sections.

2.6 Augmenting behaviour of configure steps

Individual configuration steps or the library helper they use may define configuration step options that you may use to tweak behaviour from tests.yaml.

An example is the Neutron *basic_overcloud_network* configure job which for compatibility with existing scenario tests makes use of *juju_wait* when configuring the deployed cloud.

This does not work well if the model you are testing has applications with non-standard workload status messaging.

To replace the use of *juju_wait* with Zaza's configurable wait code:

charm_name: neutron-openvswitch smoke_bundles: - focal-ussuri-dvr-snat-migrate-ovn

…

configure_options: configure_gateway_ext_port_use_juju_wait: false

CHAPTER

THREE

UTILITIES API DOCUMENTATION

3.1 Juju Controller

Module for interacting with a juju controller.

async zaza.controller.**async_add_model**(*model_name*, *config=None*, *region=None*)
Add a model to the current controller.

Parameters

- **model_name** (*str*) – Name to give the new model.
- **config** (*dict*) – Model configuration.
- **region** (*str*) – Region in which to create the model.

async zaza.controller.**async_cloud**(*name=None*)
Return information about cloud.

Parameters **name** (*Optional[str]*) – Cloud name. If not specified, the cloud where the controller lives on is returned.

Returns Information on all clouds in the controller.

Return type CloudResult

async zaza.controller.**async_destroy_model**(*model_name*)
Remove a model from the current controller.

Parameters **model_name** (*str*) – Name of model to remove

async zaza.controller.**async_get_cloud**()
Return the name of the current cloud.

Returns Name of cloud

Return type str

async zaza.controller.**async_list_models**()
Return a list of the available models.

Returns List of models

Return type list

zaza.controller.**get_cloud_type**(*name=None*)
Return type of cloud.

Parameters **name** (*Optional[str]*) – Cloud name. If not specified, the cloud where the controller lives on is returned.

Returns Type of cloud

Return type str

zaza.controller.**go_list_models**()
Execute juju models.

NOTE: Excuting the juju models command updates the local cache of models. Python-juju currently does not update the local cache on add model. https://github.com/juju/python-libjuju/issues/267

Returns None

Return type None

3.2 Juju Model

Module for interacting with a juju model.

This module contains a number of functions for interacting with a Juju model mostly via libjuju. Async function also provice a non-async alternative via 'sync_wrapper'.

exception zaza.model.**ActionFailed**(*action*, *output=None*)
Exception raised when action fails.

class zaza.model.**CloudData**(*cloud_name*, *cloud*, *credential_name*, *credential*)

property cloud
Alias for field number 1

property cloud_name
Alias for field number 0

property credential
Alias for field number 3

property credential_name
Alias for field number 2

exception zaza.model.**CommandRunFailed**(*cmd*, *result*)
Command failed to run.

exception zaza.model.**MachineError**(*units*)
Exception raised for units in error state.

exception zaza.model.**ModelTimeout**
Model timeout exception.

exception zaza.model.**RemoteFileError**
Error with a remote file.

exception zaza.model.**ServiceNotRunning**(*service*)
Exception raised when service not running.

class zaza.model.**StatusResult**(*time*, *result*)

property result
Alias for field number 1

property time
Alias for field number 0

exception zaza.model.**UnitError**(*units*)
Exception raised for units in error state.

exception zaza.model.**UnitNotFound**(*unit_name*)
Unit was not found in model.

async zaza.model.**async_add_relation**(*application_name*, *local_relation*, *remote_relation*, *model_name=None*)
Add relation between applications.

Parameters

- **application_name** (*str*) – Name of application on this side of relation
- **local_relation** (*str*) – Name of relation on this application
- **remote_relation** (*str '<application>[:<relation_name>]'*) – Name of relation on the other application.
- **model_name** (*str*) – Name of model to operate on.

async zaza.model.**async_add_unit**(*application_name*, *count=1*, *to=None*, *model_name=None*, *wait_appear=False*)
Add unit(s) to an application.

Parameters

- **application_name** (*str*) – Name of application to add unit(s) to
- **count** (*int*) – Number of units to add
- **to** (*str*) – Location to add unit i.e. lxd:0
- **model_name** (*str*) – Name of model to operate on.
- **wait_appear** (*bool*) – Whether to wait for the unit to appear in juju status

async zaza.model.**async_block_until**(**conditions*, *timeout=None*, *wait_period=0.5*, *loop=None*)
Return only after all async conditions are true.

Based on juju.utils.block_until which currently does not support async methods as conditions.

Parameters

- **conditions** (*functions*) – Functions to evaluate.
- **timeout** (*float*) – Timeout in seconds
- **wait_period** (*float*) – Time to wait between re-assessing conditions.
- **loop** (*An event loop*) – The event loop to use

async zaza.model.**async_block_until_all_units_idle**(*model_name=None*, *timeout=2700*)
Block until all units in the given model are idle.

An example accessing this function via its sync wrapper:

```
block_until_all_units_idle('modelname')
```

Parameters

- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for status to be achieved

async zaza.model.**async_block_until_charm_url**(*application*, *target_url*, *model_name=None*, *timeout=2700*)

Block until the charm url matches target_url.

An example accessing this function via its sync wrapper:

```
block_until_charm_url('cinder', 'cs:openstack-charmers-next/cinder')
```

Parameters

- **application_name** (*str*) – Name of application
- **target_url** (*str*) – Target charm url
- **model_name** (*str*) – Name of model to interact with.
- **timeout** (*float*) – Time to wait for status to be achieved

async zaza.model.**async_block_until_file_has_contents**(*application_name*, *remote_file*, *expected_contents*, *model_name=None*, *timeout=2700*)

Block until the expected_contents are present on all units.

Block until the given string (expected_contents) is present in the file (remote_file) on all units of the given application.

An example accessing this function via its sync wrapper:

```
block_until_file_has_contents(
    'keystone',
    '/etc/apache2/apache2.conf',
    'KeepAlive On',
    model_name='modelname')
```

Parameters

- **application_name** (*str*) – Name of application
- **remote_file** (*str*) – Remote path of file to transfer
- **expected_contents** (*str*) – String to look for in file
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for contents to appear in file

async zaza.model.**async_block_until_file_matches_re**(*application_name*, *remote_file*, *pattern*, *re_flags=RegexFlag.MULTILINE*, *model_name=None*, *timeout=2700*)

Block until a file matches a pattern.

Block until the provided regular expression pattern matches the given file on all units of the given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application
- **remote_file** (*str*) – Remote path of file to match
- **pattern** (*str or compiled regex*) – Regular expression

- **re_flags** (*re.RegexFlag (int flag constants from the re module)*) – Regular expression flags if the pattern is a string
- **timeout** (*float*) – Time to wait for contents to appear in file

async zaza.model.**async_block_until_file_missing**(*app*, *path*, *model_name=None*, *timeout=2700*)

Block until the file at path is not there.

Block until the file at the param 'path' is not present on the file system for all units on a given application.

An example accessing this function via its sync wrapper:

```
block_until_file_missing(
    'keystone',
    '/some/path/name')
```

Parameters

- **app** (*str*) – the application name
- **path** (*str*) – the file name to check for.
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for contents to appear in file

async zaza.model.**async_block_until_file_missing_on_machine**(*machine*, *path*, *model_name=None*, *timeout=2700*)

Block until the file at 'path' is not present for a machine.

An example accessing this function via its sync wrapper:

```
block_until_file_missing_on_machine(
    '0',
    '/some/path/name')
```

Parameters

- **machine** (*str*) – the machine
- **path** (*str*) – the file name to check for.
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for until file is missing on a machine.

async zaza.model.**async_block_until_file_ready**(*application_name*, *remote_file*, *check_function*, *model_name=None*, *timeout=2700*)

Block until the check_function passes against.

Block until the check_function passes against the provided file. It is unlikely that a test would call this function directly, rather it is provided as scaffolding for tests with a more specialised purpose.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application
- **remote_file** (*str*) – Remote path of file to transfer
- **check_function** (*function*) – Function to use to check if file is ready, must take exactly one argument which is the file contents.

- **timeout** (*float*) – Time to wait for contents to appear in file

async zaza.model.**async_block_until_machine_status_is**(*machine*, *status*, *model_name=None*, *invert_check=False*, *timeout=600*, *interval=4.0*, *refresh=True*)

Block until the agent status for a machine (doesn't) match status.

Block until the agent status of a machine does (or doesn't with the param *invert_status=True*) match the passed string (in param *status*). If the timeout is exceeded then the function raises an Exception.

Parameters

- **machine** (*str*) – the machine to watch, as provided from the get_status()
- **status** (*str*) – the string to match the machine's agent status to.
- **model_name** (*str*) – Name of model to query.
- **invert_check** (*bool*) – whether to invert the check (default False)
- **timeout** (*int*) – the time to wait for the status (or inverse of) (default 600 seconds).
- **interval** (*float*) – The minimum time between calls to get_status
- **refresh** (*bool*) – Force a refresh; do not used cached results

Raises asyncio.TimeoutError if the timeout is exceeded.

async zaza.model.**async_block_until_oslo_config_entries_match**(*application_name*, *remote_file*, *expected_contents*, *model_name=None*, *timeout=2700*)

Block until dict is represented in the file using oslo.config parser.

Block until the expected_contents are in the given file on all units of the application. For example to check for the following configuration:

```
[DEFAULT]
debug = False

[glance_store]
filesystem_store_datadir = /var/lib/glance/images/
default_store = file
```

Call the check via its sync wrapper:

```
expected_contents = {
    'DEFAULT': {
        'debug': ['False']},
    'glance_store': {
        'filesystem_store_datadir': ['/var/lib/glance/images/'],
         'default_store': ['file']}}

block_until_oslo_config_entries_match(
    'glance',
    '/etc/glance/glance-api.conf',
    expected_contents,
    model_name='modelname')
```

Parameters

- **application_name** (*str*) – Name of application

- **remote_file** (*str*) – Remote path of file to transfer
- **expected_contents** (*{}*) – The key value pairs in their corresponding sections to be looked for in the remote_file
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for contents to appear in file

async zaza.model.**async_block_until_service_status**(*unit_name*, *services*, *target_status*, *model_name=None*, *timeout=2700*, *pgrep_full=False*)

Block until all services on the unit are in the desired state.

Block until all services on the unit are in the desired state (stopped or running):

```
block_until_service_status(
    first_unit,
    ['glance-api'],
    'running',
    model_name='modelname')
```

Parameters

- **unit_name** (*str*) – Name of unit to run action on
- **services** (*[]*) – List of services to check
- **target_status** (*str*) – State services should be in (stopped or running)
- **model_name** (*str*) – Name of model to query.
- **pgrep_full** (*bool*) – Should pgrep be used rather than pidof to identify a service.
- **timeout** (*int*) – Time to wait for status to be achieved

async zaza.model.**async_block_until_services_restarted**(*application_name*, *mtime*, *services*, *model_name=None*, *timeout=2700*, *pgrep_full=False*)

Block until the given services have a start time later then mtime.

For example to check that the glance-api service has been restarted:

```
block_until_services_restarted(
    'glance',
    1528294585,
    ['glance-api'],
    model_name='modelname')
```

Parameters

- **application_name** (*str*) – Name of application
- **mtime** (*int*) – Time in seconds since Epoch to check against
- **services** (*[]*) – Listr of services to check restart time of
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for services to be restarted
- **pgrep_full** (*bool*) – Should pgrep be used rather than pidof to identify a service.

async zaza.model.**async_block_until_unit_count**(*application*, *target_count*, *model_name=None*, *timeout=2700*)

Block until the number of units matches target_count.

An example accessing this function via its sync wrapper:

```
block_until_unit_count('keystone', 4)
```

Parameters

- **application_name** (*str*) – Name of application
- **target_count** (*int*) – Number of expected units.
- **model_name** (*str*) – Name of model to interact with.
- **timeout** (*float*) – Time to wait for status to be achieved

async zaza.model.**async_block_until_unit_wl_message_match**(*unit*, *status_pattern*, *model_name=None*, *negate_match=False*, *timeout=2700*)

Block until the unit has a workload status message that matches pattern.

Parameters

- **unit** (*str*) – the unit to check against
- **status_pattern** (*str*) – Regex pattern to check status against.
- **model_name** (*Union[None, str]*) – Name of model to query.
- **negate_match** (*Union[None, bool]*) – Wait until the match is not true; i.e. none match
- **timeout** (*float*) – Time to wait for unit to achieved desired status

async zaza.model.**async_block_until_unit_wl_status**(*unit_name*, *status*, *model_name=None*, *negate_match=False*, *timeout=2700*, *subordinate_principal=None*)

Block until the given unit has the desired workload status.

A units workload status may change during a given action. This function blocks until the given unit has the desired workload status:

```
block_until_unit_wl_status(
   aunit,
   'active'
   model_name='modelname')
```

NOTE: unit.workload_status was actually reporting the application workload status. Using the full status output from model.get_status() gives us unit by unit workload status.

Parameters

- **unit_name** (*str*) – Name of unit
- **status** (*str*) – Status to wait for (active, maintenance etc)
- **model_name** (*str*) – Name of model to query.
- **negate_match** (*bool*) – Wait until the match is not true.
- **timeout** (*float*) – Time to wait for unit to achieved desired status

- **subordinate_principal** (*str*) – Name of the principal of unit_name, if unit_name is a subordinate

async zaza.model.**async_block_until_units_on_machine_are_idle**(*machine*, *model_name=None*, *timeout=2700*)

Block until all the units on a machine are idle.

Parameters

- **machine** (*str*) – the machine
- **model_name** (*str*) – Name of model to query.
- **timeout** (*float*) – Time to wait for units on machine to be idle.

async zaza.model.**async_block_until_wl_status_info_starts_with**(*app*, *status*, *model_name=None*, *negate_match=False*, *timeout=2700*)

Block until the all the units have a desired workload status.

Block until all of the units have a desired workload status that starts with the string in the status param.

Parameters

- **app** (*str*) – the application to check against
- **status** (*str*) – Status to wait for at the start of the string
- **model_name** (*Union[None, str]*) – Name of model to query.
- **negate_match** (*Union[None, bool]*) – Wait until the match is not true; i.e. none match
- **timeout** (*float*) – Time to wait for unit to achieved desired status

async zaza.model.**async_check_if_subordinates_idle**(*app*, *unit_name*)

Check if the specified unit's subordinates are idle.

Parameters

- **app** (*str*) – The name of the Juju application, ex: mysql
- **unit_name** (*str*) – The unit name of the application, ex: mysql/0

Returns The agent status, either active / idle, returned by Juju

Return type str

async zaza.model.**async_destroy_unit**(*application_name*, **unit_names*, *model_name=None*, *wait_disappear=False*)

Remove unit(s) of an application.

Parameters

- **application_name** (*str*) – Name of application to remove unit(s) from
- **model_name** (*str*) – Name of model to operate on.
- **wait_disappear** (*bool*) – Whether to wait for the unit to disappear from juju status

Parm unit_names One or more unit names. i.e. app/0

async zaza.model.**async_get_agent_status**(*app*, *unit_name*)

Get the current status of the specified unit.

Parameters

- **app** (*str*) – The name of the Juju application, ex: mysql

- **unit_name** (*str*) – The unit name of the application, ex: mysql/0

Returns The agent status, either active / idle, returned by Juju

Return type str

async zaza.model.**async_get_application**(*application_name*, *model_name=None*)
Return an application object.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application to retrieve units for

Returns Application object

Return type object

async zaza.model.**async_get_application_config**(*application_name*, *model_name=None*)
Return application configuration.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns Dictionary of configuration

Return type dict

async zaza.model.**async_get_cloud_data**(*credential_name=None*, *model_name=None*)
Get connection details and credentials for cloud supporting given model.

Parameters

- **credential_name** (*Optional[str]*) – Name of credential to retrieve.
- **model_name** (*Optional[str]*) – Model name to operate on.

Returns Credential Name, Juju Cloud Credential object

Return type *CloudData*[str, Cloud, str, CloudCredential]

Raises AssertionError

async zaza.model.**async_get_current_model**()
Return the current active model name.

Connect to the current active model and return its name.

Returns String curenet model name

Return type str

async zaza.model.**async_get_juju_model**()
Retrieve current model.

First check the environment for JUJU_MODEL. If this is not set, get the current active model.

Returns In focus model name

Return type str

async zaza.model.**async_get_latest_charm_url**(*charm_url*, *channel=None*, *model_name=None*)
Get charm url, including revision number, for latest charm version.

Parameters

- **charm_url** (*str*) – Charm url without revision number
- **channel** (*str*) – Channel to use when getting the charm from the charm store, e.g. 'development'
- **model_name** (*str*) – Name of model to operate on

async zaza.model.**async_get_lead_unit**(*application_name*, *model_name=None*)
Return the leader unit for a given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns Name of unit with leader status

Raises zaza.utilities.exceptions.JujuError

async zaza.model.**async_get_lead_unit_ip**(*application_name*, *model_name=None*)
Return the IP address of the lead unit of a given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns IP of the lead unit

Return type str

Raises zaza.utilities.exceptions.JujuError

async zaza.model.**async_get_lead_unit_name**(*application_name*, *model_name=None*)
Return name of unit with leader status for given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns Name of unit with leader status

Return type str

Raises zaza.utilities.exceptions.JujuError

async zaza.model.**async_get_machines**(*application_name*, *model_name=None*)
Return all the machines of a given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application to retrieve units for

Returns List of juju machines

Return type [juju.machine.Machine, juju.machine.Machine,...]

async zaza.model.**async_get_principle_sub_map**(*model_name=None*)
Get a map of principle units to a list subordinates.

Parameters **model_name** (*str*) – Name of model to operate on

Returns Name of unit

Return type Dict[str, [str]]

async zaza.model.**async_get_principle_unit**(*unit_name*, *model_name=None*)
Get principle unit name for subordinate.

Parameters

- **unit_name** (*str*) – Name of unit
- **model_name** (*str*) – Name of model to operate on

Returns Name of unit

Return type Union[str, None]

async zaza.model.**async_get_relation_id**(*application_name*, *remote_application_name*, *model_name=None*, *remote_interface_name=None*)
Get relation id of relation from model.

Parameters

- **application_name** (*str*) – Name of application on this side of relation
- **remote_application_name** (*str*) – Name of application on other side of relation
- **remote_interface_name** (*Optional(str)*) – Name of interface on remote end of relation
- **model_name** (*str*) – Name of model to operate on

Returns Relation id of relation if found or None

Return type any

async zaza.model.**async_get_status**(*model_name=None*, *interval=4.0*, *refresh=True*)
Return the full status, but share calls between different asyncs.

Return the full status for the model_name (current model is None), but no faster than interval time, which is a default of 4 seconds. If refresh is True, then this function waits until the interval is exceeded, and then returns the refreshed status. This is the default. If refresh is False, then the function immediately returns with the cached information.

This is to enable multiple co-routines to access the status information without making multiple calls to Juju which all essentially will return identical information.

Note that this is NOT thread-safe, but is async safe. i.e. multiple different co-operating async futures can call this (in the same thread) and all access the same status.

Parameters

- **model_name** (*str*) – Name of model to query.
- **interval** (*float*) – The minimum time between calls to get_status
- **refresh** (*bool*) – Force a refresh; do not used cached results

Returns dictionary of juju status

Return type dict

async zaza.model.**async_get_unit_service_start_time**(*unit_name*, *service*, *model_name=None*, *timeout=None*, *pgrep_full=False*)
Return the time that the given service was started on a unit.

Return the time (in seconds since Epoch) that the oldest process of the given service was started on the given unit. If the service is not running raise ServiceNotRunning exception.

If pgrep_full is True ensure that any special characters in the name of the service are escaped e.g.

service = 'aodh-evaluator: AlarmEvaluationService worker(0)'

Parameters

- **model_name** (*str*) – Name of model to query.
- **unit_name** (*str*) – Name of unit to run action on
- **service** (*str*) – Name of service to check is running
- **timeout** (*int*) – Time to wait for status to be achieved
- **pgrep_full** (*bool*) – Should pgrep be used rather than pidof to identify a service.

Returns time in seconds since Epoch on unit

Return type int

Raises ServiceNotRunning

async zaza.model.**async_get_unit_time**(*unit_name*, *model_name=None*, *timeout=None*)

Get the current time (in seconds since Epoch) on the given unit.

Parameters

- **model_name** (*str*) – Name of model to query.
- **unit_name** (*str*) – Name of unit to run action on

Returns time in seconds since Epoch on unit

Return type int

async zaza.model.**async_get_units**(*application_name*, *model_name=None*)

Return all the units of a given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application to retrieve units for

Returns List of juju units

Return type [juju.unit.Unit, juju.unit.Unit,...]

async zaza.model.**async_remove_application**(*application_name*, *model_name=None*, *forcefully_remove_machines=False*)

Remove application from model.

Parameters

- **application_name** (*str*) – Name of application
- **model_name** (*str*) – Name of model to query.
- **forcefully_remove_machines** (*bool*) – Forcefully remove the machines the application is runing on.

async zaza.model.**async_remove_relation**(*application_name*, *local_relation*, *remote_relation*, *model_name=None*)

Remove relation between applications.

Parameters

- **application_name** (*str*) – Name of application on this side of relation

- **local_relation** (*str*) – Name of relation on this application
- **remote_relation** (*str '<application>[:<relation_name>]'*) – Name of relation on the other application.
- **model_name** (*str*) – Name of model to operate on.

async zaza.model.**async_reset_application_config**(*application_name*, *config_keys*, *model_name=None*)
Reset application configuration to default values.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application
- **config_keys** (*List[str]*) – List of configuration keys to reset to their defaults.

async zaza.model.**async_resolve_units**(*application_name=None*, *wait=True*, *timeout=60*, *erred_hook=None*, *model_name=None*)
Mark all the errored units as resolved or limit to an application.

Parameters

- **application_name** (*str*) – Name of application
- **wait** (*bool*) – Whether to wait for error state to have cleared.
- **timeout** (*int*) – Seconds to wait for an individual units state to clear.
- **erred_hook** (*str*) – Only resolve units that went into an error state when running the specified hook.
- **model_name** (*str*) – Name of model to query.

async zaza.model.**async_run_action**(*unit_name*, *action_name*, *model_name=None*, *action_params=None*, *raise_on_failure=False*)
Run action on given unit.

Parameters

- **unit_name** (*str*) – Name of unit to run action on
- **action_name** (*str*) – Name of action to run
- **model_name** (*str*) – Name of model to query.
- **action_params** (*dict*) – Dictionary of config options for action
- **raise_on_failure** (*bool*) – Raise ActionFailed exception on failure

Returns Action object

Return type juju.action.Action

Raises ActionFailed

async zaza.model.**async_run_action_on_leader**(*application_name*, *action_name*, *model_name=None*, *action_params=None*, *raise_on_failure=False*)
Run action on lead unit of the given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application
- **action_name** (*str*) – Name of action to run

- **action_params** (*dict*) – Dictionary of config options for action
- **raise_on_failure** (*bool*) – Raise ActionFailed exception on failure

Returns Action object

Return type juju.action.Action

Raises ActionFailed

async zaza.model.**async_run_action_on_units**(*units*, *action_name*, *action_params=None*, *model_name=None*, *raise_on_failure=False*, *timeout=600*)

Run action on list of unit in parallel.

The action is run on all units first without waiting for the action to complete. Then block until they are done.

Parameters

- **units** (*List[str]*) – List of unit names
- **action_name** (*str*) – Name of action to run
- **action_params** (*dict*) – Dictionary of config options for action
- **model_name** (*str*) – Name of model to query.
- **raise_on_failure** (*bool*) – Raise ActionFailed exception on failure
- **timeout** (*int*) – Time to wait for actions to complete

Returns Action object

Return type juju.action.Action

Raises ActionFailed

async zaza.model.**async_run_on_leader**(*application_name*, *command*, *model_name=None*, *timeout=None*)

Juju run on leader unit.

Parameters

- **application_name** (*str*) – Application to match
- **command** (*str*) – Command to execute
- **model_name** (*str*) – Name of model unit is in
- **timeout** (*int*) – How long in seconds to wait for command to complete

Returns action.data['results'] {'Code': '', 'Stderr': '', 'Stdout': ''}

Return type dict

async zaza.model.**async_run_on_machine**(*machine*, *command*, *model_name=None*, *timeout=None*)

Juju run on unit.

This function uses a spawned process to run the *juju run* command rather that a native libjuju call as libjuju hasn't implemented *juju.Machine.run* yet: https://github.com/juju/python-libjuju/issues/403

Parameters

- **model_name** (*str*) – Name of model unit is in
- **unit_name** – Name of unit to match
- **command** (*str*) – Command to execute
- **timeout** (*int*) – How long in seconds to wait for command to complete

Returns action.data['results'] {'Code': '', 'Stderr': '', 'Stdout': ''}

Return type dict

async zaza.model.**async_run_on_unit**(*unit_name*, *command*, *model_name=None*, *timeout=None*)
Juju run on unit.

Parameters

- **model_name** (*str*) – Name of model unit is in
- **unit_name** – Name of unit to match
- **command** (*str*) – Command to execute
- **timeout** (*int*) – How long in seconds to wait for command to complete

Returns action.data['results'] {'Code': '', 'Stderr': '', 'Stdout': ''}

Return type dict

async zaza.model.**async_scp_from_unit**(*unit_name*, *source*, *destination*, *model_name=None*, *user='ubuntu'*, *proxy=False*, *scp_opts=''*)
Transfer files from to unit_name in model_name.

Parameters

- **model_name** (*str*) – Name of model unit is in
- **unit_name** (*str*) – Name of unit to scp from
- **source** (*str*) – Remote path of file(s) to transfer
- **destination** – Local destination of transferred files
- **user** – Remote username
- **proxy** (*bool*) – Proxy through the Juju API server
- **scp_opts** (*str*) – Additional options to the scp command

async zaza.model.**async_scp_to_all_units**(*application_name*, *source*, *destination*, *model_name=None*, *user='ubuntu'*, *proxy=False*, *scp_opts=''*)
Transfer files from to all units of an application.

Parameters

- **model_name** (*str*) – Name of model unit is in
- **application_name** (*str*) – Name of application to scp file to
- **source** (*str*) – Local path of file(s) to transfer
- **destination** – Remote destination of transferred files
- **user** – Remote username
- **proxy** (*bool*) – Proxy through the Juju API server
- **scp_opts** (*str*) – Additional options to the scp command

async zaza.model.**async_scp_to_unit**(*unit_name*, *source*, *destination*, *model_name=None*, *user='ubuntu'*, *proxy=False*, *scp_opts=''*)
Transfer files to unit_name in model_name.

Parameters

- **model_name** (*str*) – Name of model unit is in
- **unit_name** (*str*) – Name of unit to scp to

- **source** (*str*) – Local path of file(s) to transfer
- **destination** – Remote destination of transferred files
- **user** – Remote username
- **proxy** (*bool*) – Proxy through the Juju API server
- **scp_opts** (*str*) – Additional options to the scp command

async zaza.model.**async_set_application_config**(*application_name*, *configuration*, *model_name=None*)
Set application configuration.

NOTE: At the time of this writing python-libjuju requires all values passed to *set_config* to be *str*. https://github.com/juju/python-libjuju/issues/388

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application
- **configuration** (*Dict[str,str]*) – Dictionary of configuration setting(s)

async zaza.model.**async_upgrade_charm**(*application_name*, *channel=None*, *force_series=False*, *force_units=False*, *path=None*, *resources=None*, *revision=None*, *switch=None*, *model_name=None*)
Upgrade the given charm.

Parameters

- **application_name** (*str*) – Name of application on this side of relation
- **channel** (*str*) – Channel to use when getting the charm from the charm store, e.g. 'development'
- **force_series** (*bool*) – Upgrade even if series of deployed application is not supported by the new charm
- **force_units** (*bool*) – Upgrade all units immediately, even if in error state
- **path** (*str*) – Uprade to a charm located at path
- **resources** (*dict*) – Dictionary of resource name/filepath pairs
- **revision** (*int*) – Explicit upgrade revision
- **switch** (*str*) – Crossgrade charm url
- **model_name** (*str*) – Name of model to operate on

async zaza.model.**async_wait_for_agent_status**(*model_name=None*, *status='executing'*, *timeout=60*)
Wait for at least one unit to enter a specific agent status.

This is useful for awaiting execution after mutating charm configuration.

Parameters

- **model_name** (*str*) – Name of model to query.
- **status** (*str*) – The desired agent status we are looking for.
- **timeout** (*int*) – Time to wait for status to be achieved.

async zaza.model.**async_wait_for_application_states**(*model_name=None*, *states=None*, *timeout=2700*)
Wait for model to achieve the desired state.

Check the workload status and workload status message for every unit of every application. By default look for an 'active' workload status and a message that starts with one of the approved_message_prefixes.

Bespoke statuses and messages can be passed in with states. states takes the form:

```
states = {
    'app': {
        'workload-status': 'blocked',
        'workload-status-message-prefix': 'No requests without a prod',
        'num-expected-units': 3,
    },
    'anotherapp': {
        'workload-status-message-prefix': 'Unit is super ready'}}
wait_for_application_states('modelname', states=states)
```

The keys that can be used are:

- "workload-status" - an exact match of the workload-status
- "workload-status-message-prefix" - an exact match, that starts with the string passed.
- "workload-status-message-regex" - the entire string matches if the regex matches.
- **"workloaed-status-message" - DEPRECATED; use** "workload-status-message-prefix" instead.
- "num-expected-units' - the number of units to be 'ready'.

To match an empty string, use: "workload-status-message-regex": "^$"

NOTE: all applications that are being waited on are expected to have at least one unit for that application to be ready. Any subordinate charms which have not be related to their principle by the time this function is called will 'hang' the function; in this case (if this is expected), then the application should be passed in the :param:states parameter with a 'num-expected-units' of 0 for the app in question.

Parameters

- **model_name** (*str*) – Name of model to query.
- **states** (*dict*) – States to look for
- **timeout** (*int*) – Time to wait for status to be achieved

async zaza.model.**async_wait_for_unit_idle**(*unit_name*, *timeout=600*, *include_subordinates=False*)

Wait until the unit's agent is idle.

Parameters

- **unit_name** (*str*) – The unit name of the application, ex: mysql/0
- **timeout** (*int*) – How long to wait before timing out
- **include_subordinates** (*bool*) – Should this function wait for subordinate idle

Returns None

Return type None

zaza.model.**attach_resource**(*application*, *resource_name*, *resource_path*)

Attach resource to charm.

Parameters

- **application** (*str*) – Application to get leader settings from.

- **resource_name** (*str*) – The name of the resource as defined in metadata.yaml
- **resource_path** (*str*) – The path to the resource on disk

Returns None

Return type None

async zaza.model.**block_until_auto_reconnect_model**(**conditions*, *model=None*, *aconditions=None*, *timeout=None*, *wait_period=0.5*, *loop=None*)

Async block on the model until conditions met.

This function doesn't use model.block_until() which unfortunately raises websockets.exceptions.ConnectionClosed if the connections gets closed, which seems to happen quite frequently. This funtion blocks until the conditions are met or a timeout occurs, and reconnects the model if it becomes disconnected.

Note that conditions are just passed as an unamed list in the function call to make it work more like the more simple 'block_until' function.

Parameters

- **model** (*:class:'juju.Model()'*) – the model to use
- **conditions** (*[List[Callable[[:class:'juju.Model()'], bool]]]*) – a list of callables that need to evaluate to True.
- **aconditions** (*Optional[List[AsyncCallable[[:class:'juju.Model()'], bool]]]*) – an optional list of async callables that need to evaluate to True.
- **timeout** (*float*) – the timeout to wait for the block on.
- **wait_period** (*float*) – The time to sleep between checking the conditions.
- **loop** (*An event loop*) – The event loop to use

Raises TimeoutError if the conditions never match (assuming timeout is not None).

zaza.model.**check_model_for_hard_errors**(*model*)

Check model for any hard errors that should halt a deployment.

The check for units or machines in an error state

Raises Union[UnitError, MachineError]

zaza.model.**check_unit_workload_status**(*model*, *unit*, *states*)

Check that the units workload status matches the supplied state.

This function has the side effect of also checking for *any* units in an error state and aborting if any are found.

Parameters

- **model** (*juju.Model*) – Model object to check in
- **unit** (*juju.Unit*) – Unit to check wl status of
- **states** (*List[str]*) – Acceptable unit work load states

Raises UnitError

Returns Whether units workload status matches desired state

Return type bool

zaza.model.**check_unit_workload_status_message**(*model*, *unit*, *message=None*, *prefixes=None*, *regex=None*)

Check that the units workload status message.

Check that the units workload status message matches the supplied message, matches a regular expression (regex) or starts with one of the supplied prefixes. Raises an exception if neither prefixes or message is set. This function has the side effect of also checking for *any* units in an error state and aborting if any are found.

Note that the priority of checking is: message, then regex, then prefixes.

Parameters

- **model** (*juju.Model*) – Model object to check in
- **unit** (*juju.Unit*) – Unit to check wl status of
- **message** (*Optiona[str]*) – Expected message text
- **prefixes** (*Optional[List[str]]*) – Prefixes to match message against
- **regex** (*Optional[str]*) – A regular expression against which to test the message

Raises ValueError, UnitError

Returns Whether message matches desired string

Return type bool

zaza.model.**complete_series_upgrade**(*machine_num*)

Execute juju series-upgrade complete on machine.

NOTE: This is a new feature in juju behind a feature flag and not yet in libjuju. export JUJU_DEV_FEATURE_FLAGS=upgrade-series

Parameters **machine_num** (*str*) – Machine number

Returns None

Return type None

async zaza.model.**deployed**(*model_name=None*)

List deployed applications.

Parameters **model_name** (*string*) – Name of the model to list applications from

async zaza.model.**ensure_model_connected**(*model*)

Ensure that the model is connected.

If model is disconnected then reconnect it.

Parameters **model** (*:class:'juju.Model'*) – the model to check

zaza.model.**file_contents**(*unit_name*, *path*, *timeout=None*)

Return the contents of a file.

Parameters

- **path** – File path
- **unit_name** (*str*) – Unit name, either appname/N or appname/leader
- **timeout** (*float*) – Timeout in seconds

Returns File contents

Return type str

zaza.model.**get_actions**(*application_name*, *model_name=None*)
Get the actions an applications supports.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns Dictionary of actions and their descriptions

Return type dict

zaza.model.**get_app_ips**(*application_name*, *model_name=None*)
Return public address of all units of an application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns List of ip addresses

Return type [str, str,...]

zaza.model.**get_first_unit_name**(*application_name*, *model_name=None*)
Return name of lowest numbered unit of given application.

Parameters

- **model_name** (*str*) – Name of model to query.
- **application_name** (*str*) – Name of application

Returns Name of lowest numbered unit

Return type str

zaza.model.**get_juju_model_aliases**()
Return the model aliases from global.

Returns Model alias map

Return type dict

zaza.model.**get_unit_from_name**(*unit_name*, *model=None*, *model_name=None*)
Return the units that corresponds to the name in the given model.

Parameters

- **unit_name** (*str*) – Name of unit to match
- **model** (*model.Model()*) – Model to perform lookup in
- **model_name** (*string*) – Name of the model to perform lookup in

Returns Unit matching given name

Return type juju.unit.Unit or None

Raises UnitNotFound

zaza.model.**is_model_disconnected**(*model*)
Return True if the model is disconnected.

Parameters **model** (*:class:'juju.Model'*) – the model to check

Returns True if disconnected

Return type bool

zaza.model.**is_unit_idle**(*unit*)
Return True if the unit is in the idle state.

Note: the unit only makes progress (in terms of updating it's idle status) if this function is called as part of an asyncio loop as the status is updated in a co-routine/future.

Parameters **unit** (*:class:'juju.unit.Unit'*) – the unit to test

Returns True if the unit is in the idle state

Return type bool

zaza.model.**machines_in_state**(*model*, *states*)
Check model for machines whose state is in the list 'states'.

Parameters

- **model** (*juju.Model*) – Model object to check
- **states** (*List*) – List of states to check for

Returns List of machines

Return type List[juju.machine.Machine]

zaza.model.**prepare_series_upgrade**(*machine_num*, *to_series='xenial'*)
Execute juju series-upgrade prepare on machine.

NOTE: This is a new feature in juju behind a feature flag and not yet in libjuju. export JUJU_DEV_FEATURE_FLAGS=upgrade-series

Parameters

- **machine_num** (*str*) – Machine number
- **to_series** (*str*) – The series to which to upgrade

Returns None

Return type None

zaza.model.**run_in_model**(*model_name*)
Context manager for executing code inside a libjuju model.

Example of using run_in_model:

async with run_in_model(model_name) as model: model.do_something()

Parameters **model_name** (*str*) – Name of model to run function in

Returns The juju Model object correcsponding to model_name

Return type Iterator[:class:'juju.Model()']

zaza.model.**set_juju_model**(*model_name*)
Point environment at the given model.

Parameters **model_name** (*str*) – Model to point environment at

zaza.model.**set_juju_model_aliases**(*model_aliases*)
Store the model aliases in a global.

Parameters **model_aliases** (*dict*) – Model alias map to store

zaza.model.**set_model_constraints**(*constraints*, *model_name=None*)
Set constraints on a model.

Note: Subprocessing out to 'juju' is a temporary solution until https://bit.ly/2ujbVPA lands in libjuju.

Parameters

- **constraints** (*dict*) – Constraints to be applied to model
- **model_name** (*str*) – Name of model to operate on

zaza.model.**set_series**(*application*, *to_series*)
Execute juju set-series complete on application.

NOTE: This is a new feature in juju and not yet in libjuju.

Parameters

- **application** (*str*) – Name of application to upgrade series
- **to_series** (*str*) – The series to which to upgrade

Returns None

Return type None

zaza.model.**units_with_wl_status_state**(*model*, *state*)
Return a list of unit which have a matching workload status.

Returns Units in error state

Return type [juju.Unit, ...]

zaza.model.**unset_juju_model_aliases**()
Remove model alias data.

3.3 OpenStack Utilities

3.4 TLS Certificate Utilities

Module for working with x.509 certificates.

zaza.utilities.cert.**generate_cert**(*common_name*, *alternative_names=None*, *password=None*, *issuer_name=None*, *signing_key=None*, *signing_key_password=None*, *generate_ca=False*)
Generate x.509 certificate.

Example of how to create a certificate chain:

```
(cakey, cacert) = generate_cert(
    'DivineAuthority',
    generate_ca=True)
(crkey, crcert) = generate_cert(
    'test.com',
    issuer_name='DivineAuthority',
    signing_key=cakey)
```

Parameters

- **common_name** (*str*) – Common Name to use in generated certificate
- **alternative_names** (*Optional[list(str)]*) – List of names to add as SubjectAlternativeName
- **password** (*Optional[str]*) – Password to protect encrypted private key with
- **issuer_name** (*Optional[str]*) – Issuer name, must match provided_private_key issuer
- **signing_key** (*Optional[str]*) – PEM encoded PKCS8 formatted private key
- **signing_key_password** (*Optional[str]*) – Password to decrypt private key
- **generate_ca** (*bool*) – Generate a certificate usable as a CA certificate

Returns x.509 certificate

Return type cryptography.x509.Certificate

zaza.utilities.cert.**is_keys_valid**(*public_key_string*, *private_key_string*)

Test whether these are a valid public/private key pair.

Parameters

- **public_key_string** (*str*) – PEM encoded key data.
- **private_key_string** (*str*) – OpenSSH encoded key data.

zaza.utilities.cert.**sign_csr**(*csr*, *ca_private_key*, *ca_cert=None*, *issuer_name=None*, *ca_private_key_password=None*, *generate_ca=False*)

Sign CSR with the given key.

Parameters

- **csr** (*str*) – Certificate to sign
- **ca_private_key** (*str*) – Private key to be used to sign csr
- **ca_cert** (*str*) – Cert to base some options from
- **issuer_name** (*Optional[str]*) – Issuer name, must match provided_private_key issuer
- **ca_private_key_password** (*Optional[str]*) – Password to decrypt ca_private_key
- **generate_ca** (*bool*) – Allow resulting cert to be used as ca

Returns x.509 certificate

Return type cryptography.x509.Certificate

3.5 CLI Utilities

Module containing utilities for working with commandline tools.

class zaza.utilities.cli.**StoreModelMapping**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Implement the Action API to process model arguments.

zaza.utilities.cli.**add_model_parser**(*parser*)

Add parser for model argument to supplied parser.

Parameters **parser** (*argparse.ArgumentParser*) – argparse parser

zaza.utilities.cli.**add_test_directory_argument**(*parser*)
Add parser for location of test directory.

Parameters **parser** (*argparse.ArgumentParser*) – argparse parser

zaza.utilities.cli.**parse_arg**(*options*, *arg*, *multiargs=False*)
Parse argparse argments.

Parameters

- **options** (*argparse object*) – Argparse options
- **arg** (*string*) – Argument attribute key
- **multiargs** (*bool*) – More than one arugment or not

Returns Argparse atrribute value

Return type string

zaza.utilities.cli.**setup_logging**(*log_level='INFO'*)
Do setup for logging.

Returns Nothing: This fucntion is executed for its sideffect

Return type None

3.6 Utility Exceptions

Module of exceptions that zaza may raise.

exception zaza.utilities.exceptions.**ApplicationNotFound**(*application*)
Application not found in machines.

exception zaza.utilities.exceptions.**CephPoolNotFound**
Ceph pool not found.

exception zaza.utilities.exceptions.**CloudInitIncomplete**
Cloud init has not completed properly.

exception zaza.utilities.exceptions.**DestroyModelFailed**
The controller.destroy_model() failed in some interesting way.

exception zaza.utilities.exceptions.**JujuError**
Exception when libjuju does something unexpected.

exception zaza.utilities.exceptions.**KeystoneAuthorizationPermissive**
Authorization/Policy too permissive.

exception zaza.utilities.exceptions.**KeystoneAuthorizationStrict**
Authorization/Policy too strict.

exception zaza.utilities.exceptions.**KeystoneKeyRepositoryError**
Error in key repository.

This may be caused by isues with one of: - incomplete or missing data in *key_repository* in leader storage - synchronization of keys to non-leader units - rotation of keys

exception zaza.utilities.exceptions.**KeystoneWrongTokenProvider**
A token was issued from the wrong token provider.

exception zaza.utilities.exceptions.**MachineNotFound**
Exception when machine is not found.

exception zaza.utilities.exceptions.**MissingOSAthenticationException**
Exception when some data needed to authenticate is missing.

exception zaza.utilities.exceptions.**NeutronAgentMissing**
Agent binary does not appear in the Neutron agent list.

exception zaza.utilities.exceptions.**NeutronBGPSpeakerMissing**
No BGP speaker appeared on agent.

exception zaza.utilities.exceptions.**NovaGuestMigrationFailed**
Nova guest migration failed.

exception zaza.utilities.exceptions.**NovaGuestRestartFailed**
Nova guest restart failed.

exception zaza.utilities.exceptions.**OSVersionNotFound**
OS Version not found.

exception zaza.utilities.exceptions.**PIDCountMismatch**
PID's count doesn't match.

exception zaza.utilities.exceptions.**ProcessIdsFailed**
Process ID lookup failed.

exception zaza.utilities.exceptions.**ProcessNameCountMismatch**
Count of process names doesn't match.

exception zaza.utilities.exceptions.**ProcessNameMismatch**
Name of processes doesn't match.

exception zaza.utilities.exceptions.**ReleasePairNotFound**
Release pair was not found in OPENSTACK_RELEASES_PAIRS.

exception zaza.utilities.exceptions.**SSHFailed**
SSH failed.

exception zaza.utilities.exceptions.**SeriesNotFound**
Series not found in status.

exception zaza.utilities.exceptions.**ServiceNotFound**
Service not found on unit.

exception zaza.utilities.exceptions.**TemplateConflict**
Exception when templates are in conflict.

exception zaza.utilities.exceptions.**UbuntuReleaseNotFound**
Ubuntu release not found in list.

exception zaza.utilities.exceptions.**UnitCountMismatch**
Count of unit doesn't match.

exception zaza.utilities.exceptions.**UnitNotFound**
Unit not found in actual dict.

3.7 Generic Utilities

Collection of functions that did not fit anywhere else.

async zaza.utilities.generic.**check_call**(*cmd*)
Asynchronous function to check a subprocess call.

Parameters **cmd** (*List[str]*) – Command to execute

Returns None

Return type None

Raises subprocess.CalledProcessError if returncode !=0

async zaza.utilities.generic.**check_output**(*cmd*)
Asynchronous function to run a subprocess and get the output.

Note, as the code raises an Exception on returncode != 0, 'Code' in the dictionary will always be '0'. This is included for compatability reasons.

Parameters **cmd** (*List[str]*) – Command to execute

Returns {'Code': '', 'Stderr': '', 'Stdout': ''}

Return type dict

Raises subprocess.CalledProcessError if returncode !=0

zaza.utilities.generic.**dict_to_yaml**(*dict_data*)
Return YAML from dictionary.

Parameters **dict_data** (*dict*) – Dictionary data

Returns YAML dump

Return type string

zaza.utilities.generic.**do_release_upgrade**(*unit_name*)
Run do-release-upgrade noninteractive.

Parameters **unit_name** (*str*) – Unit Name

Returns None

Return type None

zaza.utilities.generic.**get_network_config**(*net_topology*, *ignore_env_vars=False*, *net_topology_file='network.yaml'*)
Get network info from environment.

Get network info from network.yaml, override the values if specific environment variables are set for the undercloud.

This function may be used when running network configuration from CLI to pass in network configuration settings from a YAML file.

Parameters

- **net_topology** (*string*) – Network topology name from network.yaml
- **ignore_env_vars** (*bool*) – Ignore enviroment variables or not

Returns Dictionary of network configuration

Return type dict

zaza.utilities.generic.**get_pkg_version**(*application*, *pkg*)
Return package version.

Parameters

- **application** (*string*) – Application name
- **pkg** (*string*) – Package name

Returns List of package version

Return type list

zaza.utilities.generic.**get_process_id_list**(*unit_name*, *process_name*, *expect_success=True*, *pgrep_full=False*)
Get a list of process ID(s).

Get a list of process ID(s) from a single sentry juju unit for a single process name.

Parameters

- **unit_name** – Amulet sentry instance (juju unit)
- **process_name** – Process name
- **expect_success** – If False, expect the PID to be missing, raise if it is present.
- **pgrep_full** (*bool*) – Should pgrep be used rather than pidof to identify a service.

Returns List of process IDs

Raises zaza_exceptions.ProcessIdsFailed

zaza.utilities.generic.**get_undercloud_env_vars**()
Get environment specific undercloud network configuration settings.

Get environment specific undercloud network configuration settings from environment variables.

For each testing substrate, specific undercloud network configuration settings should be exported into the environment to enable testing on that substrate.

Note: *Overcloud* settings should be declared by the test caller and should not be overridden here.

Return a dictionary compatible with zaza.openstack.configure.network functions' expected key structure.

Example exported environment variables: export default_gateway="172.17.107.1" export external_net_cidr="172.17.107.0/24" export external_dns="10.5.0.2" export start_floating_ip="172.17.107.200" export end_floating_ip="172.17.107.249"

Example o-c-t & uosci non-standard environment variables: export NET_ID="a705dd0f-5571-4818-8c30-4132cc494668" export GATEWAY="172.17.107.1" export CIDR_EXT="172.17.107.0/24" export NAMESERVER="10.5.0.2" export FIP_RANGE="172.17.107.200:172.17.107.249"

Returns Network environment variables

Return type dict

zaza.utilities.generic.**get_unit_process_ids**(*unit_processes*, *expect_success=True*)
Get unit process ID(s).

Construct a dict containing unit sentries, process names, and process IDs.

Parameters

- **unit_processes** – A dictionary of unit names to list of process names.
- **expect_success** – if False expect the processes to not be running, raise if they are.

Returns Dictionary of unit names to dictionary of process names to PIDs.

Raises zaza_exceptions.ProcessIdsFailed

zaza.utilities.generic.**get_yaml_config**(*config_file*)
Return configuration from YAML file.

Parameters **config_file** (*string*) – Configuration file name

Returns Dictionary of configuration

Return type dict

zaza.utilities.generic.**reboot**(*unit_name*)
Reboot unit.

Parameters **unit_name** (*str*) – Unit Name

Returns None

Return type None

zaza.utilities.generic.**run_via_ssh**(*unit_name*, *cmd*)
Run command on unit via ssh.

For executing commands on units when the juju agent is down.

Parameters

- **unit_name** – Unit Name
- **cmd** (*str*) – Command to execute on remote unit

Returns None

Return type None

zaza.utilities.generic.**series_upgrade**(*unit_name*, *machine_num*, *from_series='trusty'*, *to_series='xenial'*, *origin='openstack-origin'*, *files=None*, *workaround_script=None*)
Perform series upgrade on a unit.

Parameters

- **unit_name** (*str*) – Unit Name
- **machine_num** (*str*) – Machine number
- **from_series** (*str*) – The series from which to upgrade
- **to_series** (*str*) – The series to which to upgrade
- **origin** (*str*) – The configuration setting variable name for changing origin source. (openstack-origin or source)
- **files** (*list*) – Workaround files to scp to unit under upgrade
- **workaround_script** (*str*) – Workaround script to run during series upgrade

Returns None

Return type None

zaza.utilities.generic.**series_upgrade_application**(*application*, *pause_non_leader_primary=True*, *pause_non_leader_subordinate=True*, *from_series='trusty'*, *to_series='xenial'*, *origin='openstack-origin'*, *completed_machines=[]*, *files=None*, *workaround_script=None*)

Series upgrade application.

Wrap all the functionality to handle series upgrade for a given application. Including pausing non-leader units.

Parameters

- **application** (*str*) – Name of application to upgrade series
- **pause_non_leader_primary** (*bool*) – Whether the non-leader applications should be paused
- **pause_non_leader_subordinate** (*bool*) – Whether the non-leader subordinate hacluster applications should be paused
- **from_series** (*str*) – The series from which to upgrade
- **to_series** (*str*) – The series to which to upgrade
- **origin** (*str*) – The configuration setting variable name for changing origin source. (openstack-origin or source)
- **completed_machines** (*list*) – List of completed machines which do no longer require series upgrade.
- **files** (*list*) – Workaround files to scp to unit under upgrade
- **workaround_script** (*str*) – Workaround script to run during series upgrade

Returns None

Return type None

zaza.utilities.generic.**series_upgrade_non_leaders_first**(*application*, *from_series='trusty'*, *to_series='xenial'*, *completed_machines=[]*)

Series upgrade non leaders first.

Wrap all the functionality to handle series upgrade for charms which must have non leaders upgraded first.

Parameters

- **application** (*str*) – Name of application to upgrade series
- **from_series** (*str*) – The series from which to upgrade
- **to_series** (*str*) – The series to which to upgrade
- **completed_machines** (*list*) – List of completed machines which do no longer require series upgrade.

Returns None

Return type None

zaza.utilities.generic.**set_dpkg_non_interactive_on_unit**(*unit_name*, *apt_conf_d='/etc/apt/apt.conf.d/50unattended-upgrades'*)

Set dpkg options on unit.

Parameters

- **unit_name** (*str*) – Unit Name
- **apt_conf_d** (*str*) – Apt.conf file to update

zaza.utilities.generic.**set_origin**(*application*, *origin='openstack-origin'*, *pocket='distro'*)
Set the configuration option for origin source.

Parameters

- **application** (*str*) – Name of application to upgrade series
- **origin** (*str*) – The configuration setting variable name for changing origin source. (openstack-origin or source)
- **pocket** (*str*) – Origin source cloud pocket. i.e. 'distro' or 'cloud:xenial-newton'

Returns None

Return type None

zaza.utilities.generic.**validate_unit_process_ids**(*expected*, *actual*)
Validate process id quantities for services on units.

Returns True if the PIDs are validated, raises an exception if it is not the case.

Raises zaza_exceptions.UnitCountMismatch

Raises zaza_exceptions.UnitNotFound

Raises zaza_exceptions.ProcessNameCountMismatch

Raises zaza_exceptions.ProcessNameMismatch

Raises zaza_exceptions.PIDCountMismatch

zaza.utilities.generic.**wrap_do_release_upgrade**(*unit_name*, *from_series='trusty'*, *to_series='xenial'*, *files=None*, *workaround_script=None*)
Wrap do release upgrade.

In a production environment this step would be run administratively. For testing purposes we need this automated.

Parameters

- **unit_name** (*str*) – Unit Name
- **from_series** (*str*) – The series from which to upgrade
- **to_series** (*str*) – The series to which to upgrade
- **files** (*list*) – Workaround files to scp to unit under upgrade
- **workaround_script** (*str*) – Workaround script to run during series upgrade

Returns None

Return type None

3.8 Juju Utilities

Module for interacting with juju.

zaza.utilities.juju.**get_application_ip**(*application*, *model_name=None*)
Get the application's IP address.

Parameters

- **application** (*str*) – Application name
- **model_name** (*str*) – Name of model to query.

Returns Application's IP address

Return type str

zaza.utilities.juju.**get_application_status**(*application=None*, *unit=None*, *model_name=None*)
Return the juju status for an application.

Parameters

- **application** (*string*) – Application name
- **unit** (*string*) – Specific unit
- **model_name** (*str*) – Name of model to query.

Returns Juju status output for an application

Return type dict

zaza.utilities.juju.**get_cloud_configs**(*cloud=None*)
Get cloud configuration from local clouds.yaml.

libjuju does not yet have cloud information implemented. Use libjuju as soon as possible.

Parameters **cloud** – Name of specific cloud

Returns Dictionary of cloud configuration

Return type dict

zaza.utilities.juju.**get_full_juju_status**(*model_name=None*)
Return the full juju status output.

Parameters **model_name** (*str*) – Name of model to query.

Returns Full juju status output

Return type dict

zaza.utilities.juju.**get_machine_series**(*machine*, *model_name=None*)
Return the juju series for a machine.

Parameters

- **machine** (*string*) – Machine number
- **model_name** (*str*) – Name of model to query.

Returns Juju series

Return type string

zaza.utilities.juju.**get_machine_status**(*machine*, *key=None*, *model_name=None*)
Return the juju status for a machine.

Parameters

- **machine** (*string*) – Machine number
- **key** (*string*) – Key option requested
- **model_name** (*str*) – Name of model to query.

Returns Juju status output for a machine

Return type dict

zaza.utilities.juju.**get_machine_uuids_for_application**(*application*, *model_name=None*)
Return machine uuids for a given application.

Parameters

- **application** (*string*) – Application name
- **model_name** (*str*) – Name of model to query.

Returns List of machine uuuids for an application

Return type list

zaza.utilities.juju.**get_machines_for_application**(*application*, *model_name=None*)
Return machines for a given application.

Parameters

- **application** (*string*) – Application name
- **model_name** (*str*) – Name of model to query.

Returns List of machines for an application

Return type list

zaza.utilities.juju.**get_principle_applications**(*application_name*, *application_status=None*, *model_name=None*)
Get the principle applications that application_name is related to.

Parameters

- **application_name** (*string*) – Application name
- **application_status** (*dict*) – Juju status dict for application
- **model_name** (*str*) – Name of model to query.

Returns List of principle applications

Return type list

zaza.utilities.juju.**get_provider_type**()
Get the type of the undercloud.

If it can't work it out (i.e. it might be a localhost lxd install where there is no clouds.yaml) then return None.

If libjuju doesn't know (i.e. cloud is not truthy) then default to openstack.

Returns Name of the undercloud type

Return type Optional[string]

zaza.utilities.juju.**get_relation_from_unit**(*entity*, *remote_entity*, *remote_interface_name*, *model_name=None*)
Get relation data passed between two units.

Get relation data for relation with *remote_interface_name* between *entity* and *remote_entity* from the perspective of *entity*.

entity and *remote_entity* may refer to either a application or a specific unit. If application name is given first unit is found in model.

Parameters

- **entity** (*str*) – Application or unit to get relation data from
- **remote_entity** (*str*) – Application or Unit in the other end of the relation we want to query
- **remote_interface_name** (*str*) – Name of interface to query on remote end of relation
- **model_name** (*str*) – Name of model to query.

Returns dict with relation data

Return type dict

Raises model.CommandRunFailed

zaza.utilities.juju.**get_subordinate_units**(*unit_list*, *charm_name=None*, *status=None*, *model_name=None*)

Get a list of all subordinate units associated with units in unit_list.

Get a list of all subordinate units associated with units in unit_list. Subordinate can be filtered by using 'charm_name' which will only return subordinate units which have 'charm_name' in the name of the charm e.g.

get_subordinate_units(

['cinder/1']) would return ['cinder-hacluster/1', 'cinder-ceph/2'])

where as

get_subordinate_units(['cinder/1'], charm_name='hac') would return ['cinder-hacluster/1']

NOTE: The charm_name match is against the name of the charm not the application name.

Parameters

- **charm_name** (*str*) – List of unit names
- **charm_name** – charm_name to match against, can be a sub-string.
- **status** (*juju.client._definitions.FullStatus*) – Juju status to query against,
- **model_name** (*str*) – Name of model to query.

Returns List of matching unit names.

Return type []

zaza.utilities.juju.**get_unit_name_from_host_name**(*host_name*, *application_name*, *model_name=None*)

Return the juju unit name corresponding to a hostname.

Parameters

- **host_name** (*string*) – Host name to map to unit name.
- **application_name** (*string*) – Application name
- **model_name** (*str*) – Name of model to query.

Returns The unit name of the application running on host_name.

Return type str or None

zaza.utilities.juju.**get_unit_name_from_ip_address**(*ip*, *application_name*, *model_name=None*)
Return the juju unit name corresponding to an IP address.

Parameters

- **ip** (*string*) – IP address to map to unit name.
- **application_name** (*string*) – Application name
- **model_name** (*str*) – Name of model to query.

zaza.utilities.juju.**is_subordinate_application**(*application_name*, *application_status=None*, *model_name=None*)
Is the given application a subordinate application.

Parameters

- **application_name** (*string*) – Application name
- **application_status** (*dict*) – Juju status dict for application
- **model_name** (*str*) – Name of model to query.

Returns Whether application_name is a subordinate

Return type bool

zaza.utilities.juju.**leader_get**(*application*, *key=''*, *model_name=None*)
Get leader settings from leader unit of named application.

Parameters

- **application** (*str*) – Application to get leader settings from.
- **model_name** (*str*) – Name of model to query.

Returns dict with leader settings

Return type dict

Raises model.CommandRunFailed

zaza.utilities.juju.**remote_run**(*unit*, *remote_cmd*, *timeout=None*, *fatal=None*, *model_name=None*)
Run command on unit and return the output.

NOTE: This function is pre-deprecated. As soon as libjuju unit.run is able to return output this functionality should move to model.run_on_unit.

Parameters

- **remote_cmd** (*string*) – Command to execute on unit
- **timeout** – Timeout value for the command
- **fatal** (*bool*) – Command failure condidered fatal or not
- **model_name** (*str*) – Name of model to query.

Returns Juju run output

Return type string

Raises model.CommandRunFailed

3.9 Test Utils

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

Z

INDEX

A

W

Z