
Virt Test Documentation

Release 2014.04.14-1438-g0ce39

Virt Test Team

March 02, 2017

1	What is virt-test?	3
1.1	Contribution docs	3
1.2	Getting started docs	4
1.3	Advanced docs	25
1.4	Extra docs	108
1.5	virttest	121
2	Indices and tables	567
	Python Module Index	569

Note: This repo is effectively read-only and won't receive updates anymore (it's here for historical purposes).

The development of this test suite has been moved to:

<https://github.com/avocado-framework/avocado-vt>

and the up to date documentation can be seen in:

<http://avocado-vt.readthedocs.org/>

What is virt-test?

`virt-test` is a suite of tests made to exercise different linux virtualization hypervisors and related tools. Although many people make this mistake, *this test suite is not autotest*. Autotest is a larger test framework on which `virt-test` is built.

If you need information about autotest, [you should check out its project page](#).

Contents:

Contribution docs

This section documents the procedures necessary to contribute to `virt-test`. Much of the development workflow comes from the parent project autotest.

Contents:

Contact information

- [Virt-test-devel mailing list](#) (new mailing list hosted by red hat)
- IRC channel: `irc.oftc.net #virt-test`

Downloading the Source

The main source is maintained on git and may be cloned as below:

```
git clone git://github.com/autotest/virt-test.git
```

If you want to learn how to use git as an effective contribution tool, consider reading [the git workflow autotest docs](#)

Virt Test Development Workflow

As this is a generic procedure we use among autotest subprojects, relevant documentation can be seen on our main wiki:

<https://github.com/autotest/autotest/wiki/DevelopmentWorkflow>

Code Submission Check List

As this is a procedure shared among autotest managed projects, we hereby link the docs to the main autotest wiki:

<https://github.com/autotest/autotest/wiki/SubmissionChecklist>

Getting started docs

Virt Test is a complex project, with a lot of code and conventions to learn. Here we explain how to run your first test jobs and to write your first tests.

Contents:

Introduction to Virt Test

Virt-test's main purpose is to serve as an automated regression testing tool for virt developers, and for doing regular automated testing of virt technologies (provided you use it with the server testing infrastructure).

Autotest is a project that aims to provide tools and libraries to perform automated testing on the linux platform. *virt-test* is a subproject under the autotest umbrella. For more information on autotest, see [the autotest home page](#).

virt-test aims to be a centralizing project for most of the virt functional and performance testing needs. We cover:

- Guest OS install, for both Windows (WinXP - Win7) and Linux (RHEL, Fedora, OpenSUSE and others through step engine mechanism)
- Serial output for Linux guests
- Migration, networking, timedrift and other types of tests

For the qemu subtests, we can do things like:

- Monitor control for both human and QMP protocols
- Build and use qemu using various methods (source tarball, git repo, rpm)
- Some level of performance testing can be made.
- The KVM unit tests can be run comfortably from inside virt-test, we do have full integration with the unittest execution

We support x86_64 hosts with hardware virtualization support (AMD and Intel), and Intel 32 and 64 bit guest operating systems.

For an overview about virt-test, how this project was created, its goals, structure, and how to develop simple tests, you can refer to the KVM forum 2010 slides.

Getting Started

Pre-requisites

1. A supported host platforms: Red Hat Enterprise Linux (RHEL) or Fedora. OpenSUSE should also work, but currently autotest is still not packaged for it, which means you have to clone autotest and put its path in an env variable so virt tests can find the autotest libs. Debian/Ubuntu now have a new experimental package that allows one to run the virt tests in a fairly straight forward way.
2. [Install software packages \(RHEL/Fedora\)](#)

3. Install software packages (Debian/Ubuntu)
4. A copy of the virt test source

For the impatient

1. Clone the virt test repo

```
git clone git://github.com/autotest/virt-test.git
```

2. Get into the base dir

```
cd virt-test
```

3. Run the bootstrap procedure. For example, if you want to run the qemu subtest, you will run:

```
./run -t qemu --bootstrap
```

This script will check if you have the minimum requirements for the test (required commands and includes), and download the JeOS image. You can omit running this script, since the code of this script also gets to run when you call the test runner, but it is discouraged. Explicitly running `get_started.py` first is interactive, and gives you a better idea of what is going on.

4. For qemu and libvirt subtests, the default test set does not require root. However, other tests might fail due to lack of privileges.

```
$ ./run -t qemu
```

or

```
# ./run -t libvirt
```

If you ran `get_started.py`, the test runner should just run the test. If you didn't, the runner will trigger the environment setup procedure:

1. Create the `/var/tmp/libvirt_test` dir to hold images and isos
2. Download the JeOS image (180 MB, takes about 3 minutes on a fast connection) and uncompress it (takes about a minute on an HDD laptop). `p7ip` has to be present.
3. Run a predefined set of tests.

Running different tests

You can list the available tests to run by using the flag `--list-tests`

```
$ ./run -t qemu --list-tests
(will print a numbered list of tests, with a paginator)
```

Then you can pass tests that interest you with `--tests` "list of tests", for example:

1. qemu

```
$ ./run -t qemu --tests "migrate timedrift file_transfer"
```

2. Libvirt requires first importing the JeOS image. However, this cannot be done if the guest already exists. Therefore, it's wise to also conclude a set with the `remove_guest.without_disk` test.

```
# ./run -t libvirt --tests "unattended_install.import.import boot reboot remove_guest.without_disk"
```

Checking the results

The test runner will produce a debug log, that will be useful to debug problems:

```
$ ./run -t qemu --tests usb
Running setup. Please wait...
SETUP: PASS (13.52 s)
DATA DIR: /home/lmr/virt_test
DEBUG LOG: /home/lmr/Code/virt-test.git/logs/run-2014-01-27-14.25.31/debug.log
TESTS: 203
(1/203) type_specific.io-github-autotest-qemu.usb.usb_boot.usb_kbd.without_usb_hub.uhci: PASS (25.47
(2/203) type_specific.io-github-autotest-qemu.usb.usb_boot.usb_kbd.without_usb_hub.ehci: PASS (23.53
(3/203) type_specific.io-github-autotest-qemu.usb.usb_boot.usb_kbd.without_usb_hub.xhci: PASS (24.34
...
```

Here you can see that the debug log is in `/home/lmr/Code/virt-test.git/logs/run-2014-01-27-14.25.31/debug.log`. For convenience, the most recent log is pointed to by the `logs/latest` symlink.

Virt Test Runner

As you probably know, virt tests was derived from a set of tests written for the autotest testing framework. Therefore, the test suite depended entirely on autotest for libraries *and* the autotest test harness to execute the test code.

However, autotest is a large framework, that forced a steep learning curve for people and a lot of code download (the autotest git repo is quite large these days, due to more than 6 years of history).

Due to this, virt tests was separated to its own test suite project, that still can run fine under autotest (in fact, it is what we use to do daily fully automated testing of KVM and QEMU), but that can be executed separately, depending only on a handful of autotest libraries.

This doc assumes you already read the introductory GetStarted documentation. This extra doc is just to teach you some useful tricks when using the runner.

Getting Help

The best way to get help from the command line options is the `--help` flag:

```
./run --help
```

General Flow

The test runner is nothing more than a very simple test harness, that replaces the autotest harness, and a set of options, that will trigger actions to create a test list and execute it. The way the tests work is:

1. Get a dict with test parameters
2. Based on these params, prepare the environment - create or destroy vm instances, create/check disk images, among others
3. Execute the test itself, that will use several of the params defined to carry on with its operations, that usually involve:
4. If a test did not raise an exception, it PASSEd
5. If a test raised an exception it FAILed
6. Based on what happened during the test, perform cleanup actions, such as killing vms, and remove unused disk images.

The list of parameters is obtained by parsing a set of configuration files, present inside the SourceStructure. The command line options usually modify even further the parser file, so we can introduce new data in the config set.

Common Operations – Listing guests

If you want to see all guests defined, you can use

```
./run -t [test type] --list-guests
```

This will generate a list of possible guests that can be used for tests, provided that you have an image with them. The list will show which guests don't have an image currently available. If you did perform the usual bootstrap procedure, only JeOS.17.64 will be available.

Now, let's assume you have the image for another guest. Let's say you've installed Fedora 17, 64 bits, and that `--list-guests` shows it as downloaded

```
./run -t qemu --list-guests
... snip...
16 Fedora.17.32 (missing f17-32.qcow2)
17 Fedora.17.64
18 Fedora.8.32 (missing f8-32.qcow2)
```

You can list all the available tests for Fedora.17.64 (you must use the exact string printed by the test, minus obviously the index number, that's there only for informational purposes:

```
./run -t qemu -g Fedora.17.64 --list-tests
... snip ...
26 balloon_check.base
27 balloon_check.balloon-migrate
28 balloon_check.balloon-shutdown_enlarge
29 balloon_check.balloon-shutdown_evict
30 block_mirror
31 block_stream
... snip ...
```

Then you can execute one in particular. It's the same idea, just copy the individual test you want and run it:

```
./run -t qemu -g Fedora.17.64 --tests balloon_check.balloon-migrate
```

And it'll run that particular test.

Tip: By the rules of the cartesian config files, you can use:

```
./run -t qemu -g Fedora.17.64 --tests balloon_check
```

And it'll run all tests from 26-29. Very useful for large sets, such as virtio_console and usb - You can just do a:

```
./run -t qemu --tests virtio_console
... 118 tests ...
```

```
./run -t qemu --tests usb
... 64 tests ...
```

Note that in the examples above, the fact I didn't provide `-g` means that we're using the default guest OS, that is, JeOS.

Install prerequisite packages

We need git and autotest, not available on RHEL repos. So, on RHEL hosts run first:

```
rpm -ivh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

To install [EPEL](#) repos. It is important to note that EPEL is needed with the sole purpose of providing a git RHEL package. If you can manage to install git from somewhere else, then this is not necessary. Check [here](#) for up to date EPEL RPM repo location.

Install the following packages:

1. Install a toolchain in your host, which you can do with Fedora and RHEL with:

```
yum groupinstall "Development Tools"
```

1. Install tcpdump, necessary to determine guest IPs automatically

```
yum install tcpdump
```

1. Install nc, necessary to get output from the serial device and other qemu devices

```
yum install nmap-ncat
```

1. Install the p7zip file archiver so you can uncompress the JeOS [2] image.

```
yum install p7zip
```

1. Install the autotest-framework package, to provide the needed autotest libs.

```
yum install --enablerepo=updates-testing autotest-framework
```

#. Install the fakeroot package, if you want to install from the CD Ubuntu and Debian servers without requiring root:

```
yum install fakeroot
```

If you don't install the autotest-framework package (say, your distro still doesn't have autotest packages, or you don't want to install the rpm), you'll have to clone an autotest tree and export this path as the AUTOTEST_PATH variable, both as root and as your regular user. One could put the following on their ~/.bashrc file:

```
export AUTOTEST_PATH="/path/to/autotest"
```

where this AUTOTEST_PATH will guide the run script to set up the needed libraries for all tests to work.

For other packages:

```
yum install git
```

So you can checkout the source code. If you want to test the distro provided qemu-kvm binary, you can install:

```
yum install qemu-kvm qemu-kvm-tools
```

To run libvirt tests, it's required to install the virt-install utility, for the basic purpose of building and cloning virtual machines.

```
yum install virt-install
```

To run all tests that involve filedescriptor passing, you need python-devel. The reason is, this test suite is compatible with python 2.4, whereas a std lib to pass filedescriptors was only introduced in python 3.2. Therefore, we had to introduce a C python extension that is compiled on demand.

```
yum install python-devel.
```

It's useful to also install:

```
yum install python-imaging
```

Not vital, but very handy to do imaging conversion from ppm to jpeg and png (allows for smaller images).

Tests that are not part of the default JeOS set

If you want to run guest install tests, you need to be able to create floppies and isos to hold kickstart files:

```
yum install mkisofs
```

For newer distros, such as Fedora, you'll need:

```
yum install genisoimage
```

Both packages provide the same functionality, needed to create iso images that will be used during the guest installation process. You can also execute

Network tests

Last but not least, now we depend on libvirt to provide us a stable, working bridge. * By default, the kvm test uses user networking, so this is not entirely necessary. However, non root and user space networking make a good deal of the hardcoded networking tests to not work. If you might want to use bridges eventually:

```
yum install libvirt bridge-utils
```

Make sure libvirtd is started:

```
[lmr@freedom autotest.lmr]$ service libvirtd start
```

Make sure the libvirt bridge shows up on the output of brctl show:

```
[lmr@freedom autotest.lmr]$ brctl show
bridge name bridge id          STP enabled interfaces
virbr0      8000.525400678eec    yes      virbr0-nic
```

Install prerequisite packages - Debian

Keep in mind that the current autotest package is a work in progress. For the purposes of running virt-tests it is fine, but it needs a lot of improvements until it can become a more 'official' package.

The autotest debian package repo can be found at <https://launchpad.net/~lmr/+archive/autotest>, and you can add the repos on your system putting the following on /etc/apt/sources.list:

```
deb http://ppa.launchpad.net/lmr/autotest/ubuntu raring main
deb-src http://ppa.launchpad.net/lmr/autotest/ubuntu raring main
```

Then update your software list:

```
apt-get update
```

This has been tested with Ubuntu 12.04, 12.10 and 13.04.

Install the following packages:

1. Install the autotest-framework package, to provide the needed autotest libs.

```
apt-get install autotest
```

1. Install the p7zip file archiver so you can uncompress the JeOS [2] image.

```
apt-get install p7zip-full
```

1. Install tcpdump, necessary to determine guest IPs automatically

```
apt-get install tcpdump
```

1. Install nc, necessary to get output from the serial device and other qemu devices

```
apt-get install netcat-openbsd
```

1. Install a toolchain in your host, which you can do on Debian and Ubuntu with:

```
apt-get install build-essential
```

#. Install fakeroot if you want to install from CD debian and ubuntu, not requiring root:

```
apt-get install fakeroot
```

So you install the core autotest libraries to run the tests.

If you don't install the autotest-framework package (say, your distro still doesn't have autotest packages, or you don't want to install the rpm), you'll have to clone an autotest tree and export this path as the AUTOTEST_PATH variable, both as root and as your regular user. One could put the following on their ~/.bashrc file:

```
export AUTOTEST_PATH="/path/to/autotest"
```

where this AUTOTEST_PATH will guide the run script to set up the needed libraries for all tests to work.

For other packages:

```
apt-get install git
```

So you can checkout the source code. If you want to test the distro provided qemu-kvm binary, you can install:

```
apt-get install qemu-kvm qemu-utils
```

To run libvirt tests, it's required to install the virt-install utility, for the basic purpose of building and cloning virtual machines.

```
apt-get install virtinst
```

To run all tests that involve filedescriptor passing, you need python-all-dev. The reason is, this test suite is compatible with python 2.4, whereas a std lib to pass filedescriptors was only introduced in python 3.2. Therefore, we had to introduce a C python extension that is compiled on demand.

```
apt-get install python-all-dev.
```

It's useful to also install:

```
apt-get install python-imaging
```

Not vital, but very handy to do imaging conversion from ppm to jpeg and png (allows for smaller images).

Tests that are not part of the default JeOS set

If you want to run guest install tests, you need to be able to create floppies and isos to hold kickstart files:

```
apt-get install genisoimage
```

Network tests

Last bug not least, now we depend on libvirt to provide us a stable, working bridge. * By default, the kvm test uses user networking, so this is not entirely necessary. However, non root and user space networking make a good deal of the hardcode networking tests to not work. If you might want to use bridges eventually:

```
apt-get install libvirt-bin python-libvirt bridge-utils
```

Make sure libvirtd is started:

```
[lmr@freedom autotest.lmr]$ service libvirtd start
```

Make sure the libvirt bridge shows up on the output of brctl show:

```
[lmr@freedom autotest.lmr]$ brctl show
bridge name bridge id          STP enabled interfaces
virbr0      8000.525400678eec    yes      virbr0-nic
```

Writing Virt Tests

This documentation aims to help you write virt tests of your own. It's organized to explain briefly the source structure, then writing simple tests, then doing more complex stuff, such as defining custom guests.

Contents:

Virt test source structure

When starting to contribute to a project, a high level description of the directory structure is frequently useful. In virt-tests, at the time of this writing (01-09-2013), the output of the command *tree* for this structure looks like:

```
.
|-- libvirt
|   |-- cfg
|   `-- tests
|       |-- cfg
|-- openvswitch
|   |-- cfg
|   `-- tests
|       |-- cfg
|-- qemu
|   |-- cfg
|   `-- tests
|       |-- cfg
|-- shared
|   |-- autoit
|   |-- blkdebug
|   |-- cfg
|   |   |-- guest-os
|   |   |   |-- Linux
|   |   |   |   |-- Fedora
|   |   |   |   |-- JeOS
|   |   |   |   |-- LinuxCustom
|   |   |   |   |-- OpenSUSE
```

```
| | | |-- RHEL
| | | |-- SLES
| | | |-- Ubuntu
| | |-- Windows
| | | |-- Win2000
| | | |-- Win2003
| | | |-- Win2008
| | | |-- Win7
| | | |-- WindowsCustom
| | | |-- WinVista
| | | |-- WinXP
| |-- control
| |-- deps
| | |-- test_clock_getres
| | |-- test_cpu_flags
| |-- download.d
| |-- scripts
| |-- steps
| |-- unattended
|-- tests
| |-- cfg
|-- tools
|-- v2v
| |-- cfg
| |-- tests
| |-- cfg
|-- virttest
```

Talking about the top level directories:

Subtest dirs

Those directories hold specific test code for different virtualization types. Originally, virt-tests started as a set of tests for kvm, project that was known as virt-test. The request to support other virt backends, such as libvirt made us to generalize the tests infrastructure and support other backends. As of the time of this writing, we have 4 main backends, which we expect to grow:

1. qemu (used to be known as kvm)
2. libvirt
3. openvswitch (bridge technology testing)
4. v2v (testing of tools used to migrate vms from 1 technology to another)

Inside a subtest dir, the structure is, usually:

```
|-- qemu
| |-- cfg -> Config files that will be parsed by the test runner/autotest
| |-- tests -> Holds the tests specific to that backend
| |-- cfg -> Holds config snippets for the tests
```

The test runner, for example, will parse the top level config file `qemu/cfg/tests.cfg`. This file includes a number of other files, and will generate a large set of dictionaries, with all variations of a given set of parameters.

Not all virt tests require config snippets, but some might want to make use of the features of the [CartesianConfigReference | Cartesian files] and make one test source to generate several different tests. If you don't want to use that, no sweat, you're not obligated.

The snippets in tests/cfg will be used to generate subtests.cfg, a listing of all tests available for that particular backend.

Shared tests

Some tests in virt-tests are generic enough that they might run in more than one virt backend. For example, if one virt test uses guests, but does not use the qemu monitor interface (vm.monitor), it's likely that it belongs to the shared test dir (toplevel tests). The structure for it is simple:

```
|-- tests -> Tests that are shared by more than one virt backend
|   |-- cfg -> Holds config snippets for the tests
```

Shared resources dir

The virt tests often need a number of resources, be it a:

- Disk image
- Operating System CD
- Scripts and executables to run in the guest
- OEM installer files (kickstarts, windows answer files, among others)

We concentrate all those resources in the shared dir. If you look at its structure, you'll see:

```
|-- shared
|   |-- autoit -> Windows specific automation files
|   |-- blkdebug -> QEMU blkdebug config files
|   |-- cfg -> Holds base config files
|   |   |-- guest-os -> Holds a number of guest OS config snippets that'll create guest-os.cfg
|   |   |   |-- Linux
|   |   |   |   |-- Fedora
|   |   |   |   |-- JeOS
|   |   |   |   |-- LinuxCustom
|   |   |   |   |-- OpenSUSE
|   |   |   |   |-- RHEL
|   |   |   |   |-- SLES
|   |   |   |   |-- Ubuntu
|   |   |   |-- Windows
|   |   |       |-- Win2000
|   |   |       |-- Win2003
|   |   |       |-- Win2008
|   |   |       |-- Win7
|   |   |       |-- WindowsCustom
|   |   |       |-- WinVista
|   |   |       |-- WinXP
|   |-- control -> Holds autotest control files to run in the guest
|   |-- deps -> C programs that need to be compiled in the guest
|   |   |-- test_clock_getres
|   |   |-- test_cpu_flags
|   |-- download.d -> Holds resource files, that can be used to download disks
|   |-- scripts -> Holds python scripts to be executed in the guest
|   |-- steps -> Recordings of guest interaction that can be replayed
|   |-- unattended -> OEM install files (kickstarts, windows answer files)
```

Tools dir

The tools dir contains a bunch of useful tools for test writers and virt-test maintainers. Specially useful are the tools to run the unittests available for virt-test, and run_pylint.py, which runs pylint in any python file you might want, which helps to capture silly mistakes before they go public.

```
tools/
|-- cd_hash.py -> Calculates MD5 and SHA1 for ISOS (in fact, for any file)
|-- check_patch.py -> Verify whether a github or patchwork patch is OK
|-- common.py
|-- common.pyc
|-- download_manager.py -> Download resources, such as ISOS and guest images
|-- koji_pkgspec.py -> Get info about packages in Koji or Brew
|-- parallel.py
|-- parallel.pyc
|-- perf.conf
|-- regression.py -> Compare virt test jobs performance data
|-- reindent.py -> Fix indentation mistakes on your python files
|-- run_pylint.py -> Static source checker for python
|-- run_unittests.py -> Run all available virttest unittests
|-- tapfd_helper.py -> Paste a qemu cmd line produced by autotest and run it
`-- virt_disk.py -> Create floppy images and iso files
```

Virttest dir

In this dir, goes most of the library code of virt test. Over the years, the number of libraries grew quite a bit. Inside test code, those libraries are usually imported like:

```
from virttest import [library name]
```

Here's a listing with high level descriptions of each file:

```
virttest
|-- aexpect.py -> Controls subprocesses interactively
|-- base_installer.py -> Base code for virt software install
|-- bootstrap.py -> Functions to prepare environment previous to test exec
|-- build_helper.py -> Code with rules to build software
|-- cartesian_config.py -> The parser of the cartesian file format
|-- common.py
|-- data_dir.py -> Finds/sets the main data file
|-- ElementPath.py -> Library to manipulate XML
|-- ElementTree.py -> Library to manipulate XML
|-- env_process.py -> Handles setup/cleanup pre/post tests
|-- guest_agent.py -> Controls the qemu guest agent
|-- http_server.py -> Simple server for kickstart installs
|-- __init__.py
|-- installer.py -> Code for virt software install
|-- installer_unittest.py
|-- iscsi.py -> Code to handle vm images in iscsi disks
|-- iscsi_unittest.py
|-- libvirt_storage.py -> Create images for libvirt tests
|-- libvirt_vm.py -> VM class for libvirt backend
|-- libvirt_xml.py -> High level XML manipulation for libvirt test purposes
|-- libvirt_xml_unittest.py
|-- openvswitch.py -> Functions to deal with openvswitch network technology
|-- ovirt.py -> Library to handle an ovirt server
|-- ovs_utils.py -> Utils for the openvswitch test
```

```

|-- passfd.c -> Python c library for filedescriptor passing
|-- passfd.py -> Library for filedescriptor passing (python interface)
|-- passfd_setup.py -> Compiles the passfd library
|-- postprocess_iozone.py -> Code to analyze iozone results
|-- ppm_utils.py -> Code to handle QEMU screenshot file format
|-- propcan.py -> Class to handle sets of config values
|-- propcan_unittest.py
|-- qemu_installer.py -> Class to install qemu (git, rpm, etc)
|-- qemu_io.py -> Code to call qemu-io, for testing
|-- qemu_monitor.py -> Handles the qemu monitor interfaces (HMP and QMP)
|-- qemu_qtree.py -> Creates a data structure representation of qemu qtree output
|-- qemu_qtree_unittest.py
|-- qemu_storage.py -> Handles image creation for the qemu test
|-- qemu_virtio_port.py -> Code for dealing with qemu virtio ports
|-- qemu_vm.py -> VM class for the qemu test
|-- remote.py -> Functions to handle logins and remote transfers
|-- rss_client.py -> Client for the windows shell tool developed for virt-tests
|-- scheduler.py -> Functions for parallel testing
|-- standalone_test.py -> Implements a small test harness for execution independent of autotest
|-- step_editor.py -> Code for recording interaction with guests and replay them
|-- storage.py -> Base code for disk image creation
|-- syslog_server.py -> Simple syslog server to capture messages from OS installs
|-- test_setup.py -> Tests prep code (Hugepages setup, among others)
|-- utils_cgroup.py -> Utils to create and manipulate cgroups
|-- utils_disk.py -> Utils to create ISOS and floppy images
|-- utils_env.py -> Contains the class that holds the VM instances and other persistent info
|-- utils_env_unittest.py
|-- utils_koji.py -> Utils to interact with the Koji and Brew Buildsystems
|-- utils_misc.py -> Utils that don't fit in broader categories
|-- utils_misc_unittest.py
|-- utils_net.py -> VM and Host network utils
|-- utils_net_unittest.py
|-- utils_params.py -> Contains the class that holds test config data
|-- utils_spice.py -> Contains utils for spice testing
|-- utils_test.py -> Contains high level common utilities for testing
|-- utils_v2v.py -> Contains utilities for v2v testing
|-- versionable_class.py -> Classes with multiple ancestors, for openvswitch testing
|-- versionable_class_unittest.py
|-- video_maker.py -> Creates a ogg/webm video from vm screenshots
|-- virsh.py -> Calls and tests the virsh utility
|-- virsh_unittest.py
|-- virt_vm.py -> Base VM class, from where the specific tests derive from
|-- xml_utils.py -> Utils for XML manipulation
|-- xml_utils_unittest.py
`-- yumrepo.py -> Lib to create yum repositories, test helper

```

As you can see, there's quite a lot of code. We try to keep it as organized as possible, but if you have any problems just let us know (see [ContactInfo](#)).

Test Providers

Test providers are the conjunction of a loadable module mechanism inside virt-test that can pull a directory that will provide tests, config files and any dependencies, and those directories. The design goals behind test providers are:

- Make it possible for other organizations to maintain test repositories, in other arbitrary git repositories.
- Stabilize API and enforce separation of core virt-test functionality and tests.

The test provider spec is divided in Provider Layout and Definition files.

Test Provider Layout

```
.
|-- backend_1      -> Backend name. The actual name doesn't matter.
|   |-- cfg        -> Test config directory. Holds base files for the test runner.
|   |-- deps       -> Auxiliary files such as ELF files, Windows executables, images that tests need
|   |-- provider_lib -> Shared libraries among tests.
|   |-- tests      -> Python test files.
|   |-- cfg        -> Config files for tests.
`-- backend_2
    |-- cfg
    |-- deps
    |-- provider_lib
    |-- tests
    |-- cfg
```

In fact, virt-test libraries are smart enough to support arbitrary organization of python and config files inside the ‘tests’ directory. You don’t need to name the top level sub directories after backend names, although that certainly makes things easier. The term ‘backend’ is used to refer to the supported virtualization technologies by virt-test. As of this writing, the backends known by virt-test are:

- generic (tests that run in multiple backends)
- qemu
- openvswitch
- libvirt
- v2v
- libguestfs
- lvsb

The reason why you don’t need to name the directories after the backend names is that you can configure a test definition file to point out any dir name. We’ll get into

Types of Test Providers

Each test provider can be either a local filesystem directory, or a subdirectory of a git repository. Of course, the git repo subdirectory can be the repo root directory, but one of the points of the proposal is that people can hold virt-test providers inside git repos of other projects. Say qemu wants to maintain its own provider, they can do this by holding the tests, say, inside a tests/virt-test subdirectory inside qemu.git.

Test Provider definition file

The main virt-test suite needs a way to know about test providers. It does that by scanning definition files inside the ‘test-providers.d’ sub directory. Definition files are *config parser files* <<http://docs.python.org/2/library/configparser.html>> that encode information from a test provider. Here’s an example structure of a test provider file:

```
[provider]

# Test provider URI (default is a git repository, fallback to standard dir)
```

```

uri: git://git-provider.com/repo.git
#uri: /path-to-my-git-dir/repo.git
#uri: http://bla.com/repo.git
#uri: file:///usr/share/tests

# Optional git branch (for git repo type)
branch: master

# Optional git commit reference (tag or sha1)
ref: e44231e88300131621586d24c07baa8e627de989

# Pubkey: File containing public key for signed tags (git)
pubkey: example.pub

# What follows is a sequence of sections for any backends that this test
# provider implements tests for. You must specify the sub directories of
# each backend dir, reason why the subdir names can be arbitrary.

[qemu]
# Optional subdir (place inside repo where the actual tests are)
# This is useful for projects to keep virt tests inside their
# (larger) test repos. Defaults to ''.
subdir: src/tests/qemu/

[agnostic]
# For each test backend, you may have different sub directories
subdir: src/tests/generic/

```

Example of a default virt-test provider file:

```

[provider]
uri: https://github.com/autotest/tp-qemu.git
[generic]
subdir: generic/
[qemu]
subdir: qemu/
[openvswitch]
subdir: openvswitch/

```

Let's say you want to use a directory in your file system (/usr/share/tests/virt-test):

```

[provider]
uri: file:///usr/share/tests/
[generic]
subdir: virt-test/generic/
[qemu]
subdir: virt-test/qemu/
[openvswitch]
subdir: virt-test/openvswitch/

```

Any doubts about the specification, let me know - Email lmr AT redhat DOT com.

Development workflow after the Repository Split

1. Clone virt-test
2. Fork the test provider you want to contribute to in github <<https://help.github.com/articles/fork-a-repo>>
3. Clone the forked repository. In this example, we'll assume you cloned the forked repo to

```
/home/user/code/tp-libvirt
```

4. Add a file in `virt-test/test-providers.d`, with a name you like. We'll assume you chose

```
user-libvirt.ini
```

5. Contents of `user-libvirt.ini`:

```
[provider]
uri: file:///home/user/code/tp-qemu
[libvirt]
subdir: libvirt/
[libguestfs]
subdir: libguestfs/
[lvsb]
subdir: lvsb/
[v2v]
subdir: v2v/
```

6. This should be enough. Now, when you use `-list-tests`, you'll be able to see entries like:

```
...
1 user-libvirt.unattended_install.cdrom.extra_cdrom_ks.default_install.aio_native
2 user-libvirt.unattended_install.cdrom.extra_cdrom_ks.default_install.aio_threads
3 user-libvirt.unattended_install.cdrom.extra_cdrom_ks.perf.aio_native
...
```

7. Modify tests, or add new ones to your heart's content. When you're happy with your changes, you may create branches and [send us pull requests](#).

That should be it. Let us know if you have any doubts about the process through [the mailing list](#) or [opening an issue](#).

Writing your own virt test

In this article, we'll talk about:

1. Where the test files are located
2. Write a simple test file
3. Try out your new test, send it to the mailing list

Where the virt test files are located ?

The subtests can be located in 2 subdirs:

- tests - These tests are written in a fairly virt technology agnostic way, so they can be used by other virt technologies testing. More specifically, they do not use the vm monitor.

```
$ ls
autotest_control.py  fail.py          iofuzz.py        module_probe.py  ping.py
boot.py             file_transfer.py ioquit.py        multicast.py      pxe.py
boot_savevm.py      fillup_disk.py  iozone_windows.py netperf.py        rv_connect
build.py            fullscreen_setup.py jumbo.py        netstress_kill_guest.py rv_copyand
cfg                guest_s4.py      kdump.py        nfs_corrupt.py   rv_disconn
clock_getres.py     guest_test.py   linux_s3.py     nicdriver_unload.py rv_fullscr
dd_test.py          image_copy.py   lvm.py          nic_promisc.py   rv_input.p
ethtool.py          __init__.py     mac_change.py   ntttcp.py        save_resto
```

- `qemu/tests` - These are tests that do use specific qemu infrastructure, specifically the qemu monitor. other virt technologies can't use it so they go here.

```
$ ls
9p.py          __init__.py      multi_disk.py
balloon_check.py kernel_install.py negative_create.py
block_mirror.py ksm_overcommit.py nic_bonding.py
block_stream.py migration_multi_host_cancel.py nic_hotplug.py
cdrom.py        migration_multi_host_downtime_and_speed.py nmi_bsod_catch.py
cfg             migration_multi_host_ping_pong.py nmi_watchdog.py
cgroup.py       migration_multi_host.py pci_hotplug.py
cpuflags.py     migration_multi_host_with_file_transfer.py perf_qemu.py
cpu_hotplug.py  migration_multi_host_with_speed_measurement.py performance.py
enospc.py       migration.py      physical_resources_check.py
floppy.py       migration_with_file_transfer.py qemu_guest_agent.py
getfd.py        migration_with_reboot.py qemu_guest_agent_snapshot.py
hdparm.py       migration_with_speed_measurement.py qemu_img.py
```

So the thumb rule is, if it uses the qemu monitor, you stick it into `qemu/tests`, if it doesn't, you can stick it into the `tests/` dir.

Write our own, drop-in 'uptime' test - Step by Step procedure

Now, let's go and write our uptime test, which only purpose in life is to pick up a living guest, connect to it via ssh, and return its uptime.

1. Git clone `virt_test.git` to a convenient location, say `$HOME/Code/virt-test`. See *the download source documentation* <./contributing/DownloadSource>. Please do use git and clone the repo to the location mentioned.
2. Our uptime test won't need any qemu specific feature. Thinking about it, we only need a vm object and establish an ssh session to it, so we can run the command. So we can store our brand new test under `tests`. At the autotest root location:

```
[lmr@freedom virt-test.git]$ touch tests/uptime.py
[lmr@freedom virt-test.git]$ git add tests/uptime.py
```

3. Ok, so that's a start. So, we have *at least* to implement a function `run_uptime`. Let's start with it and just put the keyword `pass`, which is a no op. Our test will be like:

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    pass
```

4. Now, what is the API we need to grab a VM from our test environment? Our `env` object has a method, `get_vm`, that will pick up a given vm name stored in our environment. Some of them have aliases. `main_vm` contains the name of the main vm present in the environment, which is, most of the time, `vm1`. `env.get_vm` returns a vm object, which we'll store on the variable `vm`. It'll be like this:

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
```

5. A vm object has lots of interesting methods, which we plan on documenting them more thoroughly, but for now, we want to ensure that this VM is alive and functional, at least from a qemu process standpoint. So, we'll call

the method `verify_alive()`, which will verify whether the `qemu` process is functional and if the monitors, if any exist, are functional. If any of these conditions are not satisfied due to any problem, an exception will be thrown and the test will fail. This requirement is because sometimes due to a bug the `vm` process might be dead on the water, or the monitors are not responding.

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
```

6. Next step, we want to log into the `vm`. the `vm` method that does return a remote session object is called `wait_for_login()`, and as one of the parameters, it allows you to adjust the timeout, that is, the time we want to wait to see if we can grab an `ssh` prompt. We have top level variable `login_timeout`, and it is a good practice to retrieve it and pass its value to `wait_for_login()`, so if for some reason we're running on a slower host, the increase in one variable will affect all tests. Note that it is completely OK to just override this value, or pass nothing to `wait_for_login()`, since this method does have a default timeout value. Back to business, picking up `login` timeout from our dict of parameters:

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
    timeout = float(params.get("login_timeout", 240))
```

7. Now we'll call `wait_for_login()` and pass the timeout to it, storing the resulting session object on a variable named `session`.

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
    timeout = float(params.get("login_timeout", 240))
    session = vm.wait_for_login(timeout=timeout)
```

8. The `qemu` test will do its best to grab this session, if it can't due to a timeout or other reason it'll throw a failure, failing the test. Assuming that things went well, now you have a session object, that allows you to type in commands on your guest and retrieve the outputs. So most of the time, we can get the output of these commands through the method `cmd()`. It will type in the command, grab the `stdin` and `stdout`, return them so you can store it in a variable, and if the exit code of the command is `!= 0`, it'll throw a `aexpect.ShellError`?. So getting the output of the unix command `uptime` is as simple as calling `cmd()` with 'uptime' as a parameter and storing the result in a variable called `uptime`:

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
    timeout = float(params.get("login_timeout", 240))
    session = vm.wait_for_login(timeout=timeout)
    uptime = session.cmd('uptime')
```

9. If you want to just print this value so it can be seen on the test logs, just log the value of `uptime` using the

logging library. Since that is all we want to do, we may close the remote connection, to avoid ssh/rss sessions lying around your test machine, with the method `close()`. Now, note that all failures that might happen here are implicitly handled by the methods called. If a test went from its beginning to its end without unhandled exceptions, autotest assumes the test automatically as PASSED, *no need to mark a test as explicitly passed*. If you have explicit points of failure, for more complex tests, you might want to add some exception raising.

```
def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
    timeout = float(params.get("login_timeout", 240))
    session = vm.wait_for_login(timeout=timeout)
    uptime = session.cmd("uptime")
    logging.info("Guest uptime result is: %s", uptime)
    session.close()
```

10. Now, I deliberately introduced a bug on this code just to show you guys how to use some tools to find and remove trivial bugs on your code. I strongly encourage you guys to check your code with the script called `run_pylint.py`, located at the `utils` directory at the top of your `$AUTOTEST_ROOT`. This tool calls internally the other python tool called `pylint` to catch bugs on autotest code. I use it so much the `utils` dir of my devel autotest tree is on my `$PATH`. So, to check our new uptime code, we can call (important, if you don't have `pylint` install it with `yum install pylint` or equivalent for your distro):

```
[lmr@freedom virt-test.git]$ tools/run_pylint.py tests/uptime.py -q
***** Module virt-test.git.tests.uptime
E0602: 10,4:run_uptime: Undefined variable 'logging'
```

11. Ouch. So there's this undefined variable called `logging` on line 10 of the code. It's because I forgot to import the logging library, which is a python library to handle info, debug, warning messages. Let's Fix it and the code becomes:

```
import logging

def run_uptime(test, params, env):
    """
    Docstring describing uptime.
    """
    vm = env.get_vm(params["main_vm"])
    vm.verify_alive()
    timeout = float(params.get("login_timeout", 240))
    session = vm.wait_for_login(timeout=timeout)
    uptime = session.cmd("uptime")
    logging.info("Guest uptime result is: %s", uptime)
    session.close()
```

12. Let's re-run `run_pylint.py` to see if it's happy with the code generated:

```
[lmr@freedom virt-test.git]$ tools/run_pylint.py tests/uptime.py -q
[lmr@freedom virt-test.git]$
```

13. So we're good. Nice! Now, as good indentation does matter to python, we have another small utility called `reindent.py`, that will fix indentation problems, and cut trailing whitespaces on your code. Very nice for tidying up your test before submission.

```
[lmr@freedom virt-test.git]$ tools/reindent.py tests/uptime.py
```

14. I also use `run_pylint` with no `-q` catch small things such as wrong spacing around operators and other subtle

issues that go against PEP 8 and the [coding style document](#). Please take pylint's output with a *handful* of salt, you don't need to work each and every issue that pylint finds, I use it to find unused imports and other minor things.

```
[lmr@freedom virt-test.git]$ tools/run_pylint.py tests/uptime.py
***** Module virt-test.git.tests.uptime
C0111: 1,0: Missing docstring
C0103: 7,4:run_uptime: Invalid name "vm" (should match [a-z_][a-z0-9_]{2,30}$)
W0613: 3,15:run_uptime: Unused argument 'test'
```

15. These other complaints you don't really need to fix. Due to the tests design, they all use 3 arguments, 'vm' is a shorthand that we have been using for a long time as a variable name to hold a VM object, and the only docstring we'd like you to fill is the one in the run_uptime function.

16. Now, you can test your code. When listing the qemu tests your new test should appear in the list:

```
./run -t qemu --list-tests
```

17. Now, you can run your test to see if everything went good.

```
[lmr@freedom virt-test.git]$ ./run -t qemu --tests uptime
SETUP: PASS (1.10 s)
DATA DIR: /home/lmr/virt_test
DEBUG LOG: /home/lmr/Code/virt-test.git/logs/run-2012-11-28-13.13.29/debug.log
TESTS: 1
(1/1) uptime: PASS (23.30 s)
```

18. Ok, so now, we have something that can be git committed and sent to the mailing list

```
diff --git a/tests/uptime.py b/tests/uptime.py
index e69de29..65d46fa 100644
--- a/tests/uptime.py
+++ b/tests/uptime.py
@@ -0,0 +1,13 @@
+import logging
+
+def run_uptime(test, params, env):
+    """
+    Docstring describing uptime.
+    """
+    vm = env.get_vm(params["main_vm"])
+    vm.verify_alive()
+    timeout = float(params.get("login_timeout", 240))
+    session = vm.wait_for_login(timeout=timeout)
+    uptime = session.cmd("uptime")
+    logging.info("Guest uptime result is: %s", uptime)
+    session.close()
```

19. Oh, we forgot to add a decent docstring description. So doing it:

```
import logging

def run_uptime(test, params, env):

    """
    Uptime test for virt guests:

    1) Boot up a VM.
    2) Stablish a remote connection to it.
    3) Run the 'uptime' command and log its results.
```

```

:param test: QEMU test object.
:param params: Dictionary with the test parameters.
:param env: Dictionary with test environment.
"""

vm = env.get_vm(params["main_vm"])
vm.verify_alive()
timeout = float(params.get("login_timeout", 240))
session = vm.wait_for_login(timeout=timeout)
uptime = session.cmd("uptime")
logging.info("Guest uptime result is: %s", uptime)
session.close()

```

20. git commit signing it, put a proper description, then send it with git send-email. Profit!

Defining New Guests

Let's say you have a guest image that you've carefully prepared, and the JeOS just doesn't cut it. Here's how you add new guests:

Linux Based Custom Guest

If your guest is Linux based, you can add a config file snippet describing your test (We have a bunch of pre-set values for linux in the default config).

The drop in directory is

```
shared/cfg/guest-os/Linux/LinuxCustom
```

You can add, say, foo.cfg to that dir with the content:

```

FooLinux:
    image_name = images/foo-linux

```

Which would make it possible to specify this custom guest using

```
./run -t qemu -g LinuxCustom.FooLinux
```

Provided that you have a file called images/foo-linux.qcow2, if using the qcow2 format image. If you wish to provide a raw image file, you must use

```
./run -t qemu -g LinuxCustom.FooLinux --image-type raw
```

Other useful params to set (not an exhaustive list):

```

# shell_prompt is a regexp used to match the prompt on aexpect.
# if your custom os is based of some distro listed in the guest-os
# dir, you can look on the files and just copy shell_prompt
shell_prompt = [*]$
# If you plan to use a raw device, set image_device = yes
image_raw_device = yes
# Password of your image
password = 123456
# Shell client used (may be telnet or ssh)
shell_client = ssh
# Port were the shell client is running
shell_port = 22

```

```
# File transfer client
file_transfer_client = scp
# File transfer port
file_transfer_port = 22
```

Windows Based Custom Guest

If your guest is Linux based, you can add a config file snippet describing your test (We have a bunch of pre-set values for linux in the default config).

The drop in directory is

```
shared/cfg/guest-os/Windows/WindowsCustom
```

You can add, say, foo.cfg to that dir with the content:

```
FooWindows:
    image_name = images/foo-windows
```

Which would make it possible to specify this custom guest using

```
./run -t qemu -g WindowsCustom.FooWindows
```

Provided that you have a file called images/foo-windows.qcow2, if using the qcow2 format image. If you wish to provide a raw image file, you must use

```
./run -t qemu -g WindowsCustom.FooWindows --image-type raw
```

Other useful params to set (not an exhaustive list):

```
# If you plan to use a raw device, set image_device = yes
image_raw_device = yes
# Attention: Changing the password in this file is not supported,
# since files in winutils.iso use it.
username = Administrator
password = 1q2w3eP
```

Writing a more advanced test

Now that you wrote your first simple test, we'll try some more involved examples. First, let's talk about some useful APIs and concepts:

As virt-tests evolved, a number of libraries were written to help test writers. Let's see what some of them can do:

1. `virttest.data_dir` -> Has functions to get paths for resource files. One of the most used functions is `data_dir.get_data_dir()`, that returns the path `shared/data`, which helps you to get files.

```
from virttest import data_dir
```

What's available upfront

Very frequently we may get values from the config set. All virt tests take 3 params:

test -> Test object params -> Dict with current test params
env -> Environment file being used for the test
job

You might pick any parameter using

```
variable_name = params.get("param_name", default_value)
```

You can update the parameters using

Advanced docs

Dive into fully detailed descriptions of virt test functionality, such as the cartesian config file structure.

Contents:

Contents

- *Virt Test Primer*
 - *Autotest*
 - * *Introduction*
 - * *Server*
 - * *Client*
 - * *Virtualization Test*
 - *Virtualization Tests*
 - *Introduction*
 - * *Quickstart*
 - *Pre-requisites*
 - *Clone*
 - `./run -t <type> --bootstrap`
 - *Run default tests*
 - *Running different tests*
 - *Checking the results*
 - * *Utilities*
 - *Detailed Test Execution*
 - * *Autotest Command Line*
 - *Output*
 - *Verbosity*
 - *Job Names and Tags*
 - *Autotest client sub-commands*
 - `help`
 - `list`
 - `run`
 - *Results*
 - * *Virt-test runner*
 - *Virt-test runner output*
 - *Virt-test runner results*
 - * *File/Directory Layout*
 - *Overview*
 - *Virt-test Details*
 - * *Cartesian Configuration*
 - *Keys and values*
 - *Variants*
 - *Named variants*
 - *Dependencies*
 - *Filters*
 - *Value Substitutions*
 - *Key sub-arrays*
 - *Include statements*
 - *Combinatorial outcome*
 - *Formal definition*
 - *Examples*
 - * *Default Configuration Files*
 - *Configuration file details*
 - *Base*
 - *tests*
 - *cdkeys and windows virtio*
 - *guest hw & guest os*
 - *Sub-tests*
 - *Configuration usage details*
 - *Preserving installed Guest images*
 - *Specialized Networking*
 - *Using virtio drivers with windows*

Virt Test Primer

Autotest

Introduction

It is critical for any project to maintain a high level of software quality, and consistent interfaces to other software that it uses or uses it. Autotest is a framework for fully automated testing, that is designed primarily to test the Linux kernel, though is useful for many other functions too. It includes a client component for executing tests and gathering results, and a completely optional server component for managing a grid of client systems.

Server

Job data, client information, and results are stored in a MySQL database, either locally or on a remote system. The Autotest server manages each client and its test jobs with individual “autoserv” processes (one per client). A dispatcher process “monitor_db”, starts the autoserv processes to service requests based on database content. Finally, both command-line and a web-based graphical interface is available.

Client

The Autotest client can run either standalone or within a server harness. It is not tightly coupled with the Autotest server, though they are designed to work together. Primary design drivers include handling errors implicitly, producing consistent results, ease of installation, and maintenance simplicity.

Virtualization Test

The virtualization tests are sub-modules of the Autotest client that utilize its modular framework. The entire suite of top-level autotest tests are also available within virtualized guests. In addition, many specific sub-tests are provided within the virtualization sub-test framework. Some of the sub-tests are shared across virtualization technologies, while others are specific.

Control over the virtualization sub-tests is provided by the test-runner (script) and/or a collection of configuration files. The configuration file format is highly specialized (see section [cartesian_configuration](#)). However, by using the test-runner, little (if any) knowledge of the configuration file format is required. Utilizing the test-runner is the preferred method for individuals and developers to execute stand-alone virtualization testing.

Virtualization Tests

Introduction

The virt-test suite helps exercise virtualization features with help from qemu, libvirt, and other related tools and facilities. However, due to its scope and complexity, this aspect of Autotest has been separated into the dedicated ‘virt-test’ suite. This suite includes multiple packages dedicated to specific aspects of virtualization testing.

Within each virt-test package, are a collection of independent sub-test modules. These may be addressed individually or as part of a sequence. In order to hide much of the complexity involved in virtualization testing and development, a dedicated test-runner is included with the virt-test suite (see section [test_runner](#)).

Quickstart

Pre-requisites

1. A supported host platforms: Red Hat Enterprise Linux (RHEL) or Fedora. OpenSUSE should also work, but currently autotest is still not packaged for it, which means you have to clone autotest and put its path in an env variable so virt tests can find the autotest libs. Debian/Ubuntu now have a new experimental package that allows one to run the virt tests in a fairly straight forward way.
2. [Install software packages \(RHEL/Fedora\)](#)
3. [Install software packages \(Debian/Ubuntu\)](#)
4. A copy of the [virt test source](#)

Clone

1. Clone the virt test repo

```
git clone git://github.com/autotest/virt-test.git
```

1. Change into the repository directory

```
cd virt-test
```

`./run -t <type> --bootstrap` Where `<type>` is the virtualization test type you want to setup, for example "qemu". Explicitly using `--bootstrap` causes setup to run interactively and is highly recommended. Otherwise, the test runner will execute the same operations non-interactively. Running it interactively allows for choice and modification of to the environment to suit specific testing or setup needs.

The setup process includes checks for the minimum host software requirements and sets up a directory tree to hold data. It also downloads a minimal guest OS image (about 180 MB) called JeOS (based on Fedora). This is the default guest used when a full-blown build from an automated install is not required.

When executed as a non-root user, `./run -t <type> --bootstrap` will create and use `$HOME/virt_test` as the data directory to hold OS images, logs, temporary files, etc. Whereas for root, the system-wide location `/var/lib/virt-test` will be used. However it is invoked, as user, root, interactive, or not, a symbolic link to the data directory will be created `virt-test/shared/data` (i.e. under the directory the repository was cloned in).

Interactive `--bootstrap` may be run at any time, for example to re-generate the default configuration after pulling down a new release. Note that the `-t <type>` argument is crucial. Any subdirectory of `virt-test` which contains a file named `control` is a candidate `<type>`. Also, each `<type>` has different requirements. For example, the libguestfs tests have different software requirements than the qemu tests.

Run default tests For qemu and libvirt subtests, the default test set does not require root. However, other tests might fail due to lack of privileges.

```
./run -t qemu
```

or

```
./run -t libvirt
```

Running different tests You can list the available tests with the `-list-tests` parameter.

```
$ ./run -t qemu --list-tests
(will print a numbered list of tests, with a pagination)
```


Then, pass test *names* as a quote-protected, space-separated list to the `--tests` parameter. For example:

1. For qemu testing:

```
$ ./run -t qemu --tests "migrate time-drift file_transfer"
```

2. Many libvirt tests require the `virt-test-vm1` guest exists, and assume it is removed or restored to pristine state at the end. However, when running a custom set of tests this may not be the case. In this case, you may need to use the `--install` and/or `--remove` options to the test runner. For example:

```
# ./run -t libvirt --install --remove --tests "reboot"
```

Checking the results The test runner will produce a debug log, that will be useful to debug problems:

```
[lmr@localhost virt-test.git]$ ./run -t qemu --tests boot_with_usb
SETUP: PASS (1.20 s)
DATA DIR: /path/to/virt_test
DEBUG LOG: /path/to/virt-test.git/logs/run-2012-12-12-01.39.34/debug.log
TESTS: 10
boot_with_usb.ehci: PASS (18.34 s)
boot_with_usb.keyboard.uhci: PASS (21.57 s)
boot_with_usb.keyboard.xhci: PASS (24.56 s)
boot_with_usb.mouse.uhci: PASS (21.59 s)
boot_with_usb.mouse.xhci: PASS (23.11 s)
boot_with_usb.usb_audio: PASS (20.99 s)
boot_with_usb.hub: PASS (22.12 s)
boot_with_usb.storage.uhci: PASS (21.61 s)
boot_with_usb.storage.ehci: PASS (23.27 s)
boot_with_usb.storage.xhci: PASS (25.03 s)
```

For convenience, the most recent debug log is pointed to by the `logs/latest/debug.log` symlink.

Utilities

A number of helpful command-line utilities are provided along with the Autotest client. Depending on the installation, they could be located in various places. The table below outlines some of them along with a brief description.

Name	Description
<code>autotest-local</code>	The autotest command-line client.
<code>cartesian_config.py</code>	Test matrix configuration parser module and command-line display utility.
<code>scan_results.py</code>	Check for and pretty-print current testing status and/or results.
<code>html_report.py</code>	Command-line HTML index and test result presentation utility.
<code>run</code>	Test runner for virt-test suite.

For developers, there are a separate set of utilities to help with writing, debugging, and checking code and/or tests. Please see section [development_tools](#) for more detail.

Detailed Test Execution

Tests are executed from a copy of the Autotest client code, typically on separate hardware from the Autotest server (if there is one). Executing tests directly from a clone of the git repositories or installed Autotest is possible. The tree is configured such that test results and local configuration changes are kept separate from test and Autotest code.

For virtualization tests, variant selection(s) and configuration(s) is required either manually through specification in `tests.cfg` (see section [tests_cfg](#)) or automatically by using the test-runner (see section [run_different_tests](#)). The test-

runner is nearly trivial to use, but doesn't offer the entire extent of test customization. See the `virt_test_runner` section for more information.

Autotest Command Line

Several Autotest-client command-line options and parameters are available. Running the 'autotest' command with the '-h' or '--help' parameters will display the online help. The only required parameters are a path to the autotest control file which is detailed elsewhere in the autotest documentation.

Output Options for controlling client output are the most frequently used. The client process can "in a terminal, or placed in the background. Synchronous output via stdout/stderr is provided, however full-verbosity logs and test results are maintained separate from the controlling terminal. This allows users to respond to test output immediately, and/or an automated framework (such as the autotest server) to collect it later.

Verbosity Access to the highest possible detail level is provided when the '--verbose' option is used. There are multiple logging/message levels used within autotest, from DEBUG, to INFO, and ERROR. While all levels are logged individually, only INFO and above are displayed from the autotest command by default. Since DEBUG is one level lower than INFO, there are no provisions provided more granularity in terminal output.

Job Names and Tags The '-t', or '--tag' parameter is used to specify the TAG name that will be appended to the name of every test. JOBNAMES come from the autotest server, and scheduler for a particular client. When running the autotest client stand-alone from the command line, it's not possible to set the JOBNAMES. However, TAGs are a way of differentiating one test execution from another within a JOB. For example, if the same test is run multiple times with slight variations in parameters. TAGS are also a mechanism available on the stand-alone command line to differentiate between executions.

Autotest client sub-commands Sub-commands are a shortcut method for performing various client tasks. They are evaluated separately from the main command-line options. To use them, simply append them after any standard parameters on the client command line.

help The `help` sub-command prints out all sub-commands along with a short description of their use/purpose. This help output is in addition to the standard client command-line help output.

list The `list` sub-command searches for and displays a list of test names that contain a valid control file. The list includes a short description of each test and is sent to the default pager (i.e. more or less) for viewing.

run The `run` sub-command complements `list`, but as a shortcut for executing individual tests. Only the name of the test sub-directory is needed. For example, to execute `sleeptest`, the `bin/autotest-local run sleeptest` command may be used.

Results On the client machine, results are stored in a 'results' sub-directory, under the autotest client directory (AUTODIR). Within the 'results' sub-directory, data is grouped based on the autotest server-supplied job-name (JOBNAME). Variant shortnames (see section *variants*) represent the <TESTNAME> value used when results are recorded. When running a stand-alone client, or if unspecified, JOBNAMES is 'default'.

Relative Directory or File		Description
<AUTODIR>/results/JOBNAME/		Base directory for JOBNAME('default')
	sysinfo	Overall OS-level data from client system
	control	Copy of control file used to execute job
	status	Overall results table for each TAGged test
	sysinfo/	Test-centric OS-level data
	debug/	Client execution logs, See section <i>verbosity</i>
	Client.DEBUG, client.INFO, client.WARNING, client.ERROR	Client output at each verbosity level. Good
<TESTNAME><TAG>/		Base directory of results from a specific test
	status	Test start/end time and status report table
	keyval	Key / value parameters for test
	results/	Customized and/or nested-test results
	profiling/	Data from profiling tools during testing
	debug/	Client test output at each verbosity level
	build<TAG>/	Base directory for tests that build code
	status	Overall build status
	src/	Source code used in a build
	build/	Compile output / build scratch directory
	patches/	Patches to apply to source code
	config/	Config. Used during & for build
	debug/	Build output and logs
	summary	Info. About build test/progress.

Virt-test runner

Within the root of the virt-test sub-directory (autotest/client/tests/virt/, virt-test, or wherever you cloned the repository) is `run`. This is an executable python script which provides a single, simplified interface for running tests. The list of available options and arguments is provided by the `-h` or `--help`.

This interface also provides for initial and subsequent, interactive setup of the various virtualization sub-test types. Even if not, the setup will still be executed non-interactively before testing begins. See the section *run_bootstrap* for more information on initial setup.

To summarize it's use, execute `./run` with the subtest type as an argument to `-t` (e.g. `qemu`, `libvirt`, etc.), guest operating system with `-g` (e.g. `RHEL.6.5.x86_64`), and a quoted, space-separated list of test names with `--tests`. Everything except `-t <type>` is optional.

Virt-test runner output Assuming the `-v` verbose option is not used, the test runner will produce simple, colorized pass/fail output. Some basic statistics are provided at the end of all tests, such as pass/fail count, and total testing time. Full debug output is available by specifying the `-v` option, or by observing `logs/latest/debug.log`

Virt-test runner results When utilizing the test runner, results are logged slightly different from the autotest client. Each run logs output and results to a date & time stamped sub-directory beneath the `logs/` directory. For convenience, there is a `latest` symbolic link which always points at the previous run sub-directory. This makes it handy for tailing a currently running test in another terminal.

Relative Directory or File		Description
logs/run-YYYY-MM-DD-HH.MM.SS/		Results for a single run.
	debug.log	Debug-level output for entire run.
test.cartesian.short.name/		Results from individual test in run
	debug.log	Debug-level output from individual test
	keyval	Key / value parameters for test
	session-VM_NAME.log	Remote ssh session log to VM_NAME guest.
	VM_NAME-0.webm	5-second screenshot video of VM_NAME guest
	results/	Customized and/or nested-test results
	profiling/	Data from profiling tools (if configured)

File/Directory Layout

Overview The autotest source tree is organized in a nested structure from server, to client, to tests. The final tests element is further divided between all the independant autotest tests, and the virt test suite. This layout is intended to support easy customization at the lowest levels, while keeping the framework, tests, and configurations separated from eachother.

Traditionally, each of these elements would be nested within eachother like so:

Relative directory		Description
autotest/		Autotest server
	client/	Autotest client
	tests/	Test sub-directories
	virt/	virt-test subdirectories

However, for development and simple testing purposes, none of the server components is required, and nearly all activity will occur under the client and tests sub-directories. Further, depending on your operating environment, the client components may be available as the “autotest-framework” package. When installed, work may be solely concentrated within or beneath the `tests` sub-directory. For exclusive virtualization testing, only the `virt` sub-directory of the `tests` directory is required.

Virt-test Details Traditionally the virtualization tests directory tree would be rooted at `autotest/client/tests/virt`. However, when utilizing the `autotest-framework` package, it commonly resides under a `virt-test` directory, which may be located anywhere convenient (including your home directory).

Relative directory	Description
run.py	The test-runner script. (see section test_runner)
virt.py	Module used by the autotest framework to define the <code>test.test</code> subclass and methods needed for test execution. This is utilized when tests are executed from the autotest client.
logs/	Logs and test results when utilizing the test runner (see section test_runner_results)
virttest/	Modules for host, guest, and test utilities shared by nearly all the virt-test sub-test. The scope spans multiple virtualization hypervisors, technologies, libraries and tracking facilities. Not every component is required for every test, but all virtualization tests consume multiple modules within this tree.
common.py	Central autotest framework module utilized by nearly all other modules. It creates the top-level namespaces under which the entirety of the autotest client framework packages are made available as <code>autotest.client</code>
data_dir.py	Provides a centralized interface for virt-test code and tests to access runtime test data (os images, iso images, boot files, etc.)
standalone_test.py	Stand-in for the autotest-framework needed by the test runner. Takes the place of the <code>test.test</code> class. Also provides other test-runner specific classes and functions.
tests/	Shared virtualization sub-test modules. The largest and most complex is the unattended install. All test modules in this directory are virtualization technology agnostic. Most of the test modules are simple and well commented. They are an excellent reference for test developers starting to write a new test.
qemu, libvirt, libguestfs, etc.	Technology-specific trees organizing both test-modules and configuration.
cfg	Runtime virt test framework and test Cartesian configuration produced by <code>./run --bootstrap</code> and consumed by both the autotest-client and standalone test-runner. (See section default_configuration_files)
shared/	Runtime data shared among all virtualization tests.
cfg/	Persistent Cartesian configuration source for deriving technology-specific runtime configuration and definition (See section default_configuration_files)
unattended/	Data specific to the unattended install test. Kickstart, answer-files, as well as other data utilized during the unattended install process. Most of the files contain placeholder keywords which are substituted with actual values at run-time
control/	Autotest test control files used when executing autotest tests within a guest virtual machine.
data/	A symlink to dynamic runtime data shared among all virtualization tests. The destination and control over this location is managed by the <code>virttest/data_dir.py</code> module referenced above.
boot/	Files required for starting a virtual machine (i.e. kernel and initrd images)
images/	Virtual machine disk images and related files
isos/	Location for installation disc images

Cartesian Configuration

Cartesian Configuration is a highly specialized way of providing lists of key/value pairs within combinations of various categories. The format simplifies and condenses highly complex multidimensional arrays of test parameters into a flat list. The combinatorial result can be filtered and adjusted prior to testing, with filters, dependencies, and key/value substitutions.

The parser relies on indentation, and is very sensitive to misplacement of tab and space characters. It's highly recommended to edit/view Cartesian configuration files in an editor capable of collapsing tab characters into four space characters. Improper attention to column spacing can drastically affect output.

Keys and values Keys and values are the most basic useful facility provided by the format. A statement in the form `<key> = <value>` sets `<key>` to `<value>`. Values are strings, terminated by a linefeed, with surrounding quotes completely optional (but honored). A reference of descriptions for most keys is included in section Configuration

Parameter Reference. The key will become part of all lower-level (i.e. further indented) variant stanzas (see section [variants](#)). However, key precedence is evaluated in top-down or ‘last defined’ order. In other words, the last parsed key has precedence over earlier definitions.

Variants A ‘variants’ stanza is opened by a ‘variants:’ statement. The contents of the stanza must be indented further left than the ‘variants:’ statement. Each variant stanza or block defines a single dimension of the output array. When a Cartesian configuration file contains two variants stanzas, the output will be all possible combination’s of both variant contents. Variants may be nested within other variants, effectively nesting arbitrarily complex arrays within the cells of outside arrays. For example:

```
variants:
  - one:
      key1 = Hello
  - two:
      key2 = World
  - three:
variants:
  - four:
      key3 = foo
  - five:
      key3 = bar
  - six:
      key1 = foo
      key2 = bar
```

While combining, the parser forms names for each outcome based on prepending each variant onto a list. In other words, the first variant name parsed will appear as the left most name component. These names can become quite long, and since they contain keys to distinguishing between results, a ‘short-name’ key is also used. For example, running `cartesian_config.py` against the content above produces the following combinations and names:

```
dict 1: four.one
dict 2: four.two
dict 3: four.three
dict 4: five.one
dict 5: five.two
dict 6: five.three
dict 7: six.one
dict 8: six.two
dict 9: six.three
```

Variant shortnames represent the <TESTNAME> value used when results are recorded (see section Job Names and Tags. For convenience variants who’s name begins with a ‘@’ do not prepend their name to ‘short-name’, only ‘name’. This allows creating ‘shortcuts’ for specifying multiple sets or changes to key/value pairs without changing the results directory name. For example, this is often convenient for providing a collection of related pre-configured tests based on a combination of others (see section [tests](#)).

Named variants Named variants allow assigning a parseable name to a variant set. This enables an entire variant set to be used for in [filters](#). All output combinations will inherit the named variant key, along with the specific variant name. For example:

```
variants var1_name:
  - one:
      key1 = Hello
  - two:
      key2 = World
  - three:
```

```
variants var2_name:
    - one:
        key3 = Hello2
    - two:
        key4 = World2
    - three:

only (var2_name=one).(var1_name=two)
```

Results in the following outcome when parsed with `cartesian_config.py -c:`

```
dict    1:  (var2_name=one).(var1_name=two)
    dep = []
    key2 = World          # variable key2 from variants var1_name and variant two.
    key3 = Hello2         # variable key3 from variants var2_name and variant one.
    name = (var2_name=one).(var1_name=two)
    shortname = (var2_name=one).(var1_name=two)
    var1_name = two       # variant name in same namespace as variables.
    var2_name = one       # variant name in same namespace as variables.
```

Named variants could also be used as normal variables.:

```
variants guest_os:
    - fedora:
    - ubuntu:
variants disk_interface:
    - virtio:
    - hda:
```

Which then results in the following:

```
dict    1:  (disk_interface=virtio).(guest_os=fedora)
    dep = []
    disk_interface = virtio
    guest_os = fedora
    name = (disk_interface=virtio).(guest_os=fedora)
    shortname = (disk_interface=virtio).(guest_os=fedora)
dict    2:  (disk_interface=virtio).(guest_os=ubuntu)
    dep = []
    disk_interface = virtio
    guest_os = ubuntu
    name = (disk_interface=virtio).(guest_os=ubuntu)
    shortname = (disk_interface=virtio).(guest_os=ubuntu)
dict    3:  (disk_interface=hda).(guest_os=fedora)
    dep = []
    disk_interface = hda
    guest_os = fedora
    name = (disk_interface=hda).(guest_os=fedora)
    shortname = (disk_interface=hda).(guest_os=fedora)
dict    4:  (disk_interface=hda).(guest_os=ubuntu)
    dep = []
    disk_interface = hda
    guest_os = ubuntu
    name = (disk_interface=hda).(guest_os=ubuntu)
    shortname = (disk_interface=hda).(guest_os=ubuntu)
```

Dependencies Often it is necessary to dictate relationships between variants. In this way, the order of the resulting variant sets may be influenced. This is accomplished by listing the names of all parents (in order) after the child's vari-

ant name. However, the influence of dependencies is ‘weak’, in that any later defined, lower-level (higher indentation) definitions, and/or filters (see section [filters](#)) can remove or modify dependents. For example, if testing unattended installs, each virtual machine must be booted before, and shutdown after:

```
variants:
  - one:
      key1 = Hello
  - two: one
      key2 = World
  - three: one two
```

Results in the correct sequence of variant sets: one, two, *then* three.

Filters Filter statements allow modifying the resultant set of keys based on the name of the variant set (see section [variants](#)). Filters can be used in 3 ways: Limiting the set to include only combination names matching a pattern. Limiting the set to exclude all combination names not matching a pattern. Modifying the set or contents of key/value pairs within a matching combination name.

Names are matched by pairing a variant name component with the character(s) ‘,’ meaning OR, ‘.’ meaning AND, and ‘-’ meaning IMMEDIATELY-FOLLOWED-BY. When used alone, they permit modifying the list of key/values previously defined. For example:

```
Linux..OpenSuse:
initrd = initrd
```

Modifies all variants containing ‘Linux’ followed anywhere thereafter with ‘OpenSuse’, such that the ‘initrd’ key is created or overwritten with the value ‘initrd’.

When a filter is preceded by the keyword ‘only’ or ‘no’, it limits the selection of variant combination’s. This is used where a particular set of one or more variant combination’s should be considered selectively or exclusively. When given an extremely large matrix of variants, the ‘only’ keyword is convenient to limit the result set to only those matching the filter. Whereas the ‘no’ keyword could be used to remove particular conflicting key/value sets under other variant combination names. For example:

```
only Linux..Fedora..64
```

Would reduce an arbitrarily large matrix to only those variants who’s names contain Linux, Fedora, and 64 in them.

However, note that any of these filters may be used within named variants as well. In this application, they are only evaluated when that variant name is selected for inclusion (implicitly or explicitly) by a higher-order. For example:

```
variants:
  - one:
      key1 = Hello
variants:
  - two:
      key2 = Complicated
  - three: one two
      key3 = World
variants:
  - default:
      only three
      key2 =

only default
```

Results in the following outcome:


```
name = default.three.one
key1 = Hello
key2 =
key3 = World
```

Value Substitutions Value substitution allows for selectively overriding precedence and defining part or all of a future key's value. Using a previously defined key, it's value may be substituted in or as a another key's value. The syntax is exactly the same as in the bash shell, where as a key's value is substituted in wherever that key's name appears following a '\$' character. When nesting a key within other non-key-name text, the name should also be surrounded by '{', and '}' characters.

Replacement is context-sensitive, thereby if a key is redefined within the same, or, higher-order block, that value will be used for future substitutions. If a key is referenced for substitution, but hasn't yet been defined, no action is taken. In other words, the \$key or \${key} string will appear literally as or within the value. Nesting of references is not supported (i.e. key substitutions within other substitutions).

For example, if one = 1, two = 2, and three = 3; then, order = \${one}\${two}\${three} results in order = 123. This is particularly handy for rooting an arbitrary complex directory tree within a predefined top-level directory.

An example of context-sensitivity,

```
key1 = default value
key2 = default value

sub = "key1: ${key1}; key2: ${key2};"

variants:
  - one:
      key1 = Hello
      sub = "key1: ${key1}; key2: ${key2};"
  - two: one
      key2 = World
      sub = "key1: ${key1}; key2: ${key2};"
  - three: one two
      sub = "key1: ${key1}; key2: ${key2};"
```

Results in the following,

```
dict    1:  one
      dep = []
      key1 = Hello
      key2 = default value
      name = one
      shortname = one
      sub = key1: Hello; key2: default value;
dict    2:  two
      dep = ['one']
      key1 = default value
      key2 = World
      name = two
      shortname = two
      sub = key1: default value; key2: World;
dict    3:  three
      dep = ['one', 'two']
      key1 = default value
      key2 = default value
      name = three
```

```
shortname = three
sub = key1: default value; key2: default value;
```

Key sub-arrays Parameters for objects like VM's utilize array's of keys specific to a particular object instance. In this way, values specific to an object instance can be addressed. For example, a parameter 'vms' lists the VM objects names to instantiate in the current frame's test. Values specific to one of the named instances should be prefixed to the name:

```
vms = vm1 second_vm another_vm
mem = 128
mem_vm1 = 512
mem_second_vm = 1024
```

The result would be, three virtual machine objects are create. The third one (another_vm) receives the default 'mem' value of 128. The first two receive specialized values based on their name.

The order in which these statements are written in a configuration file is not important; statements addressing a single object always override statements addressing all objects. Note: This is contrary to the way the Cartesian configuration file as a whole is parsed (top-down).

Include statements The 'include' statement is utilized within a Cartesian configuration file to better organize related content. When parsing, the contents of any referenced files will be evaluated as soon as the parser encounters the `include` statement. The order in which files are included is relevant, and will carry through any key/value substitutions (see section [key_sub_arrays](#)) as if parsing a complete, flat file.

Combinatorial outcome The parser is available as both a python module and command-line tool for examining the parsing results in a text-based listing. To utilize it on the command-line, run the module followed by the path of the configuration file to parse. For example, `common_lib/cartesian_config.py tests/libvirt/tests.cfg`.

The output will be just the names of the combinatorial result set items (see short-names, section Variants). However, the '--contents' parameter may be specified to examine the output in more depth. Internally, the key/value data is stored/accessed similar to a python dictionary instance. With the collection of dictionaries all being part of a python list-like object. Irrespective of the internals, running this module from the command-line is an excellent tool for both reviewing and learning about the Cartesian Configuration format.

In general, each individual combination of the defined variants provides the parameters for a single test. Testing proceeds in order, through each result, passing the set of keys and values through to the harness and test code. When examining Cartesian configuration files, it's helpful to consider the earliest key definitions as "defaults", then look to the end of the file for other top-level override to those values. If in doubt of where to define or set a key, placing it at the top indentation level, at the end of the file, will guarantee it is used.

Formal definition

- A list of dictionaries is referred to as a frame.
- The parser produces a list of dictionaries (dicts). Each dictionary contains a set of key-value pairs.
- Each dict contains at least three keys: name, shortname and depend. The values of name and shortname are strings, and the value of depend is a list of strings.
- The initial frame contains a single dict, whose name and shortname are empty strings, and whose depend is an empty list.
- Parsing dict contents

- The dict parser operates on a frame, referred to as the current frame.
- A statement of the form `<key> = <value>` sets the value of `<key>` to `<value>` in all dicts of the current frame. If a dict lacks `<key>`, it will be created.
- A statement of the form `<key> += <value>` appends `<value>` to the value of `<key>` in all dicts of the current frame. If a dict lacks `<key>`, it will be created.
- A statement of the form `<key> <= <value>` pre-pends `<value>` to the value of `<key>` in all dicts of the current frame. If a dict lacks `<key>`, it will be created.
- A statement of the form `<key> ?= <value>` sets the value of `<key>` to `<value>`, in all dicts of the current frame, but only if `<key>` exists in the dict. The operators `?+=` and `?<=` are also supported.
- A statement of the form `no <regex>` removes from the current frame all dicts whose name field matches `<regex>`.
- A statement of the form `only <regex>` removes from the current frame all dicts whose name field does not match `<regex>`.
- Content exceptions
 - Single line exceptions have the format `<regex>: <key> <operator> <value>` where `<operator>` is any of the operators listed above (e.g. `=`, `+=`, `?<=`). The statement following the regular expression `<regex>` will apply only to the dicts in the current frame whose name partially matches `<regex>` (i.e. contains a substring that matches `<regex>`).
 - A multi-line exception block is opened by a line of the format `<regex>:`. The text following this line should be indented. The statements in a multi-line exception block may be assignment statements (such as `<key> = <value>`) or no or only statements. Nested multi-line exceptions are allowed.
- Parsing Variants
 - A variants block is opened by a `variants:` statement. The indentation level of the statement places the following set within the outer-most context-level when nested within other `variant:` blocks. The contents of the `variants:` block must be further indented.
 - A variant-name may optionally follow the `variants` keyword, before the `:` character. That name will be inherited by and decorate all block content as the key for each variant contained in it's the block.
 - The name of the variants are specified as `- <variant_name>:`. Each name is pre-pended to the name field of each dict of the variant's frame, along with a separator dot (`'.'`).
 - The contents of each variant may use the format `<key> <op> <value>`. They may also contain further `variants:` statements.
 - If the name of the variant is not preceeded by a `@` (i.e. `- @<variant_name>:`), it is pre-pended to the shortname field of each dict of the variant's frame. In other words, if a variant's name is preceeded by a `@`, it is omitted from the shortname field.
 - Each variant in a variants block inherits a copy of the frame in which the `variants:` statement appears. The 'current frame', which may be modified by the dict parser, becomes this copy.
 - The frames of the variants defined in the block are joined into a single frame. The contents of frame replace the contents of the outer containing frame (if there is one).
- Filters
 - Filters can be used in 3 ways:

```
* only <filter>
```

```
* no <filter>
```

```
* <filter>: (starts a conditional block, see 4.4 Filters)
```

– Syntax:

```
.. means AND  
. means IMMEDIATELY-FOLLOWED-BY
```

• Example:

```
qcow2..Fedora.14, RHEL.6..raw..boot, smp2..qcow2..migrate..ide
```

```
means match all dicts whose names have:  
(qcow2 AND (Fedora IMMEDIATELY-FOLLOWED-BY 14)) OR  
((RHEL IMMEDIATELY-FOLLOWED-BY 6) AND raw AND boot) OR  
(smp2 AND qcow2 AND migrate AND ide)
```

• Note:

```
'qcow2..Fedora.14' is equivalent to 'Fedora.14..qcow2'.
```

```
'qcow2..Fedora.14' is not equivalent to 'qcow2..14.Fedora'.  
'ide, scsi' is equivalent to 'scsi, ide'.
```

Examples

• A single dictionary:

```
key1 = value1  
key2 = value2  
key3 = value3  
  
Results in the following::  
  
Dictionary #0:  
  depend = []  
  key1 = value1  
  key2 = value2  
  key3 = value3  
  name =  
  shortname =
```

• Adding a variants block:

```
key1 = value1  
key2 = value2  
key3 = value3  
  
variants:  
  - one:  
  - two:  
  - three:
```

Results in the following:

```
Dictionary #0:  
  depend = []
```

```

    key1 = value1
    key2 = value2
    key3 = value3
    name = one
    shortname = one
Dictionary #1:
    depend = []
    key1 = value1
    key2 = value2
    key3 = value3
    name = two
    shortname = two
Dictionary #2:
    depend = []
    key1 = value1
    key2 = value2
    key3 = value3
    name = three
    shortname = three

```

- Modifying dictionaries inside a variant:

```

key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two:
      key2 <= another_prefix_
  - three:

```

Results in the following:

```

Dictionary #0:
    depend = []
    key1 = Hello World
    key2 = some_prefix_value2
    key3 = value3
    name = one
    shortname = one
Dictionary #1:
    depend = []
    key1 = value1
    key2 = another_prefix_value2
    key3 = value3
    name = two
    shortname = two
Dictionary #2:
    depend = []
    key1 = value1
    key2 = value2
    key3 = value3
    name = three
    shortname = three

```

- Adding dependencies:

```
key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two: one
      key2 <= another_prefix_
  - three: one two
```

Results in the following:

```
Dictionary #0:
  depend = []
  key1 = Hello World
  key2 = some_prefix_value2
  key3 = value3
  name = one
  shortname = one
Dictionary #1:
  depend = ['one']
  key1 = value1
  key2 = another_prefix_value2
  key3 = value3
  name = two
  shortname = two
Dictionary #2:
  depend = ['one', 'two']
  key1 = value1
  key2 = value2
  key3 = value3
  name = three
  shortname = three
```

- Multiple variant blocks:

```
key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two: one
      key2 <= another_prefix_
  - three: one two

variants:
  - A:
  - B:
```

Results in the following:

```
Dictionary #0:
  depend = []
  key1 = Hello World
  key2 = some_prefix_value2
```

```

    key3 = value3
    name = A.one
    shortname = A.one
Dictionary #1:
    depend = ['A.one']
    key1 = value1
    key2 = another_prefix_value2
    key3 = value3
    name = A.two
    shortname = A.two
Dictionary #2:
    depend = ['A.one', 'A.two']
    key1 = value1
    key2 = value2
    key3 = value3
    name = A.three
    shortname = A.three
Dictionary #3:
    depend = []
    key1 = Hello World
    key2 = some_prefix_value2
    key3 = value3
    name = B.one
    shortname = B.one
Dictionary #4:
    depend = ['B.one']
    key1 = value1
    key2 = another_prefix_value2
    key3 = value3
    name = B.two
    shortname = B.two
Dictionary #5:
    depend = ['B.one', 'B.two']
    key1 = value1
    key2 = value2
    key3 = value3
    name = B.three
    shortname = B.three

```

- Filters, no and only:

```

key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two: one
      key2 <= another_prefix_
  - three: one two

variants:
  - A:
      no one
  - B:
      only one,three

```

Results in the following:

```
Dictionary #0:
  depend = ['A.one']
  key1 = value1
  key2 = another_prefix_value2
  key3 = value3
  name = A.two
  shortname = A.two
Dictionary #1:
  depend = ['A.one', 'A.two']
  key1 = value1
  key2 = value2
  key3 = value3
  name = A.three
  shortname = A.three
Dictionary #2:
  depend = []
  key1 = Hello World
  key2 = some_prefix_value2
  key3 = value3
  name = B.one
  shortname = B.one
Dictionary #3:
  depend = ['B.one', 'B.two']
  key1 = value1
  key2 = value2
  key3 = value3
  name = B.three
  shortname = B.three
```

- Short-names:

```
key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two: one
      key2 <= another_prefix_
  - three: one two

variants:
  - @A:
      no one
  - B:
      only one,three
```

Results in the following:

```
Dictionary #0:
  depend = ['A.one']
  key1 = value1
  key2 = another_prefix_value2
  key3 = value3
  name = A.two
  shortname = two
```



```

Dictionary #1:
  depend = ['A.one', 'A.two']
  key1 = value1
  key2 = value2
  key3 = value3
  name = A.three
  shortname = three
Dictionary #2:
  depend = []
  key1 = Hello World
  key2 = some_prefix_value2
  key3 = value3
  name = B.one
  shortname = B.one
Dictionary #3:
  depend = ['B.one', 'B.two']
  key1 = value1
  key2 = value2
  key3 = value3
  name = B.three
  shortname = B.three

```

- Exceptions:

```

key1 = value1
key2 = value2
key3 = value3

variants:
  - one:
      key1 = Hello World
      key2 <= some_prefix_
  - two: one
      key2 <= another_prefix_
  - three: one two

variants:
  - @A:
      no one
  - B:
      only one,three

three: key4 = some_value

A:
  no two
  key5 = yet_another_value

```

Results in the following:

```

Dictionary #0:
  depend = ['A.one', 'A.two']
  key1 = value1
  key2 = value2
  key3 = value3
  key4 = some_value
  key5 = yet_another_value
  name = A.three
  shortname = three

```

```
Dictionary #1:
  depend = []
  key1 = Hello World
  key2 = some_prefix_value2
  key3 = value3
  name = B.one
  shortname = B.one
Dictionary #2:
  depend = ['B.one', 'B.two']
  key1 = value1
  key2 = value2
  key3 = value3
  key4 = some_value
  name = B.three
  shortname = B.three
```

Default Configuration Files

The test configuration files are used for controlling the framework, by specifying parameters for each test. The parser produces a list of key/value sets, each set pertaining to a single test. Variants are organized into separate files based on scope and/or applicability. For example, the definitions for guest operating systems is sourced from a shared location since all virtualization tests may utilize them.

For each set/test, keys are interpreted by the test dispatching system, the pre-processor, the test module itself, then by the post-processor. Some parameters are required by specific sections and others are optional. When required, parameters are often commented with possible values and/or their effect. There are select places in the code where in-memory keys are modified, however this practice is discouraged unless there's a very good reason.

When `./run --bootstrap` executed (see section [run_bootstrap](#)), copies of the sample configuration files are copied for use under the `cfg` subdirectory of the virtualization technology-specific directory. For example, `qemu/cfg/base.cfg`. These copies are the versions used by the framework for both the autotest client and test-runner.

Relative Directory or File	Description
cfg/tests.cfg	The first file read that includes all other files, then the master set of filters to select the actual test set to be run. Normally this file never needs to be modified unless precise control over the test-set is needed when utilizing the autotest-client (only).
cfg/tests-shared.cfg	Included by <code>tests.cfg</code> to indirectly reference the remaining set of files to include as well as set some global parameters. It is used to allow customization and/or insertion within the set of includes. Normally this file never needs to be modified.
cfg/base.cfg	Top-level file containing important parameters relating to all tests. All keys/values defined here will be inherited by every variant unless overridden. This is the <i>first</i> file to check for settings to change based on your environment
cfg/build.cfg	Configuration specific to pre-test code compilation where required/requested. Ignored when a client is not setup for build testing.
cfg/subtests.cfg	Automatically generated based on the test modules and test configuration files found when the <code>./run --bootstrap</code> is used. Modifications are discouraged since they will be lost next time <code>--bootstrap</code> is used.
cfg/guest-os.cfg	Automatically generated from files within <code>shared/cfg/guest-os/</code> . Defines all supported guest operating system types, architectures, installation images, parameters, and disk device or image names.
cfg/guest-hw.cfg	All virtual and physical hardware related parameters are organized within variant names. Within subtest variants or the top-level test set definition, hardware is specified by Including, excluding, or filtering variants and keys established in this file.
cfg/cdkeys.cfg	Certain operating systems require non-public information in order to operate and or install properly. For example, installation numbers and license keys. None of the values in this file are populated automatically. This file should be edited to supply this data for use by the unattended install test.
cfg/virtio-win.cfg	Paravirtualized hardware when specified for Windows testing, must have dependent drivers installed as part of the OS installation process. This file contains mandatory variants and keys for each Windows OS version, specifying the host location and installation method for each driver.

Configuration file details

Base Nearly as important as `tests.cfg`, since it's the first file processed. This file is responsible for defining all of the top-level default settings inherited by all hardware, software, subtest, and run-time short-name variants. It's critical for establishing the default networking model of the host system, pathnames, and the virtualization technology being tested. It also contains guest options that don't fit within the context of the other configuration files, such as default memory size, console video creation for tests, and guest console display options (for human monitoring). When getting started in virtualization autotest, or setting up on a new host, this is usually the file to edit first.

tests The `tests.cfg` file is responsible for acting on the complete collection of variants available and producing a useful result. In other words, all other configuration files (more or less) define "what is possible", `tests.cfg` defines what will actually happen.

In the order they appear, there are three essential sections:

- A set of pre-configured example short-name variants for several OS's, hypervisor types, and virtual hardware configurations. They can be used directly, and/or copied and modified as needed.
- An overriding value-filter set, which adjusts several key path-names and file locations that are widely applicable.
- The final top-level scoping filter set for limiting the tests to run, among the many available.

The default configuration aims to support the quick-start (see section [run](#)) with a simple and minimal test set that's easy to get running. It calls on a variant defined within the pre-configured example set as described above. It also provides the best starting place for exploring the configuration format and learning about how it's used to support virtualization testing.

cdkeys and windows virtio This is the least-accessed among the configuration files. It exists because certain operating systems require non-public information in order to operate and or install properly. Keeping this data stored in a special purpose file, keeps the data allows its privacy level to be controlled. None of the values in this file are populated automatically. This file should be hand-edited to supply this data for use by the autotest client. It is not required for the default test configured in `tests.cfg`.

The windows-centric `virtio-win.cfg` file is similar in that it is only applicable to windows guest operating systems. It supplements windows definitions from `guest-os.cfg` with configuration needed to ensure the virtio drivers are available during windows installation.

To install the virtio drivers during guest install, virtualization autotest has to inform the windows install programs **where** to find the drivers. Virtualization autotest uses a boot floppy with a Windows answer file in order to perform unattended install of windows guests. For winXP and win2003, the unattended files are simple `.ini` files, while for win2008 and later, the unattended files are XML files. Therefore, it makes the following assumptions:

- An iso file is available that contains windows virtio drivers (inf files) for both netkvm and viostor.
- For WinXP or Win2003, a pre-made floppy disk image is available with the virtio drivers and a configuration file the Windows installer will read, to fetch the right drivers.
- Comfort and familiarity editing and working with the Cartesian configuration file format, setting key values and using filters to point virtualization autotest at host files.

guest hw & guest os Two of the largest and most complex among the configuration files, this pair defines a vast number of variants and keys relating purely to guest operating system parameters and virtual hardware. Their intended use is from within `tests.cfg` (see section [tests](#)). Within `tests.cfg` short-name variants, filters are used for both OS and HW variants in these files to choose among the many available sets of options.

For example if a test requires the virtio network driver is used, it would be selected with the filter `'only virtio_net'`. This filter means content of the `virtio_net` variant is included from `guest-hw.cfg`, which in turn results in the `'nic_model = virtio'` definition. In a similar manner, all guest installation methods (with the exception of virtio for Windows) and operating system related parameters are set in `guest-os.cfg`.

Sub-tests The third most complex of the configurations, `subtests.cfg` holds variants defining all of the available virtualization sub-tests available. They include definitions for running nested non-virtualization autotest tests within guests. For example, the simplistic `'sleeptest'` may be run with the filter `'only autotest.sleeptest'`.

The `subtests.cfg` file is rarely edited directly, instead it's intended to provide a reasonable set of defaults for testing. If particular test keys need customization, this should be done within the short-name variants defined or created in `tests.cfg` (see section [tests](#)). However, available tests and their options are commented within `subtests.cfg`, so it is often referred to as a source for available tests and their associated controls.

Configuration usage details For a complete reference, refer to [the cartesian config params documentation](#)

Preserving installed Guest images See [Run tests on an existing guest](#)

Specialized Networking See [Autotest networking documentation](#)

Using virtio drivers with windows Required items include access to the virtio driver installation image, the Windows ISO files, and the `winutils.iso` CD (See section [run_bootstrap](#)). Every effort is made to standardize on files available from MSDN. For example, using the Windows7 64 bit (non SP1) requires the CD matching:

```
• cdrom_cd1 = isos/windows/en_windows_7_ultimate_x86_dvd_x15-65921.iso
```

```
• sha1sum_cd1 = 5395dc4b38f7bdb1e005ff414deedfdb16dbf610
```

This file can be downloaded from the MSDN site then it's SHA1 verified.

Next, place the windows media image (creating directory if needed) in `shared/data/isos/windows/`. Edit the `cfg/cdkeys.cfg` file to supply license information if required.

Finally, if not using the test runner, set up `cfg/tests.cfg` to include the `windows_quick` short-name variant (see section [tests](#)). Modify the network and block device filters to use `'virtio_net'` and `'virtio-blk'` instead.

Development tools / utilities

A number of utilities are available for the autotest core, client, and/or test developers. Depending on your installation type, these may be located in different sub-directories of the tree.

Name	Description
<code>run_pylint.py</code>	Wrapper is required to run pylint due to the way imports have been implemented.
<code>check_patch.py</code>	Help developers scan code tree and display or fix problems
<code>reindent.py</code>	Help developers fix simple indentation problems

Contributions

Code Contributions of additional tests and code are always welcome. If in doubt, and/or for advice on approaching a particular problem, please contact the projects members (see section [_collaboration](#)) Before submitting code, please review the [git repository configuration guidelines](#).

To submit changes, please follow [these instructions](#). Please allow up to two weeks for a maintainer to pick up and review your changes. Though, if you'd like help at any stage, feel free to post on the mailing lists and reference your pull request.

Docs Please edit the documentation directly to correct any minor inaccuracies or to clarify items. The preferred markup syntax is [ReStructuredText](#), keeping with the conventions and style found in existing documentation. For any graphics or diagrams, web-friendly formats should be used, such as PNG or SVG.

Avoid using 'you', 'we', 'they', as they can be ambiguous in reference documentation. It works fine in conversation and e-mail, but looks weird in reference material. Similarly, avoid using 'unnecessary', off-topic, or extra language. For example in American English, "[Rinse and repeat](#)" is a funny phrase, but could cause problems when translated into other languages. Basically, try to avoid anything that slows the reader down from finding facts.

For major documentation work, it's more convenient to use a different approach. The autotest wiki is stored on github as a separate repository from the project code. The wiki repository contains all the files, and allows for version control over them. To clone the wiki repository, click the `Clone URL` button on the wiki page (next to `Page History`).

When working with the wiki repository, it's sometimes convenient to render the wiki pages locally while making and committing changes. The `gollum` ruby gem may be installed so you can view the wiki locally. See [the gollum wiki readme](#) for more details.

`_contact_info`:

Contact Info. [Please refer to this page](#)

Building test applications

This is a description of how to build test applications from a test case.

Dependencies

If you write an application that is supposed to be run on the test-target, place it in the directory `../deps/<name>/` relative to where your test case is placed. The easiest way to obtain the full path to this directory is by calling `data_dir.get_deps_dir("<name>")`. Don't forget to add `from virttest import data_dir` to your test case.

Besides the source file, create a Makefile that will be used to build your test application. The below example shows a Makefile for the application for the timedrift test cases. The `remote_build` module requires that a Makefile is included with all test applications.

```
CFLAGS+=-Wall
LDLIBS+=-lrt

.PHONY: clean

all: clktest get_tsc

clktest: clktest.o

get_tsc: get_tsc.o

clean:
    rm -f clktest get_tsc
```

remote_build

To simplify the building of applications on target, and to simplify avoiding the building of applications on target when they are installed pre-built, use the `remote_build` module. This module handles both the transfer of files, and running `make` on target.

A simple example:

```
address = vm.get_address(0)
source_dir = data_dir.get_deps_dir("<testapp>")
builder = remote_build.Builder(params, address, source_dir)
full_build_path = builder.build()
```

In this case, we utilize the `.build()` method, which execute the necessary methods in `builder` to copy all files to target and run make (if needed). When done, `.build()` will return the full path on target to the application that was just built. Be sure to use this path when running your test application, as the path is changed if the parameters of the build is changed. For example:

```
session.cmd_status(%s --test" % os.path.join(full_build_path, "testapp"))
```

The `remote_build.Builder` class can give you fine-grained control over your build process as well. Another way to write the above `.build()` invocation above is:

```
builder = remote_build.Builder(params, address, source_dir)
if builder.sync_directories():
```

```
builder.make()
full_build_path = builder.full_build_path
```

This pattern can be useful if you e.g. would like to add an additional command to run before *builder.make()*, perhaps to install some extra dependencies.

Running tests on an existing guest image

virt-test knows how to install guests, and that's all fine, but most of the time, users already have a guest image they are working on, and just want to tell virt-test to use it. Also, virt-test is a large piece of infrastructure, and it's not really obvious how all the pieces fit together, so some help is required to dispose the available pieces conveniently, so users can accomplish their own testing goals. So, let's get started.

A bit of context on how autotest works

The default upstream configuration file instructs autotest to perform the following tasks:

1. Install a Linux guest, on this case, Fedora 15, due to the fact it is publicly available, so *everybody* can try it out. The hardware configuration for the VM:
 - one qcow2 image on an ide bus
 - one network card, model rtl8139
 - two cpus
 - no pci hotplug devices will be attached to this vm
 - Also, the VM is not going to use hugepage memory explicitly
2. Run a boot test.
3. Run a shutdown test.

```
# Runs qemu-kvm, f15 64 bit guest OS, install, boot, shutdown
- @qemu_kvm_f15_quick:
  # We want qemu-kvm for this run
  qemu_binary = /usr/bin/qemu-kvm
  qemu_img_binary = /usr/bin/qemu-img
  only qcow2
  only rtl8139
  only ide
  only smp2
  only no_pci_assignable
  only smallpages
  only Fedora.15.64
  only unattended_install.cdrom, boot, shutdown
```

This is defined in such a way that the kvm test config system will generate only 3 tests. Let's see at the tests generated. On the kvm test dir \$AUTOTEST_ROOT/client/tests/kvm, you can call the configuration parser:

```
[lmr@freedom kvm]$ ../../common_lib/cartesian_config.py tests.cfg
dict    1:  smp2.Fedora.15.64.unattended_install.cdrom
dict    2:  smp2.Fedora.15.64.boot
dict    3:  smp2.Fedora.15.64.shutdown
```

You can see on top of the file tests.cfg some *includes* that point us from where all the test information comes from:

```
# Copy this file to tests.cfg and edit it.
#
# This file contains the test set definitions. Define your test sets here.
include tests_base.cfg
include cdkeys.cfg
include virtio-win.cfg
```

tests_base.cfg is a pretty large file, that contains a lot of *variants*, that are blocks defining tests, vm hardware and pre/post processing directives that control autotest infrastructure behavior. You can check out the definition of each of the variants restricting the test sets (qcow2, rtl8139, smp2, no_pci_assignable, smallpages, Fedora 15.64, unattended_install.cdrom, boot, shutdown) on tests_base.cfg.

About guest install

It is no mystery that for a good deal of the virtualization tests we are going to execute, a guest with an *operating system* on its disk image is needed. To get this OS there, we have some methods defined:

1. Install the VM with the OS CDROM through an engine that interacts with the VM using VNC, simulating a human being, called *step engine*. This engine works surprisingly well, frequently yielding successful installations. However, each *step* is a point of failure of the whole process, so we moved towards handing the dirty install control to the guest OS itself, as many of them have this capacity.
2. Install the VM using the automated install mechanisms provided by the guest OS itself. In windows, we have a mechanism called *answer files*, for Fedora and RHEL we have *kickstarts*, and for OpenSUSE we have *autoyast*. Of course, other OS, such as debian, also have their own mechanism, however they are not currently implemented in autotest (hint, hint).

And then an even simpler alternative:

1. Just copy a known good guest OS image, that was already installed, and use it. This tends to be faster and less error prone, since the install *is already done*, so we don't need to work on failures on this step. The inevitable question that arises:
 - *Q. Hey, why don't you just go with that on the first place?*
 - *A. Because installing a guest exercises several aspects of the VM, such as disk, network, hardware probing, so on and so forth, so it's a good functional test by itself. Also, installing manually a guest may work well for a single developer working on his/her patches, but certainly does not scale if you need fully automated test done on a regular basis, as there is the need of someone going there and making the install, which seriously, is a waste of human resources. KVM autotest is also a tool for doing such a massively automated test on a regular basis.*

Also, this method assumes the least possible for the person running the tests, as they won't need to have preinstalled guests, and because we *always* get the same vm, with the same capabilities and same configuration. Now that we made this point clear, let's explain how to use your preinstalled guest.

Needed setup for a typical linux guest

virt-test relies heavily on *cartesian config files*. Those files use a flexible file format, defined on [the file format documentation](#) If you are curious about the defaults assumed for Linux or Windows guests, you can always check the file base.cfg.sample, which contains all our guest definitions (look at the Linux or Windows variant). Without diving too much into it, it's sufficient to say that you need a guest to have a root password of 123456 and an enabled ssh daemon which will allow you to log in as root. The password can be also configured through the config files.

Before you start

1. Make sure you have the appropriate packages installed. You can read [the install prerequisite packages \(client section\)](#) for more information. For this how to our focus is not to build kvm from git repos, so we are assuming you are going to use the default qemu installed in the system. However, if you are interested in doing so, you might want to recap our docs on [building qemu-kvm and running unittests](#).

Step by step procedure

1. Git clone autotest to a convenient location, say \$HOME/Code/autotest. See [the download source documentation](#). Please do use git and clone the repo to the location mentioned.
2. Execute the `./run -t qemu --bootstrap` command (see [the get started documentation <GetStarted>](#)). Since we are going to boot our own guests, you can safely skip each and every iso download possible.
3. Edit the file `tests.cfg`. You can see we have a session overriding Custom Guest definitions present on `tests_base.cfg`. If you want to use a raw block device (see [image_raw_device](#)), you can uncomment the lines mentioned on the comments. When `image_raw_device = yes`, virt-test will not append a ‘qcow2’ extension to the image name. **Important:** If you opt for a raw device, you must comment out the line that appends a base path to image names (one that looks like `image_name(.*?) ?<= /tmp/kvm_autotest_root/images/`)

```
CustomGuestLinux:
    # Here you can override the default login credentials for your custom guest
    username = root
    password = 123456
    image_name = custom_image_linux
    image_size = 10G
    # If you want to use a block device as the vm disk, uncomment the 2 lines
    # below, pointing the image name for the device you want
    #image_name = /dev/mapper/vg_linux_guest
    #image_raw_device = yes
```

4. Some lines below, you will also find this config snippet. This is for the case where you want to specify new base directories for kvm autotest to look images, cdroms and floppies.

```
# Modify/comment the following lines if you wish to modify the paths of the
# image files, ISO files or qemu binaries.
#
# As for the defaults:
# * qemu and qemu-img are expected to be found under /usr/bin/qemu-kvm and
#   /usr/bin/qemu-img respectively.
# * All image files are expected under /tmp/kvm_autotest_root/images/
# * All install iso files are expected under /tmp/kvm_autotest_root/isos/
# * The parameters cdrom_unattended, floppy, kernel and initrd are generated
#   by virt-test, so remember to put them under a writable location
#   (for example, the cdrom share can be read only)
image_name(.*?) ?<= /tmp/kvm_autotest_root/images/
cdrom(.*?) ?<= /tmp/kvm_autotest_root/
floppy ?<= /tmp/kvm_autotest_root/
```

5. Change the fields `image_name`, `image_size` to your liking. Now, the **example** test set that uses custom guest configuration can be found some lines below:

```
# Runs your own guest image (qcow2, can be adjusted), all migration tests
# (on a core2 duo laptop with HD and 4GB RAM, F15 host took 3 hours to run)
# Be warned, disk stress + migration can corrupt your image, so make sure
# you have proper backups
```

```
- @qemu_kvm_custom_migrate:
    # We want qemu-kvm for this run
    qemu_binary = /usr/bin/qemu-kvm
    qemu_img_binary = /usr/bin/qemu-img
    only qcow2
    only rtl8139
    only ide
    only smp2
    only no_pci_assignable
    only smallpages
    only CustomGuestLinux
    only migrate
```

6. Since we want to execute this custom migrate test set, we need to look at the last couple of lines of the configuration file:

```
# Choose your test list from the testsets defined
only qemu_kvm_f15_quick
```

7. This line needs to become

```
# Choose your test list from the testsets defined
only qemu_kvm_custom_migrate
```

8. Now, if you haven't changed any of the settings of the previous blocks, now our configuration system will run tests with the following expectations:

- `qemu-kvm` and `qemu` are under `/usr/bin/qemu-kvm` and `/usr/bin/qemu-kvm`, respectively. *Please remember RHEL installs `qemu-kvm` under `"/usr/libexec"`.*
 - Our guest image is under `/tmp/kvm_autotest_root/images/custom_image_linux.qcow2`, since the test set specifies only `qcow2`.
 - All current combinations for our migrate tests variant will be executed with your custom image. It is never enough to remember that some of the tests can corrupt your `qcow2` (or `raw`) image.
1. If you want to verify all tests that the config system will generate, you can run the parser to tell you that. This set took 3 hours to run on my development laptop setup.

```
[lmr@freedom kvm]$ ../../common_lib/cartesian_config.py tests.cfg
dict 1: smp2.CustomGuestLinux.migrate.tcp
dict 2: smp2.CustomGuestLinux.migrate.unix
dict 3: smp2.CustomGuestLinux.migrate.exec
dict 4: smp2.CustomGuestLinux.migrate.mig_cancel
dict 5: smp2.CustomGuestLinux.migrate.with_set_speed.tcp
dict 6: smp2.CustomGuestLinux.migrate.with_set_speed.unix
dict 7: smp2.CustomGuestLinux.migrate.with_set_speed.exec
dict 8: smp2.CustomGuestLinux.migrate.with_set_speed.mig_cancel
dict 9: smp2.CustomGuestLinux.migrate.with_reboot.tcp
dict 10: smp2.CustomGuestLinux.migrate.with_reboot.unix
dict 11: smp2.CustomGuestLinux.migrate.with_reboot.exec
dict 12: smp2.CustomGuestLinux.migrate.with_reboot.mig_cancel
dict 13: smp2.CustomGuestLinux.migrate.with_file_transfer.tcp
dict 14: smp2.CustomGuestLinux.migrate.with_file_transfer.unix
dict 15: smp2.CustomGuestLinux.migrate.with_file_transfer.exec
dict 16: smp2.CustomGuestLinux.migrate.with_file_transfer.mig_cancel
dict 17: smp2.CustomGuestLinux.migrate.with_autotest.dbench.tcp
dict 18: smp2.CustomGuestLinux.migrate.with_autotest.dbench.unix
dict 19: smp2.CustomGuestLinux.migrate.with_autotest.dbench.exec
dict 20: smp2.CustomGuestLinux.migrate.with_autotest.dbench.mig_cancel
```

```
dict    21:  smp2.CustomGuestLinux.migrate.with_autotest.stress.tcp
dict    22:  smp2.CustomGuestLinux.migrate.with_autotest.stress.unix
dict    23:  smp2.CustomGuestLinux.migrate.with_autotest.stress.exec
dict    24:  smp2.CustomGuestLinux.migrate.with_autotest.stress.mig_cancel
dict    25:  smp2.CustomGuestLinux.migrate.with_autotest.monotonic_time.tcp
dict    26:  smp2.CustomGuestLinux.migrate.with_autotest.monotonic_time.unix
dict    27:  smp2.CustomGuestLinux.migrate.with_autotest.monotonic_time.exec
dict    28:  smp2.CustomGuestLinux.migrate.with_autotest.monotonic_time.mig_cancel
```

2. If you want to make sure virt-test is assigning images to the right places, you can tell the config system to print the params contents for each test.

```
[lmr@freedom kvm]$ ../../common_lib/cartesian_config.py -c tests.cfg | less
... lots of output ...
```

3. In any of the dicts you should be able to see an `image_name` key that has something like the below. `virt-test` will only append `'image_format'` to this path and then use it, so in the case mentioned, `'/tmp/kvm_autotest_root/images/custom_image_linux.qcow2'`

```
image_name = /tmp/kvm_autotest_root/images/custom_image_linux
```

4. After you have verified things, you can run autotest using the command line `get_started.py` has informed you:

```
$AUTOTEST_ROOT/client/bin/autotest $AUTOTEST_ROOT/client/tests/kvm/control
```

5. Profit!

Common questions

- Q: How do I restrict the test set so it takes less time to run?
- A: You can look at the output of the cartesian config parser and check out the test combinations. If you look at the output above, and say you want to run only migration + file transfer tests, your test set would look like the below snippet. Make sure you validate your changes calling the parser again.

```
# Runs your own guest image (qcow2, can be adjusted), all migration tests
# (on a core2 duo laptop with HD and 4GB RAM, F15 host took 3 hours to run)
# Be warned, disk stress + migration can corrupt your image, so make sure
# you have proper backups
- @qemu_kvm_custom_migrate:
    # We want qemu-kvm for this run
    qemu_binary = /usr/bin/qemu-kvm
    qemu_img_binary = /usr/bin/qemu-img
    only qcow2
    only rtl8139
    only ide
    only smp2
    only no_pci_assignable
    only smallpages
    only CustomGuestLinux
    only migrate.with_file_transfer
```

```
[lmr@freedom kvm]$ ../../common_lib/cartesian_config.py tests.cfg
dict    1:  smp2.CustomGuestLinux.migrate.with_file_transfer.tcp
dict    2:  smp2.CustomGuestLinux.migrate.with_file_transfer.unix
dict    3:  smp2.CustomGuestLinux.migrate.with_file_transfer.exec
dict    4:  smp2.CustomGuestLinux.migrate.with_file_transfer.mig_cancel
```

Profiling

What is profiling

Profiling, by its definition (see [this wikipedia article](#) for a non formal introduction), is to run an analysis tool to inspect the behavior of a certain property of the system (be it memory, CPU consumption or any other).

How autotest can help with profiling?

Autotest provides support for running profilers during the execution of tests, so we know more about a given system resource. For the `kvm` test, our first idea of profiling usage was to run the `kvm_stat` program, that usually ships with `kvm`, to provide data useful for debugging. `kvm_stat` provides the number of relevant `kvm` events every time it is called, so by the end of a `virt-test` test we end up with a long list of information like this one:

kvm_ack_i	kvm_age_p	kvm_apic	kvm_apic_	kvm_apic_	kvm_async	kvm_async	kvm_async	kvm_async	kvm_async	kvm
1	54	11	5	0	0	0	0	0	0	0

How to control the execution of profilers ?

Profiling in `virt-test` is controlled through configuration files. You can set the profilers that are going to run by setting the variable `profilers`. On `tests_base.cfg.sample`, the section of the file that sets the profilers that run by default looks like this:

```
# Profilers. You can add more autotest profilers (see list on client/profilers)
# to the line below. You can also choose to remove all profilers so no profiling
# will be done at all.
profilers = kvm_stat
```

How to add a profiler?

So, say you want to run the `perf` profiler in addition to `kvm_stat`. You can just edit that place and put ‘`perf`’ right next to it:

```
# Profilers. You can add more autotest profilers (see list on client/profilers)
# to the line below. You can also choose to remove all profilers so no profiling
# will be done at all.
profilers = kvm_stat perf
```

How to remove all profilers (including `kvm_stat`)?

If you want no profiling at all for your tests, `profilers` can be changed to be an empty string:

```
# Profilers. You can add more autotest profilers (see list on client/profilers)
# to the line below. You can also choose to remove all profilers so no profiling
# will be done at all.
profilers =
```

Of course, the config system makes it easy to override the value of *any* param for your test variable, so you can have fine grained control of things. Say you don’t want to run profilers on your new ‘`crazy_test`’ variant, which you have developed. Easy:

```
- crazy_test:
    type = crazy_test
    profilers =
```

So this will turn of profilers just for this particular test of yours.

Networking

Here we have notes about networking setup in virt-test.

Configuration

How to configure to allow all the traffic to be forwarded across the virbr0 bridge:

```
echo "-I FORWARD -m physdev --physdev-is-bridged -j ACCEPT" > /etc/sysconfig/iptables-forward-bridged
lokit --custom-rules=ipv4:filter:/etc/sysconfig/iptables-forward-bridged
service libvirtd reload
```

How to configure Static IP address in virt-test

Sometimes, we need to test with guest(s) which have static ip address(es).

- eg. No real/emulated DHCP server in test environment.
- eg. Test with old image we don't want to change the net config.
- eg. Test when DHCP exists problem.

Create a bridge (for example, 'vbr') in host, configure its ip to 192.168.100.1, guest can access host by it. And assign nic(s)' ip in tests.cfg, and execute test as usual.

tests.cfg:

```
ip_nic1 = 192.168.100.119
nic_mac_nic1 = 11:22:33:44:55:67
bridge = vbr
```

TestCases

Nttcp

The Nttcp test suite is a network performance test for windows, developed by Microsoft. It is *not* a freely redistributable binary, so you must download it from the website, here's the direct link for download (keep in mind it might change):

<http://download.microsoft.com/download/f/1/e/f1e1ac7f-e632-48ea-83ac-56b016318735/NT%20Testing%20TCP%20Tool.msi>

The knowledge base article associated with it is:

<http://msdn.microsoft.com/en-us/windows/hardware/gg463264>

You need to add the package to winutils.iso, the iso with utilities used to test windows. First, download the iso. [The get started documentation](#) can help you out with downloading if you like it, but the direct download link is here:

<http://lmr.fedorapeople.org/winutils/winutils.iso>

You need to put all its contents on a folder and create a new iso. Let's say you want to download the iso to `/home/kermit/Downloads/winutils.iso`. You can create the directory, go to it:

```
mkdir -p /home/kermit/Downloads
cd /home/kermit/Downloads
```

Download the iso, create 2 directories, 1 for the mount, another for the contents:

```
wget http://people.redhat.com/mrodrigu/kvm/winutils.iso
mkdir original
sudo mount -o loop winutils.iso original
mkdir winutils
```

Copy all contents from the original cd to the new structure:

```
cp -r original/* winutils/
```

Create the destination nttcp directory on that new structure:

```
mkdir -p winutils/NTttcp
```

Download the installer and copy autoit script to the new structure, unmount the original mount:

```
cd winutils/NTttcp
wget http://download.microsoft.com/download/f/1/e/f1e1ac7f-e632-48ea-83ac-56b016318735/NT%20Testing%
cp /usr/local/autotest/client/virt/scripts/ntttcp.au3 ./
sudo umount original
```

Backup the old winutils.iso and create a new winutils.iso using mkisofs:

```
sudo mv winutils.iso winutils.iso.bak
mkisofs -o winutils.iso -max-iso9660-filenames -relaxed-filenames -D --input-charset iso8859-1 winut
```

And that is it. Don't forget to keep winutils in an appropriate location that can be seen by virt-test.

Performance Testing

Performance subtests

network

- netperf (linux and windows)
- ntttcp (windows)

block

- **iozone** (linux)
- **iozone** (windows) (iozone has its own result analysis module)
- **iometer** (windows) (not push upstream)
- **ffsb** (linux)
- **qemu_iotests** (host)
- **fio** (linux)

Environment setup

Autotest already supports prepare environment for performance testing, guest/host need to be reboot for some configuration. [setup script](#)

Autotest supports to numa pinning. Assign “numanode=-1” in tests.cfg, then vcpu threads/vhost_net threads/VM memory will be pinned to last numa node. If you want to pin other processes to numa node, you can use numactl and taskset.

```
memory: numactl -m $n $cmdline
cpu: taskset $node_mask $thread_id
```

The following content is manual guide.

```
1.First level pinning would be to use numa pinning when starting the guest.
e.g numactl -c 1 -m 1 qemu-kvm -smp 2 -m 4G <> (pinning guest memory and cpus to numa-node 1)

2.For a single instance test, it would suggest trying a one to one mapping of vcpu to physical core.
e.g
get guest vcpu threads id
#taskset -p 40 $vcpus1 (pinning vcpu1 thread to physical cpu #6 )
#taskset -p 80 $vcpus2 (pinning vcpu2 thread to physical cpu #7 )

3.To pin vhost on host. get vhost PID and then use taskset to pin it on the same socket.
e.g
taskset -p 20 $vhost (pinning vcpu2 thread to physical cpu #5 )

4.In guest, pin the IRQ to one core and the netperf to another.
1) make sure irqbalance is off - `service irqbalance stop`
2) find the interrupts - `cat /proc/interrupts`
3) find the affinity mask for the interrupt(s) - `cat /proc/irq/<irq#>/smp_affinity`
4) change the value to match the proper core. make sure the value is cpu mask.
e.g pin the IRQ to first core.
    echo 01>/proc/irq/$virq0-input/smp_affinity
    echo 01>/proc/irq/$virq0-output/smp_affinity
5) pin the netserver to another core.
e.g
taskset -p 02 netserver

5.For host to guest scenario. to get maximum performance. make sure to run netperf on different cores.
e.g
numactl -m 1 netperf -T 4 (pinning netperf to physical cpu #4)
```

Execute testing

- Submit jobs in Autotest server, only execute netperf.guest_exhost for three times.

tests.cfg:

```
only netperf.guest_exhost
variants:
    - repeat1:
    - repeat2:
    - repeat3:
# vbr0 has a static ip: 192.168.100.16
bridge=vbr0
# virbr0 is created by libvirtd, guest nic2 get ip by dhcp
bridge_nic2 = virbr0
# guest nic1 static ip
```

```
ip_nic1 = 192.168.100.21
# external host static ip:
client = 192.168.100.15
```

Result files:

```
# cd /usr/local/autotest/results/8-debug_user/192.168.122.1/
# find .|grep RHS
kvm.repeat1.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
kvm.repeat2.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
kvm.repeat3.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
```

- Submit same job in another env (different packages) with same configuration

Result files:

```
# cd /usr/local/autotest/results/9-debug_user/192.168.122.1/
# find .|grep RHS
kvm.repeat1.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
kvm.repeat2.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
kvm.repeat3.r61.virtio_blk.smp2.virtio_net.RHEL.6.1.x86_64.netperf.exhost_guest/results/netperf-resu
```

Analysis result

- Config file: perf.conf

```
[ntttcp]
result_file_pattern = *.RHS
ignore_col = 1
avg_update =

[netperf]
result_file_pattern = *.RHS
ignore_col = 2
avg_update = 4,2,3|14,5,12|15,6,13

[iozone]
result_file_pattern =
```

- Execute regression.py to compare two results:

```
login autotest server
# cd /usr/local/autotest/client/tools
# python regression.py netperf /usr/local/autotest/results/8-debug_user/192.168.122.1/ /usr/local/aut
```

- T-test:

scipy: <http://www.scipy.org/> t-test: http://en.wikipedia.org/wiki/Student's_t-test Two python modules (scipy and numpy) are needed. Script to install numpy/scipy on rhel6 automatically: <https://github.com/kongove/misc/blob/master/scripts/install-numpy-scipy.sh> Unpaired T-test is used to compare two samples, user can check p-value to know if regression bug exists. If the difference of two samples is considered to be not statistically significant($p \leq 0.05$), it will add a '+' or '-' before p-value. ('+': $\text{avg_sample1} < \text{avg_sample2}$, '-': $\text{avg_sample1} > \text{avg_sample2}$) "- only over 95% confidence results will be added "+/-" in "Significance" part. "+" for cpu-usage means regression, "+" for throughput means improvement."

Regression results

[netperf.exhost_guest.html](#) [fio.html](#) - Every Avg line represents the average value based on n repetitions of the same test, and the following SD line represents the Standard Deviation between the n repetitions. - The Standard deviation

is displayed as a percentage of the average. - The significance of the differences between the two averages is calculated using unpaired T-test that takes into account the SD of the averages. - The paired t-test is computed for the averages of same category. - only over 95% confidence results will be added “+/-” in “Significance” part. “+” for cpu-usage means regression, “+” for throughput means improvement.

Highlight HTML result o green/red -> good/bad o Significance is larger than 0.95 -> green dark green/red -> important (eg: cpu) light green/red -> other o test time o version (only when diff) o other: repeat time, title o user light green/red to highlight small (< %5) DIFF o highlight Significance with same color in one row o add doc link to result file, and describe color in doc

netperf.avg.html - Raw data that the averages are based on.

Setup a virtual environment for multi host tests

Problem:

For multi-host tests multiple physical systems are often required. It is possible to use two virtual guests for most of autotest tests except for virt-tests (kvm, libvirt, ...).

However, It is possible to use Nested Virtualization to serve as first level (L0) guests and run nested guests inside them.

This page explains, how to setup (Fedora/RHEL) and use single computer with nested virtualization for such cases. Be careful that nested virtualization works usually right, but there are cases where it might lead to hidden problems. Do not use nested virtualization for production testing.

Nested Virtualization:

1. Emulated:

- **qemu** very slow

2. hardware accelerated:

- **Hardware for the accelerated nested virtualization**

AMD Phenom and never core extension (smv, NPT) Intel Nehalem and never core extension (vmx, EPT)

- **Software which supports the accelerated nested virtualization**

kvm, xen, vmware, almost the same speed like native guest (1.0-0.1 of native guest performance). Performance depends on the type of load. IO load could be quite slow. Without vt-d or AMD-Vi and network device pass through.

Configuration for multi-host virt tests:

Config of host system

- Intel CPU

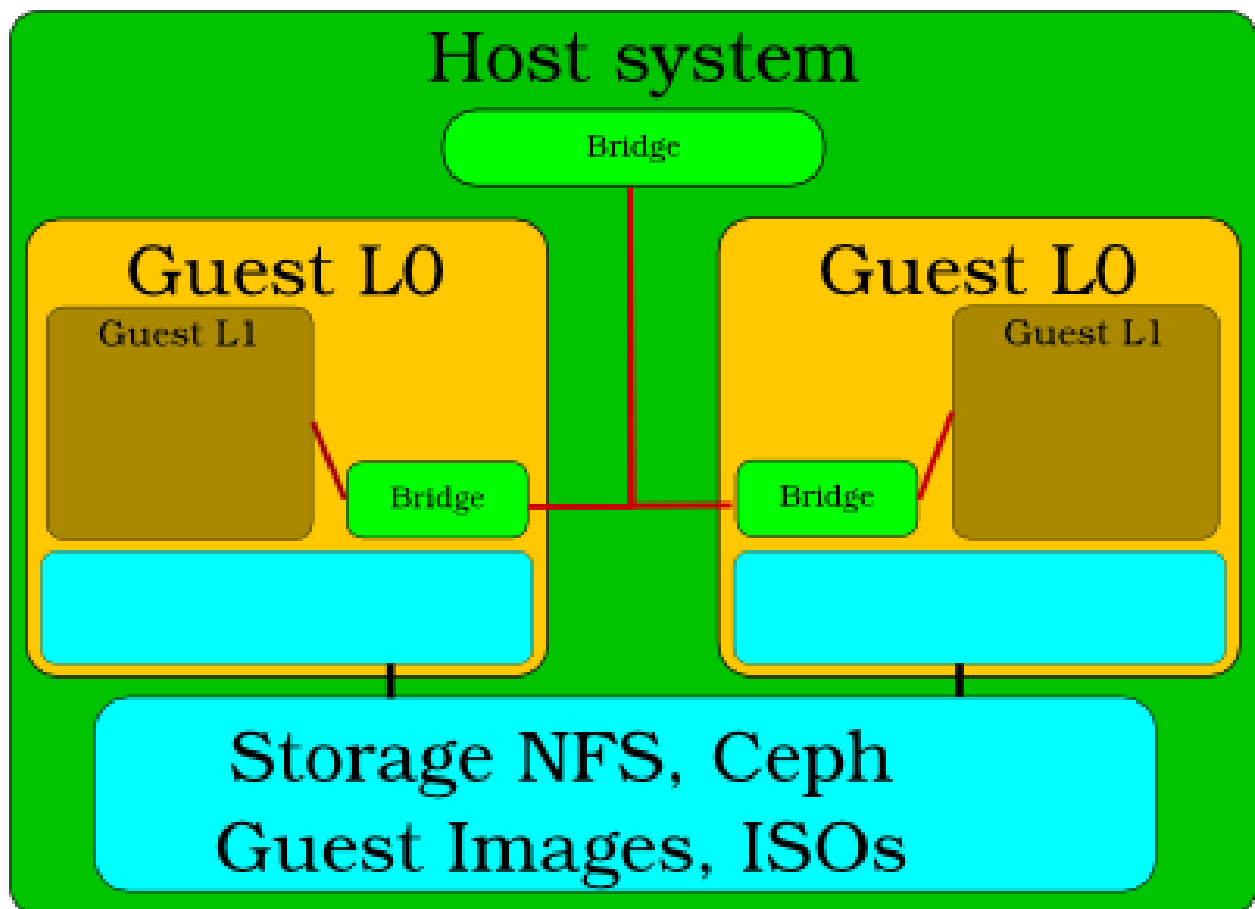
options kvm_intel nested=1 to the end of some modules config file /etc/modprobe.d/modules.conf

- AMD CPU

options kvm_amd nested=1 to the end of some modules config file /etc/modprobe.d/modules.conf

Config of Guest L0

- Intel CPU



- **Virtual manager** *Procesor config->CPU Features->vmx set to require*
- **Native qemu-kvm** *qemu-kvm -cpu core2duo,+vmx -enable-kvm*

- **AMD CPU**

- **Virtual manager** *Procesor config->CPU Features->svm set to require*
- **Native qemu-kvm** *qemu-kvm -cpu qemu64,+svm -enable-kvm*

Config of Guest L0 System Connect to host bridge with guest L0 bridge without DHCP (dhcp collision with host system dhcp).

1. Destroy libvirt bridge which contain dhcp.
2. Enable network service `systemctl enable network.service`
3. Add new bridge `virbr0` manually and insert to them L0 network adapter `eth0` which is connected to host bridge

```
#interface connected to host system bridge
vi /etc/sysconfig/network-scripts/ifcfg-eth0
    NM_CONTROLLED="no"
    DEVICE="eth0"
    ONBOOT="yes"
    BRIDGE=virbr0

#Bridge has name virbr0 for compatibility with standard autotest settings.
vi /etc/sysconfig/network-scripts/ifcfg-virbr0
    DHCP_HOSTNAME="atest-guest"
    NM_CONTROLLED="no"
    BOOTPROTO="dhcp"
    ONBOOT="yes"
    IPV6INIT="no"
    DEVICE=virbr0
    TYPE=Bridge
    DELAY=0
```

and for sure disable NetworkManager `systemctl disable NetworkManager.service`

Check Guest L0 System

`modprobe kvm-intel` or `modprobe kvm-amd` should work

Start for multi-host virt tests:

Manually from host machine

```
cd autotest/client/tests/virt/qemu/ sudo rm -rf results.*; sudo ../../../../server/autoserv -m
guestL0_1,guestL0_2 multi_host.srv
```

From autotest gui

1. Start autotest frontend RPC or WEB interface <https://github.com/autotest/autotest/wiki/SysAdmin>
2. Select multi_host test from pull-request <https://github.com/autotest/autotest-server-tests/pull/1>

More details:

Set up/configuration, the root directory (of this git repo) also has simple scripts to create L1 and L2 guests. And reference of L1, L2 libvirt files are also added – <https://github.com/kashyapc/nvmx-haswell/blob/master/SETUP-nVMX.rst>

Multi Host Migration Tests

Running Multi Host Migration Tests

virt-test is our test suite, but for simplicity purposes it can only run on a single host. For multi host tests, you'll need the full autotest + virt-test package, and the procedure is more complex. We'll try to keep this procedure as objective as possible.

Prerequisites

This guide assumes that:

1. You have at least 2 virt capable machines that have shared storage setup in [insert specific path]. Let's call them `host1.foo.com` and `host2.foo.com`.
2. You can ssh into both of those machines without a password (which means there is an SSH key setup with the account you're going to use to run the tests) as root.
3. The machines should be able to communicate freely, so beware of the potential firewall complications. On each of those machines you need a specific NFS mount setup:
 - `/var/lib/virt_test/isos`
 - `/var/lib/virt_test/steps_data`
 - `/var/lib/virt_test/gpg`

They all need to be backed by an NFS share read only. Why read only? Because it is safer, we exclude the chance to delete this important data by accident. Besides the data above is only needed in a read only fashion. `fstab` example:

```
myserver.foo.com:/virt-test/iso /var/lib/virt_test/isos nfs ro,nosuid,nodev,noatime,intr,hard,tcp 0 0
myserver.foo.com:/virt-test/steps_data /var/lib/virt_test/steps_data nfs rw,nosuid,nodev,noatime,intr,hard,tcp 0 0
myserver.foo.com:/virt-test/gpg /var/lib/virt_test/gpg nfs rw,nosuid,nodev,noatime,intr,hard,tcp 0 0
```

- `/var/lib/virt_test/images`
- `/var/lib/virt_test/images_archive`

Those all need to be backed by an NFS share read write (or any other shared storage you might have). This is necessary because both hosts need to see the same coherent storage. `fstab` example:

```
myserver.foo.com:/virt-test/images_archive /var/lib/virt_test/images_archive nfs rw,nosuid,nodev,noatime,intr,hard,tcp 0 0
myserver.foo.com:/virt-test/images /var/lib/virt_test/images nfs rw,nosuid,nodev,noatime,intr,hard,tcp 0 0
```

The images dir must be populated with the installed guests you want to run your tests on. They must match the file names used by guest OS in virt-test. For example, for RHEL 6.4, the image name virt-test uses is:

```
rhel64-64.qcow2
```

double check your files are there:

```
$ ls /var/lib/virt_test/images
$ rhel64-64.qcow2
```

Setup step by step

First, clone the autotest repo recursively. It's a repo with lots of submodules, so you'll see a lot of output:

```
$ git clone --recursive https://github.com/autotest/autotest.git
... lots of output ...
```

Then, edit the `global_config.ini` file, and change the key:

```
serve_packages_from_autoserv: True
```

to:

```
serve_packages_from_autoserv: False
```

Then you need to update `virt-test`'s config files and sub tests (that live in separate repositories that are not git submodules). You don't need to download the JeOS file in this step, so simply answer 'n' to the quest

Note: The bootstrap procedure described below will be performed automatically upon running the `autoserv` command that triggers the test. The problem is that then you will not be able to see the config files and modify filters prior to actually running the test. Therefore this documentation will instruct you to run the steps below manually.

```
$ export AUTOTEST_PATH=.;client/tests/virt/run -t qemu --bootstrap --update-providers
16:11:14 INFO | qemu test config helper
16:11:14 INFO |
16:11:14 INFO | 1 - Updating all test providers
16:11:14 INFO | Fetching git [REP 'git://github.com/autotest/tp-qemu.git' BRANCH 'master'] -> /var/tr
16:11:17 INFO | git commit ID is 6046958afalccab7f22bblala73347d9c6ed3211 (no tag found)
16:11:17 INFO | Fetching git [REP 'git://github.com/autotest/tp-libvirt.git' BRANCH 'master'] -> /var
16:11:19 INFO | git commit ID is edc07c0c4346f9029930b062c573ff6f5433bc53 (no tag found)
16:11:20 INFO |
16:11:20 INFO | 2 - Checking the mandatory programs and headers
16:11:20 INFO | /usr/bin/7za
16:11:20 INFO | /usr/sbin/tcpdump
16:11:20 INFO | /usr/bin/nc
16:11:20 INFO | /sbin/ip
16:11:20 INFO | /sbin/arping
16:11:20 INFO | /usr/bin/gcc
16:11:20 INFO | /usr/include/bits/unistd.h
16:11:20 INFO | /usr/include/bits/socket.h
16:11:20 INFO | /usr/include/bits/types.h
16:11:20 INFO | /usr/include/python2.6/Python.h
16:11:20 INFO |
16:11:20 INFO | 3 - Checking the recommended programs
16:11:20 INFO | Recommended command missing. You may want to install it if not building it from sour
16:11:20 INFO | Recommended command qemu-img missing. You may want to install it if not building from
16:11:20 INFO | Recommended command qemu-io missing. You may want to install it if not building from
16:11:20 INFO |
16:11:20 INFO | 4 - Verifying directories
16:11:20 INFO |
16:11:20 INFO | 5 - Generating config set
16:11:20 INFO |
16:11:20 INFO | 6 - Verifying (and possibly downloading) guest image
16:11:20 INFO | File JeOS 19 x86_64 not present. Do you want to download it? (y/n) n
16:11:30 INFO |
16:11:30 INFO | 7 - Checking for modules kvm, kvm-amd
16:11:30 WARNI| Module kvm is not loaded. You might want to load it
16:11:30 WARNI| Module kvm-amd is not loaded. You might want to load it
16:11:30 INFO |
16:11:30 INFO | 8 - If you wish, take a look at the online docs for more info
16:11:30 INFO |
16:11:30 INFO | https://github.com/autotest/virt-test/wiki/GetStarted
```

Then you need to copy the multihost config file to the appropriate place:

```
cp client/tests/virt/test-providers.d/downloads/io-github-autotest-qemu/qemu/cfg/multi-host-tests.cfg
```

Now, edit the file:

```
server/tests/multihost_migration/control.srv
```

In there, you have to change the EXTRA_PARAMS to restrict the number of guests you want to run the tests on. On this example, we're going to restrict our tests to RHEL 6.4. The particular section of the control file should look like:

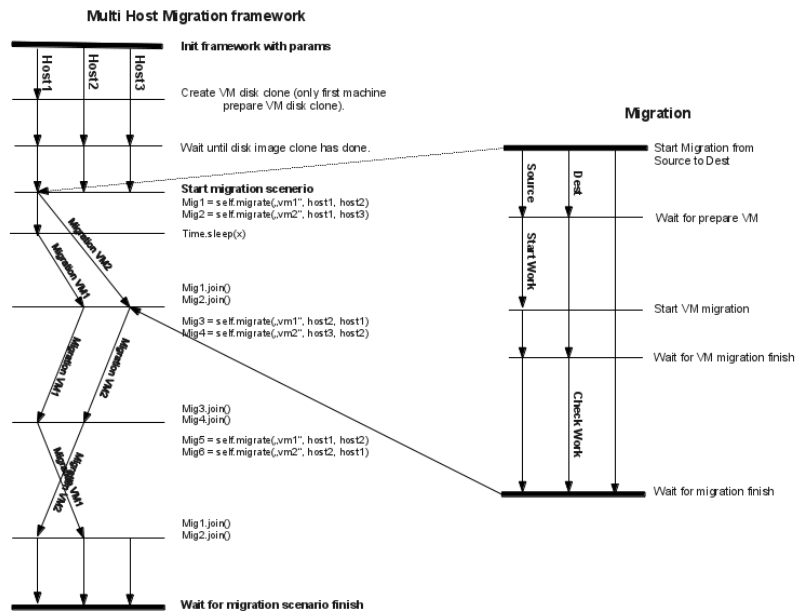
```
EXTRA_PARAMS = """
only RHEL.6.4.x86_64
"""
```

It is important to stress that the guests must be installed for this to work smoothly. Then the last step would be to run the tests. Using the same convention for the machine hostnames, here's the command you should use:

```
server/autotest-remote -m host1.foo.com,host2.foo.com server/tests/multihost_migration/control.srv
```

Now, you'll see a boatload of output from the autotest remote output. This is normal, and you should be patient until all the tests are done.

Writing Multi Host Migration tests



Scheme: Source file for the diagram above (LibreOffice file)

Example:

```
class TestMultihostMigration(virt_utils.MultihostMigration):
    def __init__(self, test, params, env):
        super(testMultihostMigration, self).__init__(test, params, env)
```

```

def migration_scenario(self):
    srchost = self.params.get("hosts")[0]
    dsthost = self.params.get("hosts")[1]

    def worker(mig_data):
        vm = env.get_vm("vm1")
        session = vm.wait_for_login(timeout=self.login_timeout)
        session.sendline("nohup dd if=/dev/zero of=/dev/null &")
        session.cmd("killall -0 dd")

    def check_worker(mig_data):
        vm = env.get_vm("vm1")
        session = vm.wait_for_login(timeout=self.login_timeout)
        session.cmd("killall -9 dd")

    # Almost synchronized migration, waiting to end it.
    # Work is started only on first VM.

    self.migrate_wait(["vm1", "vm2"], srchost, dsthost,
                      worker, check_worker)

    # Migration started in different threads.
    # It allows to start multiple migrations simultaneously.

    # Starts one migration without synchronization with work.
    mig1 = self.migrate(["vm1"], srchost, dsthost,
                        worker, check_worker)

    time.sleep(20)

    # Starts another test simultaneously.
    mig2 = self.migrate(["vm2"], srchost, dsthost)
    # Wait for mig2 finish.
    mig2.join()
    mig1.join()

mig = TestMultihostMigration(test, params, env)
# Start test.
mig.run()

```

When you call:

```

mig = TestMultihostMigration(test, params, env):

```

What happens is

1. VM's disks will be prepared.
2. The synchronization server will be started.
3. All hosts will be synchronized after VM create disks.

When you call the method:

```

migrate():

```

What happens in a diagram is:

source	destination
It prepare VM if machine is not started.	
Start work on VM.	
<code>mig.migrate_vms_src()</code>	<code>mig.migrate_vms_dest()</code>
Check work on VM after migration.	
Wait for finish migration on all hosts.	

It's important to note that the migrations are made using the `tcp` protocol, since the others don't support multi host migration.

```
def migrate_vms_src(self, mig_data):
    vm = mig_data.vms[0]
    logging.info("Start migrating now...")
    vm.migrate(mig_data.dst, mig_data.vm_ports)
```

This example migrates only the first machine defined in migration. Better example is in `virt_utils.MultihostMigration.migrate_vms_src`. This function migrates all machines defined for migration.

Cartesian Config

Reference documentation of the cartesian config format.

Contents:

Parameters

The test configuration file is used for controlling the framework by specifying parameters for each test. The parser produces a list of dictionaries (see an explanation of the file format?), each of which specifies the parameters for a single test. Each parameter is used (read) by the test dispatching system, by the pre-processor, by the post-processor, or by the test itself.

Some parameters are required and others are optional.

Most parameters should be specified in the test configuration file by the user. Few parameters are produced automatically by the configuration file parser, so when using the parser, these must not be specified by the user. Recent `kvm-autotest` support passing some basic parameters through the command line?.

All parameters are strings, except `depend?`, which is a list of strings. `depend?` is automatically generated by the parser, so the user probably should not be concerned with it.

You may also want to check the complete [reference documentation](#) on parameters.

Addressing objects (VMs, images, NICs etc)

Before listing the parameters, it is important to clarify how objects like VMs should be addressed in the test parameters.

For the following example we shall assume that our system accepts a parameter `vms?` which lists the VM objects to be used in the current test. Typical usage of the parameter would be:

```
vms = vm1 second_vm another_vm
```

This would indicate that our test requires 3 VMs. Let us now assume that a VM object accepts a parameter `mem` which specifies the amount of memory to give the VM. In order to specify `mem` for **vm1**, we may write:

```
mem_vm1 = 512
```


and in order to specify it for **second_vm** we may write:

```
mem_second_vm = 1024
```

If we wanted to specify `mem` for all existing VM objects, we would write:

```
mem = 128
```

However, this would only apply to **another_vm**, because the previous statements, which each specify `mem` for a single VM, override the statement that specifies `mem` for all VMs. The order in which these statements are written in a configuration file is not important; statements addressing a single object always override statements addressing all objects.

Let us now further assume that a VM object accepts a parameter `images`, which lists the disk image objects to be used by the VM. Typical usage of `images`, with regard to **vm1**, would be:

```
images_vm1 = first_image image2 a_third_image yet_another_image
```

We shall also assume that an image object accepts two parameters: `image_name`, which specifies the filename of the disk image, and `image_size`, which specifies the size of the image (e.g. 10G). In order to specify these with regard to **first_image**, which is the first image of **vm1**, we may write:

```
image_name_first_image_vm1 = fc8-32-no-acpi
image_size_first_image_vm1 = 20G
```

Note the order in which the objects are addressed: first the parameter, then the image, then the VM. In order to specify these parameters for all images of **vm1**, we may write:

```
image_name_vm1 = fc8-32
image_size_vm1 = 10G
```

However, these statements would not apply to **first_image** of **vm1**, because the previous statements, which addressed this image specifically, override the statements that address all objects. If we chose to specify these parameters for all images of all VMs, we would write:

```
image_name = fc8-32-something
image_size = 5G
```

However, these would not apply to the images of **vm1**, because previous statements apply specifically to those images.

Parameters used by the test dispatching system

The test dispatching system consists of the control file and the framework's main python module (currently named `kvm_runtest_2.py`). This system executes the proper test according to the supplied parameters.

Parameter	Effect/meaning	Required?
type?	Specifies the type of test to run (e.g. boot, migration etc)	yes
skip?	If equals 'yes', the test will not be executed	no
name?	The full name (not type) of the test (e.g. qcow2.ide.Fedora.8.32.install); see test configuration file format?. This parameter is generated by the parser.	yes
short-name?	The short name of the test (e.g. Fedora.8.32.install); see test configuration file format?. This parameter is generated by the parser. It specifies the tag to append to the Autotest test name, so that eventually the test name becomes something like kvm_runtest_2.Fedora.8.32.install.	yes
depend?	The full names of the dependencies of this test (e.g. ['qcow2.openSUSE-11.install', 'openSUSE-11.boot']). This parameter is a list of strings, not a string, and is generated by the parser. The test dispatcher will not run a test if one or more of its dependencies have run and failed.	yes

Parameters used by the preprocessor

The preprocessor runs before the test itself. It prepares VMs and images for the test, according to the supplied parameters.

Parameter	Effect/meaning	Required?
vms?	Lists the VM objects to be used in the test. Listed VMs that do not exist will be created. Existing VMs that are not listed will be destroyed and removed.	yes

VM preprocessor parameters These parameters should be specified for each VM as explained above in addressing objects.

Parameter	Effect/meaning	Required?
start_vm?	If equals 'yes', the VM will be started if it is down ; this parameter should be set to 'yes', for a certain VM object, in all tests that require the VM to be up.	no
restart_vm?	If equals 'yes', the VM will be (re)started, regardless of whether it's already up or not	no
start_vm_for_migration?	If equals 'yes', the VM will be (re)started with the -incoming option so that it accepts incoming migrations; this parameter should be set to 'yes' for the destination VM object in a migration test.	no

The following parameters are remembered by a VM object when it is created or started. They cannot be changed while a VM is up. In order to change them, the VM must be restarted with new parameters.

Parameter	Effect/meaning	Required (when creating or starting a VM)
cdrom?	Specifies the name of an image file to be passed to QEMU with the -cdrom option. This is typically an ISO image file.	no
md5sum?	If specified, the VM will not be started (and thus the test will fail) if the MD5 sum of the cdrom image doesn't match this parameter. This is intended to verify the identity of the image used.	no
md5sum1	Similar to md5sum , but specifies the MD5 sum of only the first MB of the cdrom image. If specified, this parameter is used instead of md5sum . Calculating the MD5 sum of the first MB of an image is much quicker than calculating it for the entire image.	no
mem	Specifies the amount of memory, in MB, the VM should have	yes
display	Selects the rendering method to be used by the VM; valid values are 'vnc', 'sdl' and 'nographic'. If 'vnc' is selected, the VM will be assigned an available VNC port automatically.	no
extra_params?	Specifies a string to append to the QEMU command line, e.g. '-snapshot'	no
use_telnet?	If equals 'yes', communication with the guest will be done via Telnet; otherwise SSH will be used.	no
ssh_port?	Specifies the guest's SSH/Telnet port; should normally be 22, unless Telnet is used, in which case this parameter should be 23.	if the VM should support SSH/Telnet communication
ssh_prompt?	A regular expression describing the guest's shell prompt	if the VM should support SSH/Telnet communication
user-name?	Specifies the username with which to attempt to log into the guest whenever necessary	if the VM should support SSH/Telnet communication
pass-word?	Specifies the password with which to attempt to log into the guest whenever necessary	if the VM should support SSH/Telnet communication
cmd_shutdown	Specifies the shell command to be used to shut the guest down (via SSH/Telnet) whenever necessary	if the VM should support being shutdown via SSH/Telnet
cmd_reboot	Specifies the shell command to be used to reboot the guest (via SSH/Telnet) whenever necessary	if the VM should support rebooting via SSH/Telnet
images	Lists the image objects to be used by the VM	yes
nics	Lists the NIC objects to be used by the VM	yes

A VM will be restarted automatically if a parameter change leads to a different QEMU command line (for example, when **mem** changes). However, when other parameters change (such as **cmd_shutdown**) the VM will not be automatically restarted (unless **restart_vm** is set to 'yes'), and the change will have no effect.

Image preprocessor parameters The following parameters should be specified for each image of each VM, as explained in addressing objects.

Parameter	Effect/meaning	Required?
create_image	If equals 'yes', the image file will be created using qemu-img if it doesn't already exist	no
force_create_image	If equals 'yes', the image file will be created using qemu-img regardless of whether it already exists. If the file already exists it will be overwritten by a blank image file.	no
image_name	Specifies the image filename without the extension	yes
image_format	Specifies the format of the image to be created/used, e.g. qcow2, raw, vmdk etc	yes
image_size	Specifies the size of the image to be created, in a format understood by qemu-img (e.g. 10G)	only when creating an image
drive_format	Specifies a string to pass to QEMU as the drive's 'if' parameter (e.g. ide, scsi)	no
image_snapshot	If equals 'yes', 'snapshot=on' will be appended to the 'drive' option passed to QEMU	no
image_boot?	If equals 'yes', 'boot=on' will be appended to the 'drive' option passed to QEMU	no

NIC preprocessor parameters The following parameters should be specified for each NIC of each VM, as explained in the section “addressing objects”.

Parameter	Effect/meaning	Required?
nic_model?	A string to pass to QEMU as the NIC's 'model' parameter (e.g. e1000, virtio)	no

Parameters used by the postprocessor

The postprocessor runs after the test itself. It can shut down VMs, remove image files and clean up the test's results dir.

The suffix **_on_error** may be added to all parameters in this section (including VM and image parameters) to define special behavior for tests that fail or result in an error. The suffix should be added **after** all object addressing suffixes. If a parameter is specified without the suffix, it applies both when the test passes and when it fails. If a parameter is specified with the suffix, it applies only when the test fails, and overrides the parameter without the suffix.

For example, if we wanted the postprocessor to shut down **vm1** after the test, but only if the test failed, we'd write:

```
kill_vm_vm1_on_error = yes
```

If we wanted to shut down **another_vm** only if the test **passed**, we'd write:

```
kill_vm_another_vm = yes
kill_vm_another_vm_on_error = no
```

Since PPM files are normally used for debugging test failures, it would be very reasonable to choose to keep them only if the test fails. In that case we'd write:

```
keep_ppm_files = no
keep_ppm_files_on_error = yes
```

The following parameters define the postprocessor's behavior:

Parameter	Effect/meaning	Re-quired?
vms?	Lists the VM objects to be handled by the postprocessor	yes
keep_ppm_files	If equals 'yes', the PPM image files in the test's debug directory will not be removed	no

VM postprocessor parameters These parameters should be specified for each VM as explained above in “addressing objects”.

Parameter	Effect/meaning	Re-quired?
kill_vm	If equals 'yes', the VM will be shut down after the test	no
kill_vm_gracefully	If equals 'yes', and kill_vm equals 'yes', the first attempt to kill the VM will be done via SSH/Telnet with a clean shutdown command (rather than a quick 'quit' monitor command)	no
kill_vm_timeout	If kill_vm equals 'yes', this parameter specifies the time duration (in seconds) to wait for the VM to shut itself down, before attempting to shut it down externally; if this parameter isn't specified the VM killing procedure will start immediately following the test. This parameter is useful for tests that instruct a VM to shut down internally and need the postprocessor to shut it down only if it fails to shut itself down in a given amount of time	no
images	Lists the images objects, for this VM, to be handled by the postprocessor	no

Image postprocessor parameters These parameters should be specified for each image of each VM as explained above in “addressing objects”.

Parameter	Effect/meaning	Required?
remove_image	If equals 'yes', the image file will be removed after the test	no

Test parameters

Any number of additional parameters may be specified for each test, and they will be available for the test to use. See the tests? page for a list of tests and the parameters they use.

Real world example

The following example dictionary is taken from a dictionary list used in actual tests. The list was generated by the config file parser.

```
Dictionary #363:
  cmd_reboot = shutdown -r now
  cmd_shutdown = shutdown -h now
  depend = ['custom.qcow2.ide.default.up.Linux.Fedora.9.32.e1000.install',
            'custom.qcow2.ide.default.up.Linux.Fedora.9.32.e1000.setup',
            'custom.qcow2.ide.default.up.Linux.Fedora.9.32.default_nic.install',
            'custom.qcow2.ide.default.up.Linux.Fedora.9.32.default_nic.setup']
  drive_format = ide
  image_boot = yes
  image_format = qcow2
  image_name = fc9-32
  image_size = 10G
  images = image1
  keep_ppm_files = no
  keep_ppm_files_on_error = yes
  kill_vm = no
```

```
kill_vm_gracefully = yes
kill_vm_on_error = yes
main_vm = vm1
mem = 512
migration_dst = dst
migration_src = vm1
migration_test_command = help
name = custom.qcow2.ide.default.up.Linux.Fedora.9.32.e1000.migrate.1
nic_model = e1000
nics = nic1
password = 123456
shortname = Fedora.9.32.e1000.migrate.1
ssh_port = 22
ssh_prompt = \[root@.{0,50}][\#\$]
start_vm = yes
start_vm_for_migration_dst = yes
type = migration
username = root
vms = vm1 dst
```

The test dispatching system This test's **name** is a rather long string that indicates all the variants this test belongs to; its **shortname**, however, is much shorter: **Fedora.9.32.e1000.migrate.1**.

The test depends on 4 other tests, as indicated by the `depend?` parameter. The listed strings are the **names** of these tests. If any of these 4 tests runs and fails, the current test will be skipped.

Preprocessing This test requires two VMs as indicated by the `vms?` parameter: one will be called **vm1** and the other **dst**. The parameter **start_vm**, which lacks a VM suffix and therefore applies to both VMs, indicates that if any of these VM objects does not exist or is not up, it will be started. However, **start_vm_for_migration_dst = yes** indicates that the VM **dst** should be started with the `-incoming` option so that it accepts an incoming migration.

The `images` parameter indicates that a single image object will be used by each VM, and they will both be called **image1**. This poses no problem because an image object only exists within the scope of its owner VM. However, both image objects actually point to the same image file, as indicated by **image_name = fc9-32**. If `image_name` appeared with some suffix (e.g. **image_name_image1_vm1** or **image_name_vm1**) it would be attributed to a single VM, not both. **image_format = qcow2** indicates that this is a qcow2 image file, so the actual filename becomes `fc9-32.qcow2`. **image_boot = yes** instructs the preprocessor to add `'boot=on'` to the `-drive` option in the QEMU command line. **drive_format = ide** adds `'if=ide'`. No image file is created during the preprocessing phase of this test because both **create_image** and **force_create_image** are not specified.

The `nics` parameter indicates that each VM should be equipped with a single NIC object named **nic1**. **nic_model = e1000** indicates that all NICs (due to the lack of a suffix) should be of the `e1000` model. If one wished to specify a different NIC model for each VM, one could specify, for example, **nic_model_vm1 = e1000** and **nic_model_dst = rtl8139**.

The parameters `mem`, `ssh_port?`, `ssh_prompt?`, `username?`, `password?`, `cmd_reboot?` and `cmd_shutdown?` apply to both VMs. See #VM preprocessor parameters for an explanation of these parameters.

The test itself The parameters `migration_src?`, `migration_dst?` and `migration_test_command?` are used by the migration test. They instruct it to migrate from **vm1** to **dst** and use the shell command **help** to test that the VM is alive following the migration.

The parameter **main_vm** happens to be specified because the format of the configuration file makes it easy to set a parameter for a large group of tests. However, in the case of a migration test, this parameter is not used and its presence is harmless.

Postprocessing `keep_ppm_files = no` and `keep_ppm_files_on_error = yes?` indicate that normally the PPM files (images left in the test's 'debug' directory) will not be kept; however, if the test fails, they will. This makes sense because the PPM files take quite a lot of hard drive space, and they are mostly useful to debug failures.

`kill_vm = no` indicates that normally both VMs should be left alone following the test.

`kill_vm_on_error = yes?` indicates that in the case of a failure, both VMs should be destroyed. This makes sense because if a migration test fails, the VMs involved may not be functional for the next test, thus causing it to fail.

If they are killed by the postprocessor, the preprocessor of the next test will automatically start them, assuming `start_vm = yes?` is specified for the next test. The parameter `kill_vm_gracefully` indicates that if a VM is to be killed, it should first be attempted via SSH/Telnet with a shutdown shell command, specified by the `cmd_shutdown?` parameter.

Cartesian Config Reference

bridge

Description Sets the name of the bridge to which a VM nic will be added to. This only applies to scenarios where 'nic_mode' is set to 'tap'.

It can be set as a default to all nics:

```
bridge = virbr0
```

Or to a specific nic, by prefixing the parameter key with the nic name, that is for attaching 'nic1' to bridge 'virbr1':

```
bridge_nic1 = virbr1
```

Defined On

- [client/tests/kvm/tests_base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)
- [client/virt/virt_utils.py](#)

Referenced By No other documentation currently references this configuration key.

cdroms

Description Sets the list of cdrom devices that a VM will have.

Usually a VM will start with a single cdrom, named 'cd1'.

```
cdroms = cd1
```

But a VM can have other cdroms such as 'unattended' for unattended installs:

```
variants:
  - @Linux:
      unattended_install:
          cdroms += " unattended"
```

And ‘winutils’ for Microsoft Windows VMs:

```
variants:
  - @Windows:
      unattended_install.cdrom, whql.support_vm_install:
          cdroms += " winutils"
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)
- [client/tests/kvm/virtio-win.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

cd_format

Description Sets the format for a given cdrom drive. This directive exists to do some special magic for cd drive formats ‘ahci’ and ‘usb2’ (see [client/virt/kvm_vm.py](#) for more information).

Currently used options in virt-test are: ahci and usb2.

Example:

```
variants:
  - usb.cdrom:
      cd_format = usb2
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See also

- [drive_format](#)

check_image

Description Configures if we want to run a check on the image files during post processing. A check usually means running ‘qemu-img info’ and ‘qemu-img check’.

This is currently only enabled when image_format is set to ‘qcow2’.

```
variants:
  - @qcow2:
      image_format = qcow2
      check_image = yes
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [images](#)
- [image_name](#)
- [image_format](#)
- [create_image](#)
- [remove_image](#)

convert_ppm_files_to_png

Description Configures whether files generated from screenshots in [PPM format](#) should be automatically converted to [PNG](#) files.

```
convert_ppm_files_to_png = yes
```

Usually we’re only interested in spending time converting files for easier viewing on situations with failures:

```
convert_ppm_files_to_png_on_error = yes
```

Defined On The stock configuration key (without suffix) is not currently defined on any sample cartesian configuration file.

The configuration key with the ‘on_error’ suffix is defined on:

- [client/tests/kvm/base.cfg.sample](#)

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

create_image

Description Configures if we want to create an image file during pre processing, if it does **not** already exists. To force the creation of the image file even if it already exists, use `force_create_image`.

To create an image file if it does **not** already exists:

```
create_image = yes
```

Defined On

- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/tests/kvm/tests/qemu_img.py`
- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `images`
- `image_name`
- `image_format`
- `create_image`
- `force_create_image`
- `remove_image`

display

Description Sets the VM display type. Of course, only one display type is allowed, and current valid options are: `vnc`, `sdl`, `spice` and `nographic`.

```
display = vnc
```

For VNC displays, the port number is dynamically allocated within the 5900 - 6100 range.

```
display = sdl
```

An SDL display does not use a port, but simply behaves as an X client. If you want to send the SDL display to a different X Server, see `x11_display`?

```
display = spice
```

For spice displays, the port number is dynamically allocated within the 8000 - 8100 range.

```
display = nographic
```

nographic for qemu/kvm means that the VM will have no graphical display and that serial I/Os will be redirected to console.

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

drive_cache

Description Sets the caching mode a given drive. Currently the valid values are: writethrough, writeback, none and unsafe.

Example:

```
drive_cache = writeback
```

This option can also be set specifically to a drive:

```
drive_cache_cdl = none
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

drive_format

Description Sets the format for a given drive.

Usually this passed directly to qemu 'if' sub-option of '-drive' command line option. But in some special cases, such as when drive_format is set to 'ahci' or 'usb2', some special magic happens (see [client/virt/kvm_vm.py](#) for more information).

Currently available options in qemu include: ide, scsi, sd, mtd, floppy, pflash, virtio.

Currently used options in virt-test are: ide, scsi, virtio, ahci, usb2.

Example:

```
drive_format = ide
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/virt/kvm_vm.py`

Referenced By No other documentation currently references this configuration key.

drive_index

Description Sets the index, that is, ordering precedence of a given drive. Valid values are integers starting with 0.

Example:

```
drive_index_image1 = 0
drive_index_cd1 = 1
```

This will make the drive that has 'image1' appear before the drive that has 'cd1'.

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/guest-os.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`
- `client/tests/kvm/virtio-win.cfg.sample`

Used By

- `client/virt/kvm_vm.py`

Referenced By No other documentation currently references this configuration key.

drive_werror

Description Sets the behavior for the VM when a drive encounters a read or write error. This is passed to QEMU 'werror' sub-option of the '-drive' command line option.

Valid for QEMU are: ignore, stop, report, enospc.

Example:

```
drive_werror = stop
```

Defined On

- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

drive_serial

Description Sets the serial number to assign to the drive device.

Defined On This configuration key is not currently defined on any sample cartesian configuration file.

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

file_transfer_client

Description Sets the kind of application, thus protocol, that will be spoken when transferring files to and from the guest.

virt-test currently allows for two options: 'scp' or 'rss'.

For Linux VMs, we default to SSH:

```
variants:
  - @Linux:
      file_transfer_client = scp
```

And for Microsoft Windows VMs we default to rss:

```
variants:
  - @Windows:
      file_transfer_client = rss
```

Defined On

- [client/tests/kvm/guest-os.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [redirs](#)
- [file_transfer_port](#)
- [guest_port_file_transfer](#)

file_transfer_port

Description Sets the port on which the application used to transfer files to and from the guest will be listening on.

When `file_transfer_client` is `scp`, this is by default 22:

```
variants:
  - @Linux:
      file_transfer_client = scp
      file_transfer_port = 22
```

And for `rss`, the default is port 10023:

```
variants:
  - @Windows:
      file_transfer_client = rss
      file_transfer_port = 10023:
```

Defined On

- [client/tests/kvm/guest-os.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [redirs](#)
- [file_transfer_client](#)
- [guest_port_file_transfer](#)

force_create_image

Description Configures if we want to create an image file during pre processing, **even if it already exists**. To create an image file only if it **does not** exist, use `create_image` instead.

To create an image file **even if it already exists**:

```
force_create_image = yes
```

Defined On

- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [images](#)
- [image_name](#)
- [image_format](#)
- [create_image](#)
- [check_image](#)
- [remove_image](#)

guest_port

Description `guest_port` is not a configuration item itself, but the basis (prefix) of other real configuration items such as:

- [guest_port_remote_shell](#)
- [guest_port_file_transfer](#)
- [guest_port_unattended_install](#)

Defined On Variations of `guest_port` are defined on the following files:

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)

Referenced By No other documentation currently references this configuration key.

See also

- [redirs](#)

guest_port_remote_shell

Description Sets the port of the remote shell server that runs inside guests. On Linux VMs, this is the set by default to the standard SSH port (22), and for Windows guests, set by default to port 10022.

This is a specialization of the `guest_port` configuration entry.

Example, default entry:

```
guest_port_remote_shell = 22
```

Overridden on Windows variants:

```
variants:
  - @Windows:
      guest_port_remote_shell = 10022
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/guest-os.cfg.sample`

Used By

- `client/virt/kvm_vm.py`
- `client/virt/virt_utils.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `redirs`
- `shell_port?`

`guest_port_file_transfer`

Description Sets the port of the server application running inside guests that will be used for transferring files to and from this guest.

On Linux VMs, the `file_transfer_client` is set by default to 'scp', and this the port is set by default to the standard SSH port (22).

For Windows guests, the `file_transfer_client` is set by default to 'rss', and the port is set by default to 10023.

This is a specialization of the `guest_port` configuration entry.

Example, default entry:

```
guest_port_file_transfer = 22
```

Overridden on Windows variants:

```
variants:
  - @Windows:
      guest_port_file_transfer = 10023
```

Defined On

- `client/tests/kvm/guest-os.cfg.sample`

Used By

- [client/virt/kvm_vm.py](#)
- [client/virt/virt_utils.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [redirs](#)
- [file_transfer_port](#)
- [file_transfer_client](#)

guest_port_unattended_install

Description Sets the port of the helper application/script running inside guests that will be used for flagging the end of the unattended install.

Both on Linux and Windows VMs, the default value is 12323:

```
guest_port_unattended_install = 12323
```

This must match with the port number on unattended install files. On Linux VMs, this is hardcoded on kickstart files ‘%post’ section:

```
%post --interpreter /usr/bin/python
...
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('', 12323))
server.listen(1)
(client, addr) = server.accept()
client.send("done")
client.close()
```

This is a specialization of the `guest_port` configuration entry.

Defined On

- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/tests/kvm/tests/unattended_install.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [redirs](#)

images

Description Sets the list of disk devices (backed by a image file or device) that a VM will have.

Usually a VM will start with a single image, named `image1`:

```
images = image1
```

But a VM can have other images. One example is when we test the maximum number of disk devices supported on a VM:

```
# Tests
variants:
  - multi_disk: install setup image_copy unattended_install.cdrom
    variants:
      - max_disk:
          images += " stg stg2 stg3 stg4 stg5 stg6 stg7 stg8 stg9 stg10 stg11 stg12 stg13 stg14 stg15 stg16 stg17 stg18 stg19 stg20 stg21 stg22 stg23 stg24 stg25 stg26 stg27 stg28 stg29 stg30 stg31 stg32 stg33 stg34 stg35 stg36 stg37 stg38 stg39 stg40 stg41 stg42 stg43 stg44 stg45 stg46 stg47 stg48 stg49 stg50 stg51 stg52 stg53 stg54 stg55 stg56 stg57 stg58 stg59 stg60 stg61 stg62 stg63 stg64 stg65 stg66 stg67 stg68 stg69 stg70 stg71 stg72 stg73 stg74 stg75 stg76 stg77 stg78 stg79 stg80 stg81 stg82 stg83 stg84 stg85 stg86 stg87 stg88 stg89 stg90 stg91 stg92 stg93 stg94 stg95 stg96 stg97 stg98 stg99 stg100 stg101 stg102 stg103 stg104 stg105 stg106 stg107 stg108 stg109 stg110 stg111 stg112 stg113 stg114 stg115 stg116 stg117 stg118 stg119 stg120 stg121 stg122 stg123 stg124 stg125 stg126 stg127 stg128 stg129 stg130 stg131 stg132 stg133 stg134 stg135 stg136 stg137 stg138 stg139 stg140 stg141 stg142 stg143 stg144 stg145 stg146 stg147 stg148 stg149 stg150 stg151 stg152 stg153 stg154 stg155 stg156 stg157 stg158 stg159 stg160 stg161 stg162 stg163 stg164 stg165 stg166 stg167 stg168 stg169 stg170 stg171 stg172 stg173 stg174 stg175 stg176 stg177 stg178 stg179 stg180 stg181 stg182 stg183 stg184 stg185 stg186 stg187 stg188 stg189 stg190 stg191 stg192 stg193 stg194 stg195 stg196 stg197 stg198 stg199 stg200 stg201 stg202 stg203 stg204 stg205 stg206 stg207 stg208 stg209 stg210 stg211 stg212 stg213 stg214 stg215 stg216 stg217 stg218 stg219 stg220 stg221 stg222 stg223 stg224 stg225 stg226 stg227 stg228 stg229 stg230 stg231 stg232 stg233 stg234 stg235 stg236 stg237 stg238 stg239 stg240 stg241 stg242 stg243 stg244 stg245 stg246 stg247 stg248 stg249 stg250 stg251 stg252 stg253 stg254 stg255 stg256 stg257 stg258 stg259 stg260 stg261 stg262 stg263 stg264 stg265 stg266 stg267 stg268 stg269 stg270 stg271 stg272 stg273 stg274 stg275 stg276 stg277 stg278 stg279 stg280 stg281 stg282 stg283 stg284 stg285 stg286 stg287 stg288 stg289 stg290 stg291 stg292 stg293 stg294 stg295 stg296 stg297 stg298 stg299 stg300 stg301 stg302 stg303 stg304 stg305 stg306 stg307 stg308 stg309 stg310 stg311 stg312 stg313 stg314 stg315 stg316 stg317 stg318 stg319 stg320 stg321 stg322 stg323 stg324 stg325 stg326 stg327 stg328 stg329 stg330 stg331 stg332 stg333 stg334 stg335 stg336 stg337 stg338 stg339 stg340 stg341 stg342 stg343 stg344 stg345 stg346 stg347 stg348 stg349 stg350 stg351 stg352 stg353 stg354 stg355 stg356 stg357 stg358 stg359 stg360 stg361 stg362 stg363 stg364 stg365 stg366 stg367 stg368 stg369 stg370 stg371 stg372 stg373 stg374 stg375 stg376 stg377 stg378 stg379 stg380 stg381 stg382 stg383 stg384 stg385 stg386 stg387 stg388 stg389 stg390 stg391 stg392 stg393 stg394 stg395 stg396 stg397 stg398 stg399 stg400 stg401 stg402 stg403 stg404 stg405 stg406 stg407 stg408 stg409 stg410 stg411 stg412 stg413 stg414 stg415 stg416 stg417 stg418 stg419 stg420 stg421 stg422 stg423 stg424 stg425 stg426 stg427 stg428 stg429 stg430 stg431 stg432 stg433 stg434 stg435 stg436 stg437 stg438 stg439 stg440 stg441 stg442 stg443 stg444 stg445 stg446 stg447 stg448 stg449 stg450 stg451 stg452 stg453 stg454 stg455 stg456 stg457 stg458 stg459 stg460 stg461 stg462 stg463 stg464 stg465 stg466 stg467 stg468 stg469 stg470 stg471 stg472 stg473 stg474 stg475 stg476 stg477 stg478 stg479 stg480 stg481 stg482 stg483 stg484 stg485 stg486 stg487 stg488 stg489 stg490 stg491 stg492 stg493 stg494 stg495 stg496 stg497 stg498 stg499 stg500 stg501 stg502 stg503 stg504 stg505 stg506 stg507 stg508 stg509 stg510 stg511 stg512 stg513 stg514 stg515 stg516 stg517 stg518 stg519 stg520 stg521 stg522 stg523 stg524 stg525 stg526 stg527 stg528 stg529 stg530 stg531 stg532 stg533 stg534 stg535 stg536 stg537 stg538 stg539 stg540 stg541 stg542 stg543 stg544 stg545 stg546 stg547 stg548 stg549 stg550 stg551 stg552 stg553 stg554 stg555 stg556 stg557 stg558 stg559 stg560 stg561 stg562 stg563 stg564 stg565 stg566 stg567 stg568 stg569 stg570 stg571 stg572 stg573 stg574 stg575 stg576 stg577 stg578 stg579 stg580 stg581 stg582 stg583 stg584 stg585 stg586 stg587 stg588 stg589 stg590 stg591 stg592 stg593 stg594 stg595 stg596 stg597 stg598 stg599 stg600 stg601 stg602 stg603 stg604 stg605 stg606 stg607 stg608 stg609 stg610 stg611 stg612 stg613 stg614 stg615 stg616 stg617 stg618 stg619 stg620 stg621 stg622 stg623 stg624 stg625 stg626 stg627 stg628 stg629 stg630 stg631 stg632 stg633 stg634 stg635 stg636 stg637 stg638 stg639 stg640 stg641 stg642 stg643 stg644 stg645 stg646 stg647 stg648 stg649 stg650 stg651 stg652 stg653 stg654 stg655 stg656 stg657 stg658 stg659 stg660 stg661 stg662 stg663 stg664 stg665 stg666 stg667 stg668 stg669 stg670 stg671 stg672 stg673 stg674 stg675 stg676 stg677 stg678 stg679 stg680 stg681 stg682 stg683 stg684 stg685 stg686 stg687 stg688 stg689 stg690 stg691 stg692 stg693 stg694 stg695 stg696 stg697 stg698 stg699 stg700 stg701 stg702 stg703 stg704 stg705 stg706 stg707 stg708 stg709 stg710 stg711 stg712 stg713 stg714 stg715 stg716 stg717 stg718 stg719 stg720 stg721 stg722 stg723 stg724 stg725 stg726 stg727 stg728 stg729 stg730 stg731 stg732 stg733 stg734 stg735 stg736 stg737 stg738 stg739 stg740 stg741 stg742 stg743 stg744 stg745 stg746 stg747 stg748 stg749 stg750 stg751 stg752 stg753 stg754 stg755 stg756 stg757 stg758 stg759 stg760 stg761 stg762 stg763 stg764 stg765 stg766 stg767 stg768 stg769 stg770 stg771 stg772 stg773 stg774 stg775 stg776 stg777 stg778 stg779 stg780 stg781 stg782 stg783 stg784 stg785 stg786 stg787 stg788 stg789 stg790 stg791 stg792 stg793 stg794 stg795 stg796 stg797 stg798 stg799 stg800 stg801 stg802 stg803 stg804 stg805 stg806 stg807 stg808 stg809 stg810 stg811 stg812 stg813 stg814 stg815 stg816 stg817 stg818 stg819 stg820 stg821 stg822 stg823 stg824 stg825 stg826 stg827 stg828 stg829 stg830 stg831 stg832 stg833 stg83
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/guest-os.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/virt/kvm_vm.py`
- `virt_env_process.py`
- `client/tests/kvm/tests/enospc.py`
- `client/tests/kvm/tests/image_copy.py`

Referenced By No other documentation currently references this configuration key.

images_good

Description Sets the URI of a NFS server that hosts “good” (think “golden”) images, that will be copied to the local system prior to running other tests.

The act of copying of “good” images is an alternative to installing a VM from scratch before running other tests.

The default value is actually an invalid value that must be changed if you intend to use this feature:

```
images_good = 0.0.0.0:/autotest/images_good
```

Defined On

- `client/tests/kvm/base.cfg.sample`

Used By

- `client/virt/tests/image_copy.py`

Referenced By No other documentation currently references this configuration key.

image_format

Description Sets the format of the backing image file for a given drive.

The value of this configuration key is usually passed verbatim to image creation commands. It's worth noticing that QEMU has support for many formats, while virt-test currently plays really well only with **qcow2** and **raw**.

You can also use **vmdk**, but it's considered 'not supported', at least on image conversion tests.

To set the default image format:

```
image_format = qcow2
```

To set the image format for another image:

```
# Tests
variants:
  - block_hotplug: install setup image_copy unattended_install.cdrom
    images += " stg"
    image_format_stg = raw
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- [client/virt/virt_vm.py](#)
- [client/tests/kvm/tests/qemu_img.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [images](#)
- [image_name](#)
- [image_size](#)

image_name

Description Sets the name of an image file.

If the image file is not a block device (see `image_raw_device`) the actual file created will be named accordingly (together with the extension, according to `image_format`).

When this configuration key is used without a suffix, it's setting the name of all images without a specific name. The net effect is that it sets the name of the 'default' image. Example:

```
# Guests
variants:
  - @Linux:
      variants:
        - Fedora:
            variants:
              - 15.64:
                  image_name = f15-64
```

This example means that when a Fedora 15 64 bits is installed, and has a backing image file created, it's going to be named starting with 'f15-64'. If the `image_format` specified is 'qcow2', then the complete filename will be 'f15-64.qcow2'.

When this configuration key is used with a suffix, it sets the name of a specific image. Example:

```
# Tests
variants:
  - block_hotplug: install setup image_copy unattended_install.cdrom
      images += " stg"
      image_name_stg = storage
```

Defined On

- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)
- [client/tests/kvm/tests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)
- [client/tests/kvm/tests/qemu_img.py](#)

Referenced By

- [How to run virt-test tests on an existing guest image?](#)

See Also

- [images](#)
- [image_format](#)
- [image_raw_device](#)

[image_raw_device](#)

Description Flags whether the backing image for a given drive is a block device instead of a regular file.

By default we assume all images are backed by files:

```
image_raw_device = no
```

But suppose you define a new variant, for another guest, that will have a disk backed by a block device (say, an LVM volume):

```
CustomGuestLinux:
    image_name = /dev/vg/linux_guest
    image_raw_device = yes
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/tests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By

- [How to run virt-test tests on an existing guest image?](#)

image_size

Description Sets the size of image files. This applies to images creation and also validation tests (when checking that a image was properly created according to what was requested).

By default the image size is set to 10G:

```
image_size = 10G
```

But a VM can have other drives, backed by other image files (or block devices), with different sizes:

```
# Tests
variants:
    - block_hotplug: install setup image_copy unattended_install.cdrom
      images += " stg"
      boot_drive_stg = no
      image_name_stg = storage
      image_size_stg = 1G
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)
- [client/tests/kvm/tests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)
- [client/tests/kvm/tests/qemu_img.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [images](#)
- [image_name](#)
- [image_format](#)

keep_ppm_files

Description Configures whether should we keep the original screedump files in [PPM](#) format when converting them to [PNG](#), according to [convert_ppm_files_to_png](#)

To keep the PPM files:

```
keep_ppm_files = yes
```

To keep the PPM files only on situations with failures:

```
keep_ppm_files_on_error = yes
```

Defined On This configuration key is not currently defined on any sample cartesian configuration file, but a sample (commented out) appears on:

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

keep_screendumps

Description Flags whether screendumps (screenshots of the VM console) should be kept or delete during post processing.

To keep the screendumps:

```
keep_screendumps = yes
```

Usually we're only interested in keeping screendumps on situations with failures, to ease the debugging:

```
keep_screendumps_on_error = yes
```

Defined On The stock configuration key (without suffix) is not currently defined on any sample cartesian configuration file.

The configuration key with the 'on_error' suffix is defined on:

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

kill_unresponsive_vms

Description Configures whether VMs that are running, but do not have a responsive session (for example via SSH), should be destroyed (of course, not gracefully) during post processing.

This behavior is enabled by default. To turn it off and leave unresponsive VMs lying around (usually **not** recommended):

```
kill_unresponsive_vms = no
```

Defined On

- `client/tests/kvm/base.cfg.sample`

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See also

- `kill_vm`
- `kill_vm_timeout`
- `kill_vm_gracefully`

kill_vm

Description Configures whether a VM should be shutdown during post processing. How exactly the VM will be shutdown is configured by other parameters such as `kill_vm_gracefully` and `kill_vm_timeout`.

To force shutdown during post processing:

```
kill_vm = yes
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/guest-os.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`
- `client/tests/kvm/unittests.cfg.sample`

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See also

- [kill_vm_timeout](#)
- [kill_vm_gracefully](#)
- [kill_unresponsive_vms](#)

[kill_vm_gracefully](#)

Description Flags whether a graceful shutdown command should be sent to the VM guest OS before attempting to either halt the VM at the hypervisor side (sending an appropriate command to QEMU or even killing its process).

Of course, this is only valid when `kill_vm` is set to ‘yes’.

To force killing VMs without using a graceful shutdown command (such as ‘shutdown -h now’):

```
kill_vm_gracefully = no
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See also

- [kill_vm](#)
- [kill_vm_timeout](#)
- [kill_unresponsive_vms](#)

[kill_vm_timeout](#)

Description Configures the amount of time, in seconds, to wait for VM shutdown during the post processing.

This is only relevant if `kill_vm` is actually set to ‘yes’.

To set the timeout to one minute:

```
kill_vm_timeout = 60
```

Defined On

- [client/tests/kvm/guest-os.cfg.sample](#)
- [client/tests/kvm/subtests.cfg.sample](#)

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See also

- `kill_vm`
- `kill_vm_gracefully`
- `kill_unresponsive_vms`

login_timeout

Description Sets the amount of time, in seconds, to wait for a session (SSH/Telnet/Netcat) with the VM.

To set the timeout to 6 minutes:

```
login_timeout = 360
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/virt/tests/autotest.py`
- `client/virt/tests/boot.py`
- `client/virt/tests/clock_getres.py`
- `client/virt/tests/ethtool.py`
- `client/virt/tests/file_transfer.py`
- `client/virt/tests/fillup_disk.py`
- `client/virt/tests/guest_s4.py`
- `client/virt/tests/guest_test.py`
- `client/virt/tests/iofuzz.py`
- `client/virt/tests/ioquit.py`
- `client/virt/tests/iozone_windows.py`
- `client/virt/tests/jumbo.py`
- `client/virt/tests/kdump.py`
- `client/virt/tests/linux_s3.py`
- `client/virt/tests/lvm.py`
- `client/virt/tests/mac_change.py`

- `client/virt/tests/multicast.py`
- `client/virt/tests/netperf.py`
- `client/virt/tests/nicdriver_unload.py`
- `client/virt/tests/nic_promisc.py`
- `client/virt/tests/ping.py`
- `client/virt/tests/shutdown.py`
- `client/virt/tests/softlockup.py`
- `client/virt/tests/stress_boot.py`
- `client/virt/tests/vlan.py`
- `client/virt/tests/watchdog.py`
- `client/virt/tests/whql_client_install.py`
- `client/virt/tests/whql_submission.py`
- `client/virt/tests/yum_update.py`
- `client/tests/kvm/tests/balloon_check.py`
- `client/tests/kvm/tests/cdrom.py`
- `client/tests/kvm/tests/cpu_hotplug.py`
- `client/tests/kvm/tests/enospc.py`
- `client/tests/kvm/tests/floppy.py`
- `client/tests/kvm/tests/hdparm.py`
- `client/tests/kvm/tests/migration_multi_host.py`
- `client/tests/kvm/tests/migration.py`
- `client/tests/kvm/tests/migration_with_file_transfer.py`
- `client/tests/kvm/tests/migration_with_reboot.py`
- `client/tests/kvm/tests/multi_disk.py`
- `client/tests/kvm/tests/nic_bonding.py`
- `client/tests/kvm/tests/nic_hotplug.py`
- `client/tests/kvm/tests/nmi_watchdog.py`
- `client/tests/kvm/tests/pci_hotplug.py`
- `client/tests/kvm/tests/physical_resources_check.py`
- `client/tests/kvm/tests/qemu_img.py`
- `client/tests/kvm/tests/set_link.py`
- `client/tests/kvm/tests/smbios_table.py`
- `client/tests/kvm/tests/stop_continue.py`
- `client/tests/kvm/tests/system_reset_bootable.py`
- `client/tests/kvm/tests/timedrift.py`
- `client/tests/kvm/tests/timedrift_with_migration.py`

- [client/tests/kvm/tests/timedrift_with_reboot.py](#)
- [client/tests/kvm/tests/timedrift_with_stop.py](#)
- [client/tests/kvm/tests/trans_hugepage_defrag.py](#)
- [client/tests/kvm/tests/trans_hugepage.py](#)
- [client/tests/kvm/tests/trans_hugepage_swapping.py](#)
- [client/tests/kvm/tests/usb.py](#)
- [client/tests/kvm/tests/vmstop.py](#)

Referenced By No other documentation currently references this configuration key.

main_monitor

Description Sets the default monitor for a VM, meaning that when a test accesses the **monitor** property of a **VM** class instance, that one monitor will be returned.

Usually a VM will have a single monitor, and that will be a regular Human monitor:

```
main_monitor = humanmonitor1
```

If a **main_monitor** is not defined, the **monitor** property of a **VM** class instance will assume that the first monitor set in the monitors list is the main monitor.

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [monitors](#)
- [monitor_type](#)
- [client/virt/kvm_monitor.py](#)

main_vm

Description Sets name of the main VM.

There's nothing special about this configuration item, except that most tests will also reference its value when fetching a VM from the Environment (see class **Env** on file [client/virt/virt_utils.py](#)).

The default name of the main VM is **vm1**:

```
main_vm = vm1
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Referenced By No other documentation currently references this configuration key.

mem

Description Sets the amount of memory (in MB) a VM will have.

The amount of memory a VM will have for most tests is of the main VM is 1024:

```
mem = 1024
```

But for running KVM unittests, we currently set that to 512:

```
mem = 512
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

migration_mode

Description If migration mode is specified, the VM will be started in incoming mode for migration. Valid modes for migration are: **tcp**, **unix** and **exec**.

To start a VM in incoming mode for receiving migration data via tcp:

```
migration_mode = tcp
```

A port will be allocated from the range 5200 to 6000.

Defined On This configuration item is currently not defined on a sample cartesian configuration file.

Used By

- [client/tests/kvm/migration_control.srv](#)

Referenced By No other documentation currently references this configuration key.

monitors

Description Sets the list of [monitors](#) that a VM currently has running. See [QEMU has two types of monitors:

- The regular, also known as Human monitor, intended for interaction with people (but also very much used by other tools, Autotest inclusive)
- The QMP monitor, a monitor that speaks the [QMP](#) protocol.

Usually a VM will have a single monitor, and that will be a regular Human monitor:

```
monitors = humanmonitor1
main_monitor = humanmonitor1
monitor_type_humanmonitor1 = human
monitor_type = human
```

The monitor type is defined by `monitor_type`.

Here's a more detailed explanation of the configuration snippet above:

```
monitors = humanmonitor1
```

The default VM will have only one monitor, named **humanmonitor1**.

```
main_monitor = humanmonitor1
```

The main monitor will also be **humanmonitor1**. When a test has to talk to a monitor, it usually does so through the main monitor.

```
monitor_type_humanmonitor1 = human
```

This configuration sets the specific type of the **humanmonitor1** to be **human**.

```
monitor_type = human
```

And finally this configuration sets the default monitor type also to be **human**.

Suppose you define a new monitor for your VMs:

```
monitors += ' monitor2'
```

Unless you also define:

```
monitor_type_monitor2 = qmp
```

monitor2 will also be a human monitor.

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/tests/kvm/kvm.py](#)
- [client/virt/kvm_vm.py](#)
- [client/virt/virt_test_utils.py](#)

Note: most tests that interact with the monitor do so through the **monitor** property of the **VM** class, and not by evaluating this parameter value. This is usually only done by the **VM** class.

Referenced By No other documentation currently references this configuration key.

See Also

- [client/virt/kvm_monitor.py](#)

monitor_type

Description Sets the type of the [monitor](#). QEMU has two types of monitors:

- The regular, also known as Human monitor, intended for interaction with people (but also very much used by other tools, Autotest inclusive)
- The QMP monitor, a monitor that speaks the [QMP](#) protocol.

To set the default monitor type to be a [QMP](#) monitor:

```
monitor_type = qmp
```

To set the type of a specific monitor use:

```
monitor_type_humanmonitor1 = human
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)
- [client/tests/kvm/unittests.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [client/virt/kvm_monitor.py](#)

nic_mode

Description Configures the mode of a Network Interface Card.

Suitable values for this configuration item are either **user** or **tap**.

[User mode](#) networking is the default **on QEMU**, but [Tap mode](#) is the current default in Autotest:

```
nic_mode = tap
```

When **nic_mode** is set to [Tap](#) you should also set a bridge.

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`
- `client/tests/kvm/migration_control.srv`

Used By

- `client/virt/kvm_vm.py`
- `client/tests/kvm/tests/physical_resources_check.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `bridge`
- `redirs`

nics

Description Sets the list of network interface cards that a VM will have.

Usually a VM will start with a single nic, named `nic1`:

```
nics = nic1
```

But a VM can have other nics. Some tests (usually network related) add other nics. One obvious example is the [bonding test](#):

```
# Tests
variants:
  - nic_bonding: install setup image_copy unattended_install.cdrom
    nics += ' nic2 nic3 nic4'
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/virt/virt_vm.py`
- `client/virt/kvm_vm.py`
- `client/tests/kvm/tests/nic_bonding.py`
- `client/tests/kvm/tests/physical_resources_check.py`

Referenced By No other documentation currently references this configuration key.

pre_command

Description Configures a command to be executed during pre processing.

The pre processing code will execute the given command, waiting for an amount of time and failing the test unless the command is considered noncritical.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [pre_command_timeout](#)
- [pre_command_non_critical?](#)

pre_command_timeout

Description Configures the amount of time to wait while executing a command during pre processing.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [pre_command](#)
- [pre_command_non_critical?](#)

pre_command_noncritical

Description Flags if the command configured to to be executed during pre processing, is not critical, that is, if an error during its execution should only logged or fail the test.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `pre_command`
- `pre_command_timeout`

profilers

Description Sets the list of Autotest profilers to be enabled during the test run (they're removed from the job's list of profilers when the test finishes).

This is commonly used to enable the `kvm_stat` profiler:

```
profilers = kvm_stat
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/subtests.cfg.sample`

Used By

- `client/virt/virt_utils.py`

Referenced By No other documentation currently references this configuration key.

See Also

- Setting up profiling on virt-test
- Using and developing job profilers

post_command

Description Configures a command to be executed during post processing.

The pre processing code will execute the given command, waiting for an amount of time and failing the test unless the command is considered noncritical.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `post_command_timeout`
- `post_command_non_critical?`

`post_command_timeout`

Description Configures the amount of time to wait while executing a command during post processing.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `post_command`
- `post_command_non_critical?`

`post_command_noncritical`

Description Flags if the command configured to to be executed during pre processing, is not critical, that is, if an error during its execution should only logged or fail the test.

Defined On This configuration key is not currently defined on any sample cartesian configuration file in its stock format.

Used By

- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `post_command`
- `post_command_timeout`

qemu_binary

Description Sets either the name or full path for the QEMU binary.

By default this is as simple as possible:

```
qemu_binary = qemu
```

But while testing the qemu-kvm userspace, one could use:

```
qemu_binary = /usr/bin/qemu-kvm
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/tests.cfg.sample`
- `client/tests/kvm/unittests.cfg.sample`

Used By

- `client/virt/kvm_vm.py`
- `client/virt/virt_env_process.py`

Referenced By No other documentation currently references this configuration key.

See Also

- `qemu_img_binary`

qemu_img_binary

Description Sets either the name or full path for the **qemu-img** binary.

By default this is as simple as possible:

```
qemu_img_binary = qemu-img
```

Defined On

- `client/tests/kvm/base.cfg.sample`
- `client/tests/kvm/tests.cfg.sample`
- `client/tests/kvm/unittests.cfg.sample`

Used By

- `client/virt/virt_vm.py`
- `client/tests/kvm/tests/qemu_img.py`

Referenced By No other documentation currently references this configuration key.

See Also

- [qemu_binary](#)

qxl_dev_nr

Description Sets the number of display devices available through [SPICE](#). This is only valid when [qxl](#) is set.

The default configuration enables a single display device:

```
qxl_dev_nr = 1
```

Note that due to a limitation in the current Autotest code (see [client/virt/kvm_vm.py](#)) this setting is only applied when the QEMU syntax is:

```
# qemu -qxl 2
```

and not applied when the syntax is:

```
# qemu -vga qxl
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [qxl](#)
- [vga?](#)
- [display](#)

qxl

Description Flags if the [VGA](#) device should be an of type [qxl](#).

The default configuration enables a [qxl](#) VGA:

```
qxl = on
```

Note that if [vga?](#) is also set, [qxl](#) takes precedence over it.

Defined On

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [qxl_dev_nr](#)
- [vga?](#)

redirs

Description Sets the network redirections between host and guest. These are only used and necessary when using 'user' mode network.

Example:

```
redirs = remote_shell
guest_port_remote_shell = 22
```

A port will be allocated on the host, usually within the range 5000-6000, and all traffic to/from this port will be redirect to guest's port 22.

Defined On

- [client/tests/kvm/tests_base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See also

- [guest_port](#)
- [guest_port_remote_shell](#)

remove_image

Description Configures if we want to remove image files during post processing.

To keep all images after running tests:

```
remove_image = no
```

On a test with multiple transient images, to remove all but the main image (**image1**), use:

```
remove_image = yes
remove_image_image1 = no
```

Defined On

- [client/tests/kvm/tests_base.cfg.sample](#)

Used By

- [client/virt/virt_env_process.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [images](#)
- [image_name](#)
- [image_format](#)
- [create_image](#)
- [force_create_image](#)

spice

Description Sets extra arguments to be passed to the QEMU **-spice** command line argument.

Note that there's no need to pass a port number, as this will be automatically allocated from the 8000 - 8100 range.

By default, the extra arguments disable authentication:

```
spice = disable-ticketing
```

Defined On

- [client/tests/kvm/base.cfg.sample](#)

Used By

- [client/virt/kvm_vm.py](#)

Referenced By No other documentation currently references this configuration key.

See Also

- [qxl](#)
- [qxl_dev_nr](#)
- [vga?](#)
- [display](#)

Cartesian Config Tricks

Changing test order

The Cartesian Config system implemented in virt-test does have some limitations - for example, the order of tests is dependent on the order on which each variant is defined on the config files, making executing tests on a different order a daunting prospect.

In order to help people with this fairly common use case, we'll demonstrate how to use some of the cartesian config features to accomplish executing your tests in the order you need. In this example, we're going to execute the *unix* migration mode tests before the *tcp* one. In the actual cartesian config file, *tcp* is always going to be executed before *unix* on a normal virt-test execution.

Create a custom config For the sake of simplicity, we'll create the file under *backends/qemu/cfg/custom.cfg*:

```
$ touch backends/qemu/cfg/custom.cfg
```

Then, let's add the following text to it (please keep in mind that our maintainers are constantly adding new variants to the base virt-test config, so you might need to tweak the contents to match the current state of the config):

```
include tests-shared.cfg

variants:
  - @custom_base:
    only JeOS.20
    only i440fx
    only smp2
    only qcow2
    only virtio_net
    only virtio_blk
    no hugepages
    no 9p_export
    no gluster
    no pf_assignable
    no vf_assignable
    no rng_random
    no rng_egd
    variants:
      - @custom_1:
        only migrate.default.unix
      - @custom_2:
        only migrate.default.tcp
```

There you go. Note that you are not obligated to use @ at your variant names, it's just for the sake of not polluting the tag namespace too much. Now, let's test to see if this config file is generating us just the 2 tests we actually want:

```
$ virttest/cartesian_config.py backends/qemu/cfg/custom.cfg
dict 1: qcow2.virtio_blk.smp2.virtio_net.JeOS.20.x86_64.io-github-autotest-qemu.migrate.unix
dict 2: qcow2.virtio_blk.smp2.virtio_net.JeOS.20.x86_64.io-github-autotest-qemu.migrate.tcp
```

There you go. Now, you can simply execute this command line with:

```
./run -t qemu -c backends/qemu/cfg/custom.cfg
```

And then you'll see your tests executed in the correct order:

```
$ ./run -t qemu -c backends/qemu/cfg/custom.cfg
SETUP: PASS (2.31 s)
```

```
DATA DIR: /home/user/virt_test
DEBUG LOG: /home/user/Code/virt-test.git/logs/run-2014-12-19-12.12.29/debug.log
TESTS: 2
(1/2) qcow2.virtio_blk.smp2.virtio_net.JeOS.20.x86_64.io-github-autotest-qemu.migrate.unix: PASS (31.00 s)
(2/2) qcow2.virtio_blk.smp2.virtio_net.JeOS.20.x86_64.io-github-autotest-qemu.migrate.tcp: PASS (22.00 s)
TOTAL TIME: 53.25 s
TESTS PASSED: 2
TESTS FAILED: 0
SUCCESS RATE: 100.00 %
```

This is the base idea - you can extend and filter variants on a cartesian config set as much as you'd like, and tailor it to your needs.

Extra docs

Extra information that does not quite fit in other areas of the docs.

Contents:

Links with downloadable images for virt tests

This is a central location that we aim to keep up to date with locations of iso files that might be needed for testing.

Update: Now we have a central location to define such downloads. In the source tree:

```
shared/download.d/
```

Contains a bunch of .ini files, each one with download definitions. It is expected that this will be more up to date than this page. You can see the available downloads and download the files using:

```
tools/download_manager.py
```

Winutils ISO

<http://lmr.fedorapeople.org/winutils/winutils.iso>

JeOS image

You can find the JeOS images here:

<http://lmr.fedorapeople.org/jeos/>

You'll find .7za (p7zipped) files for versions of the JeOS available, as well as their MD5SUM files.

GlusterFS support

GlusterFS is an open source, distributed file system capable of scaling to several petabytes (actually, 72 brontobytes!) and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design and can deliver exceptional performance for diverse workloads.

More details of GlusterFS can be found under

<http://www.gluster.org/about/>

GlusterFS is added as a new block backend for qemu and to make use of this feature we require the following components.

More details of GlusterFS-QEMU Integration can be found under

<http://raobharata.wordpress.com/2012/10/29/qemu-glusterfs-native-integration/>

1. Qemu- 1.3, 03Dec2012
2. GlusterFS-3.4
3. Libvirt-1.0.1, 15Dec2012

How to use in virt-test

You can use virt-test to test GlusterFS support with following steps.

1. Edit qemu/cfg/tests.cfg with following changes,

```
only glusterfs_support
remove 'only no_glusterfs_support' line from the file
```

- 2) Optionally, edit shared/cfg/guest-hw.cfg for the gluster volume name and brick path, default is going to be,

```
gluster_volume_name = test-vol
gluster_brick = /tmp/gluster
```

How to use manually

The following is just an example to show how we create gluster volume and run a guest on that volume manually.

Starting Gluster daemon

```
service glusterd start
```

Gluster volume creation

```
gluster volume create [volume-name] [hostname/host_ip]:/[brick_path]
```

E:g: *gluster volume create test-vol satheesh.ibm.com://home/satheesh/images_gluster*

Qemu Img creation

```
qemu-img create gluster://[hostname]:0/[volume-name]/[image-name] [size]
```

E:g: *qemu-img create gluster://satheesh.ibm.com:0/test-vol/test_gluster.img 10G*

Example of qemu cmd Line

```
qemu-system-x86_64 --enable-kvm -smp 4 -m 2048 -drive file=gluster://satheesh.ibm.com/test-vol/test_
```

Setting up a Regression Test Farm for KVM

You have all upstream code, and you're wondering if the internal Red Hat testing of KVM has a lot of internal 'secret sauce'. No, it does not.

However, it is a complex endeavor, since there are *lots* of details involved. The farm setup and maintenance is not easy, given the large amounts of things that can fail (machines misbehave, network problems, git repos unavailable, so on and so forth). *You have been warned.*

With all that said, we'll share what we have been doing. We did clean up our config files and extensions and released them upstream, together with this procedure, that we hope it will be useful to you guys. Also, this will cover KVM testing on a single host, as tests involving multiple hosts and Libvirt testing are a work in progress.

The basic steps are:

1. Install an autotest server.
2. Add machines to the server (test nodes). Those machines are the virt hosts that will be tested.
3. Prepare the virt test jobs and schedule them.
4. Set up cobbler in your environment so you can install hosts.
5. Lots of trial and error until you get all little details sorted out.

We took years repeating all the steps above and perfecting the process, and we are willing to document it all to the best extent possible. I'm afraid however, that you'll have to do your homework and adapt the procedure to your environment.

Some conventions

We are assuming you will install autotest to its default upstream location

`/usr/local/autotest`

Therefore a lot of paths referred here will have this as the base dir.

CLI vs Web UI

During this text, we'll use frequently the terms CLI and Web UI.

By CLI we mean specifically the program:

`/usr/local/autotest/cli/autotest-rpc-client`

That is located in the autotest code checkout.

By Web UI, we mean the web interface of autotest, that can be accessed through

<http://your-autotest-server.com/afe>

Step 1 - Install an autotest server

Provided that you have internet on your test lab, this should be the easiest step. Pick up either a VM accessible in your lab, or a bare metal machine (it really doesn't make a difference, we use a VM here). We'll refer it from now on as the "Server" box.

The hard drive of the Server should hold enough room for test results. We found out that at least 250 GB holds data for more than 6 months, provided that QEMU doesn't crash a lot.

You'll follow the procedure described on

<https://github.com/autotest/autotest/wiki/AutotestServerInstallRedHat>

for Red Hat derivatives (such as Fedora and RHEL), and

<https://github.com/autotest/autotest/wiki/AutotestServerInstall>

for Debian derivatives (Debian, Ubuntu).

Note that using the install script referred right in the beginning of the documentation is the preferred method, and should work pretty well if you have internet on your lab. In case you don't have internet there, you'd need to follow the instructions after the 'installing with the script' instructions. Let us know if you have any problems.

Step 2 - Add test machines

It should go without saying, but the machines you have to add have to be virtualization capable (support KVM).

You can add machines either by using the CLI or the Web UI, following the documentation:

<https://github.com/autotest/autotest/wiki/ConfiguringHosts>

If you don't want to read that, I'll try to write a quick howto.

Say you have two x86_64 hosts, one AMD and the other, Intel. Their hostnames are:

foo-amd.bazcorp.com foo-intel.bazcorp.com

I would create 2 labels, amd64 and intel64, I would also create a label to indicate the machines can be provisioned by cobbler. This is because you can tell autotest to run a job in any machine that matches a given label.

Logged as the autotest user:

```
$ /usr/local/autotest/cli/autotest-rpc-client label create -t amd64
Created label:
    'amd64'
$ /usr/local/autotest/cli/autotest-rpc-client label create -t intel64
Created label:
    'intel64'
$ /usr/local/autotest/cli/autotest-rpc-client label create hostprovisioning
Created label:
    'hostprovisioning'
```

Then I'd create each machine with the appropriate labels

```
$ /usr/local/autotest/cli/autotest-rpc-client host create -t amd64 -b hostprovisioning foo-amd.bazcorp.com
Added host:
    foo-amd.bazcorp.com

$ /usr/local/autotest/cli/autotest-rpc-client host create -t amd64 -b hostprovisioning foo-intel.bazcorp.com
Added host:
    foo-intel.bazcorp.com
```

Step 3 - Prepare the test jobs

Now you have to copy the plugin we have developed to extend the CLI to parse additional information for the virt jobs:

```
cp /usr/local/autotest/contrib/virt/site_job.py /usr/local/autotest/cli/
```

This should be enough to enable all the extra functionality.

You also need to copy the site-config.cfg file that we published as a reference, to the qemu config module:

```
cp /usr/local/autotest/contrib/virt/site-config.cfg /usr/local/autotest/client/tests/virt/qemu/cfg
```

Be aware that you *need* to read this file well, and later, configure it to your testing needs. We specially stress that you might want to create private git mirrors of the git repos you want to test, so you tax the upstream mirrors less, and have increased reliability.

Right now it is able to run regression testing on Fedora 18, and upstream kvm, provided that you have a cobbler instance functional, with a profile called f18-autotest-kvm that can be properly installed on your machines. Having that properly set up may open another can of worms.

One simple way to schedule the jobs, that we does use at our server, is to use cron to schedule daily testing jobs of the things you want to test. Here is an example that should work ‘out of the box’. Provided that you have an internal mailing list that you created with the purpose of receiving email notifications, called autotest-virt-jobs@foocorp.com, you can stick that on the crontab of the user autotest in the Server:

```
07 00 * * 1-7 /usr/local/autotest/cli/autotest-rpc-client job create -B never -a never -s -e autotest
15 00 * * 1-7 /usr/local/autotest/cli/autotest-rpc-client job create -B never -a never -s -e autotest
07 01 * * 1-7 /usr/local/autotest/cli/autotest-rpc-client job create -B never -a never -s -e autotest
15 01 * * 1-7 /usr/local/autotest/cli/autotest-rpc-client job create -B never -a never -s -e autotest
```

That should be enough to have one sanity and stable job for:

- Fedora 18.
- qemu.git userspace and kvm.git kernel.

What does these ‘stable’ and ‘sanity’ jobs do? In short:

- Host OS (Fedora 18) installation through cobbler
- Latest kernel for the Host OS installation (either the last kernel update build for fedora, or check out, compile and install kvm.git).

sanity job

- Install latest Fedora 18 qemu-kvm, or compiles the latest qemu.git
- Installs a VM with Fedora 18, boots, reboots, does simple, single host migration with all supported protocols
- Takes about two hours. In fact, internally we test more guests, but they are not widely available (RHEL 6 and Windows 7), so we just replaced them with Fedora 18.

stable job

- Same as above, but many more networking, timedrift and other tests

Setup cobbler to install hosts

Cobbler is an installation server, that control DHCP and/or PXE boot for your x86_64 bare metal virtualization hosts. You can learn how to set it up in the following resource:

<https://github.com/cobbler/cobbler/wiki/Start%20Here>

You will set it up for simple installations, and you probably just need to import a Fedora 18 DVD into it, so it can be used to install your hosts. Following the import procedure, you’ll have a ‘profile’ created, which is a label that describes an OS that can be installed on your virtualization host. The label we chose, as already mentioned is f18-autotest-kvm. If you want to change that name, you’ll have to change site-config.cfg accordingly.

Also, you will have to add your test machines to your cobbler server, and will have to set up remote control (power on/off) for them.

The following is important:

The hostname of your machine in the autotest server has to be the name of your system in cobbler.

So, for the hypothetical example you'll have to have set up systems with names foo-amd.bazcorp.com foo-intel.bazcorp.com in cobbler. That's right, the 'name' of the system has to be the 'hostname'. Otherwise, autotest will ask cobbler and cobbler will not know which machine autotest is taking about.

Other assumptions we have here:

1) We have a (read only, to avoid people deleting isos by mistake) NFS share that has the Fedora 18 DVD and other ISOS. The structure for the base dir could look something like:

```
.
|-- linux
|   |-- Fedora-18-x86_64-DVD.iso
|-- windows
|   |-- en_windows_7_ultimate_x64_dvd_x15-65922.iso
|   |-- virtio-win.iso
|   |-- winutils.iso
```

This is just in case you are legally entitled to download and use Windows 7, for example.

2) We have another NFS share with space for backups of qcow2 images that got corrupted during testing, and you want people to analyze them. The structure would be:

```
.
|-- foo-amd
|-- bar-amd
```

That is, one directory for each host machine you have on your grid. Make sure they end up being properly configured in the kickstart.

Now here is one excerpt of kickstart with some of the gotchas we learned with experience. Some notes:

- This is not a fully formed, functional kickstart, just in case you didn't notice.
- This is provided in the hopes you read it, understand it and adapt things to your needs. If you paste this into your kickstart and tell me it doesn't work, I WILL silently ignore your email, and if you insist, your emails will be filtered out and go to the trash can.

```
install

text
reboot
lang en_US
keyboard us
rootpw --iscrypted [your-password]
firewall --disabled
selinux --disabled
timezone --utc America/New_York
firstboot --disable
services --enabled network --disabled NetworkManager
bootloader --location=mbr
ignoredisk --only-use=sda
zerombr
clearpart --all --drives=sda --initlabel
autopart
network --bootproto=dhcp --device=eth0 --onboot=on
```

```
%packages
@core
@development-libs
@development-tools
@virtualization
wget
dnsmasq
genisoimage
python-imaging
qemu-kvm-tools
gdb
iasl
libvirt
python-devel
ntpddate
gstreamer-plugins-good
gstreamer-python
dmidecode
popt-devel
libblkid-devel
pixman-devel
mtools
koji
tcpdump
bridge-utils
dosfstools
%end

%post

echo "[nfs-server-that-holds-iso-images]:[nfs-server-that-holds-iso-images]/base_path/iso /var/lib/virt-test/iso"
echo "[nfs-server-that-holds-iso-images]:[nfs-server-that-holds-iso-images]/base_path/steps_data /var/lib/virt-test/steps_data"
echo "[nfs-server-that-has-lots-of-space-for-backups]:/base_path/[dir-that-holds-this-hostname-backup] /var/lib/virt-test/backups"
mkdir -p /var/lib/virt-test/isos
mkdir -p /var/lib/virt-test/steps_data
mkdir -p /var/lib/virt-test/images
mkdir -p /var/lib/virt-test/images_archive

mkdir --mode=700 /root/.ssh
echo 'ssh-dss [the-ssh-key-of-the-Server-autotest-user]' >> /root/.ssh/authorized_keys
chmod 600 /root/.ssh/authorized_keys

ntpddate [your-ntp-server]
hwclock --systohc

systemctl mask tmp.mount
%end
```

Painful trial and error process to adjust details

After all that, you can start running your test jobs and see what things will need to be fixed. You can run your jobs easily by logging into your Server, with the autotest user, and use the command:

```
/usr/local/autotest/cli/autotest-rpc-client job create -B never -a never -s -e autotest-virt-jobs@foo
```

As you might have guessed, this will schedule a Fedora 18 sanity job. So go through it and fix things step by step. If anything, you can take a look at this:

<https://github.com/autotest/autotest/wiki/DiagnosingFailures>

And see if it helps. You can also ask on the mailing list, but *please, pretty please* do your homework before you ask us to guide you through all the process step by step. This is already a step by step procedure.

All right, good luck, and happy testing!

Installing Windows virtio drivers with virt-test

So, you want to use virt-test to install windows guests. You also want them to be installed with the paravirtualized drivers developed for windows. You have come to the right place.

A bit of context on windows virtio drivers install

This method of install so far covers the storage (viosstor) and network (NetKVM) drivers. virt-test uses a boot floppy with a Windows answer file in order to perform unattended install of windows guests. For winXP and win2003, the unattended files are simple .ini files, while for win2008 and later, the unattended files are XML files.

In order to install the virtio drivers during guest install, KVM autotest has to inform the windows install programs **where** to find the drivers. So, we work from the following assumptions:

1. You already have an iso file that contains windows virtio drivers (inf files) for both netkvm and viostor. If you are unsure how to generate that iso, there's an example script under contrib, inside the kvm test directory. Here is an example of how the files inside this cd would be organized, assuming the iso image is mounted under /tmp/virtio-win (the actual cd has more files, but we took only the parts that concern to the example, win7 64 bits).

```
/tmp/virtio-win/
/tmp/virtio-win/vista
/tmp/virtio-win/vista/amd64
/tmp/virtio-win/vista/amd64/netkvm.cat
/tmp/virtio-win/vista/amd64/netkvm.inf
/tmp/virtio-win/vista/amd64/netkvm.pdb
/tmp/virtio-win/vista/amd64/netkvm.sys
/tmp/virtio-win/vista/amd64/netkvmco.dll
/tmp/virtio-win/vista/amd64/readme.doc
/tmp/virtio-win/win7
/tmp/virtio-win/win7/amd64
/tmp/virtio-win/win7/amd64/balloon.cat
/tmp/virtio-win/win7/amd64/balloon.inf
/tmp/virtio-win/win7/amd64/balloon.pdb
/tmp/virtio-win/win7/amd64/balloon.sys
/tmp/virtio-win/win7/amd64/blnsrv.exe
/tmp/virtio-win/win7/amd64/blnsrv.pdb
/tmp/virtio-win/win7/amd64/vioser.cat
/tmp/virtio-win/win7/amd64/vioser.inf
/tmp/virtio-win/win7/amd64/vioser.pdb
/tmp/virtio-win/win7/amd64/vioser.sys
/tmp/virtio-win/win7/amd64/vioser-test.exe
/tmp/virtio-win/win7/amd64/vioser-test.pdb
/tmp/virtio-win/win7/amd64/viostor.cat
/tmp/virtio-win/win7/amd64/viostor.inf
/tmp/virtio-win/win7/amd64/viostor.pdb
/tmp/virtio-win/win7/amd64/viostor.sys
/tmp/virtio-win/win7/amd64/wdfcoinstaller01009.dll
...
```

If you are planning on installing WinXP or Win2003, you should also have a pre-made floppy disk image with the virtio drivers *and* a configuration file that the installer program will read to fetch the right drivers from it. Unfortunately, I don't have much info on how to build that file, you probably would have the image already assembled if you are willing to test those guest OS.

So you have to map the paths of your cd containing the drivers on the config variables. We hope to improve this in cooperation with the virtio drivers team.

Step by step procedure

We are assuming you already have the virtio cd properly assembled with you, as well as windows iso files that *do* match the ones provided in our test_base.cfg.sample*. Don't worry though, we try as much as possible to use files from MSDN, to standardize.

We will use win7 64 bits (non sp1) as the example, so the CD you'd need is:

```
cdrom_cd1 = isos/windows/en_windows_7_ultimate_x86_dvd_x15-65921.iso
shasum_cd1 = 5395dc4b38f7bdb1e005ff414deedfdb16dbf610
```

This file can be downloaded from the MSDN site, so you can verify the SHA1 sum of it matches.

1. Git clone autotest to a convenient location, say \$HOME/Code/autotest. See [the download source documentation](#). Please do use git and clone the repo to the location mentioned.
2. Execute the get_started.py script (see the get started documentation <../basic/GetStarted>'. It will create the directories where we expect the cd files to be available. You don't need to download the Fedora 14 DVD, but you do need to download the winutils.iso cd (on the example below, I have skipped the download because I do have the file, so I can copy it to the expected location, which is in this case /tmp/kvm_autotest_root/isos/windows). Please, do read the documentation mentioned on the script to avoid missing packages installed and other misconfiguration.
3. Create a windows dir under /tmp/kvm_autotest_root/isos
4. Copy your windows 7 iso to /tmp/kvm_autotest_root/isos/windows
5. Edit the file cdkeys.cfg and put the windows 7 64 bit key on that file
6. Edit the file win-virtio.cfg and verify if the paths are correct. You can see that by looking this session:

```
64:
    unattended_install.cdrom, whql.support_vm_install:
        # Look at your cd structure and see where the drivers are
        # actually located (viostor and netkvm)
        virtio_storage_path = 'F:\win7\amd64'
        virtio_network_path = 'F:\vista\amd64'

        # Uncomment if you have a nw driver installer on the iso
        #virtio_network_installer_path = 'F:\RHEV-Network64.msi'
```

7. If you are using the cd with the layout mentioned on the beginning of this article, the paths are already correct. However, if they're different (more likely), you have to adjust paths. Don't forget to read and do all the config on win-virtio.cfg file as instructed by the comments.
8. On tests.cfg, you have to enable virtio install of windows 7. On the block below, you have to change only rtl8139 to only virtio_net and only ide to only virtio-blk. You are informing autotest that you only want a vm with virtio hard disk and network device installed.

```
# Runs qemu-kvm, Windows Vista 64 bit guest OS, install, boot, shutdown
- @qemu_kvm_windows_quick:
    # We want qemu-kvm for this run
```



```

qemu_binary = /usr/bin/qemu-kvm
qemu_img_binary = /usr/bin/qemu-img
# Only qcow2 file format
only qcow2
# Only rtl8139 for nw card (default on qemu-kvm)
only rtl8139
# Only ide hard drives
only ide
# qemu-kvm will start only with -smp 2 (2 processors)
only smp2
# No PCI assignable devices
only no_pci_assignable
# No large memory pages
only smallpages
# Operating system choice
only Win7.64
# Subtest choice. You can modify that line to add more subtests
only unattended_install.cdrom, boot, shutdown

```

9. You have to change the bottom of tests.cfg to look like the below, Which means you are informing autotest to only run the test set mentioned above, rather than the default, that installs Fedora 15.

```
only qemu_kvm_windows_quick
```

10. As informed on the output of get_started.py, the command you can execute to run autotest is (please run this AS ROOT or sudo)

```
$HOME/Code/autotest/client/bin/autotest $HOME/Code/autotest/client/tests/kvm/control
```

11. Profit! You automated install of Windows 7 with the virtio drivers will be carried out.

If you want to install other guests, as you might imagine, you can change only Win7.64 with other guests, say only Win2008.64.sp2. Now, during the first time you perform your installs, it's good to watch the installation to see if there aren't problems such as a **wrong cd key** preventing your install from happening. virt-test prints the qemu command line used, so you can see which vnc display you can connect to to watch your vm being installed.

Please give us feedback on whether this procedure was helpful - email me at lmr AT redhat DOT com.

Running QEMU unittests with virt-test

For a while now, qemu-kvm does contain a unittest suite that can be used to assess the behavior of some KVM subsystems. Ideally, they are supposed to PASS, provided you are running both the latest qemu-kvm and the latest linux Avi's tree. virt-test for quite a long time has support for running them in an automated way. It's a good opportunity to put your git branch to unittest, starting from a clean state (KVM autotest will fetch from your git tree, leaving your actual development tree intact and doing things from scratch, and that is less likely to mask problems).

A bit of context on virt-test build tests

People usually don't know that virt-test has support to build and install QEMU/KVM for testing purposes, from many different software sources. You can:

1. Build qemu-kvm from a git repo (most common choice for developers hacking on code)
2. Install qemu-kvm from an rpm file (people testing a newly built rpm package)
3. Install qemu-kvm straight from the Red Hat build systems (Koji is the instance of the build system for Fedora, Brew is the same, but for RHEL. With this we can perform quality control on both Fedora and RHEL packages, trying to anticipate breakages before the packages hit users)

For this article, we are going to focus on git based builds. Also, we are focusing on Fedora and RHEL. We'll try to write the article in a pretty generic fashion, you are welcome to improve this with details on how to do the same on your favorite linux distribution.

Before you start

You need to verify that you can actually build qemu-kvm from source, as well as the unittest suite.

1. Make sure you have the appropriate packages installed. You can read [the install prerequisite packages \(client\) section](#) for more information.

Step by step procedure

1. Git clone autotest to a convenient location, say `$HOME/Code/autotest`. See [the download source documentation](#) Please do use git and clone the repo to the location mentioned.
2. Execute the `get_started.py` script (see [the get started documentation](#). If you just want to run unittests, you can safely skip each and every iso download possible, as *qemu-kvm will straight boot small kernel images (the unittests)* rather than full blown OS installs.
3. As running unittests is something that's fairly independent of other virt-test testing you can do, and it's something people are interested in, we prepared a *special control file* and a *special configuration file* for it. On the `kvm` directory, you can see the files `unittests.cfg` `control.unittests`. You only need to edit `unittests.cfg`.
4. The file `unittests.cfg` is a stand alone configuration for running unittests. It is comprised by a build variant and a unittests variant. Edit the file, it'll look like:

```
... bunch of params needed for the virt-test preprocessor
# Tests
variants:
  - build:
      type = build
      vms = ''
      start_vm = no
      # Load modules built/installed by the build test?
      load_modules = no
      # Save the results of this build on test.resultsdir?
      save_results = no
      variants:
        - git:
            mode = git
            user_git_repo = git://git.kernel.org/pub/scm/virt/kvm/qemu-kvm.git
            user_branch = next
            user_lbranch = next
            test_git_repo = git://git.kernel.org/pub/scm/virt/kvm/kvm-unit-test.git

  - unittest:
      type = unittest
      vms = ''
      start_vm = no
      unittest_timeout = 600
      testdev = yes
      extra_params += " -S"
      # In case you want to execute only a subset of the tests defined on the
      # unittests.cfg file on qemu-kvm, uncomment and edit test_list
      #test_list = idt_test hypercall vmexit realmode
```

```
only build.git unittest
```

5. As you can see above, you have places to specify both the userspace git repo and the unittest git repo. You are then free to replace `user_git_repo` with your own git repo. It can be a remote git location, or it can simply be the path to a cloned tree inside your development machine.
6. As of Fedora 15, that ships with gcc 4.6.0, the compilation is more strict, so things such as an unused variable in the code **will** lead to a build failure. You can disable that level of strictness by providing *extra configure script options* to your qemu-kvm userspace build. Right below the `user_git_repo` line, you can set the variable `extra_configure_options` to include `--disable-werror`. Let's say you also want virt-test to fetch from my local tree, `/home/lmr/Code/qemu-kvm`, master branch, same for the `kvm-unit-tests` repo. If you make those changes, your build variant will look like:

```
- git:
  mode = git
  user_git_repo = /home/lmr/Code/qemu-kvm
  extra_configure_options = --disable-werror
  user_branch = master
  user_lbranch = master
  test_git_repo = /home/lmr/Code/kvm-unit-tests
```

7. Now you can just run `virt-test` as usual, you just have to change the main control file (called `control` with the `unittest` one `control.unittests`

```
$HOME/Code/autotest/client/bin/autotest $HOME/Code/autotest/client/tests/kvm/control.unittests
```

8. The output of a typical unittest execution looks like. Notice that autotest informs you where the logs of each individual unittests are located, so you can check that out as well.

```
07/14 18:49:44 INFO | unittest:0052| Running apic
07/14 18:49:44 INFO |   kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 18:49:46 INFO | unittest:0096| Waiting for unittest apic to complete, timeout 600, output
07/14 18:59:46 ERROR| unittest:0108| Exception happened during apic: Timeout elapsed (600s)
07/14 18:59:46 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 18:59:46 INFO | unittest:0052| Running smptest
07/14 19:00:15 INFO |   aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:00:16 INFO |   kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:00:17 INFO | unittest:0096| Waiting for unittest smptest to complete, timeout 600, out
07/14 19:00:17 INFO |   aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:00:18 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:00:18 INFO | unittest:0052| Running smptest3
07/14 19:00:18 INFO |   kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:00:19 INFO | unittest:0096| Waiting for unittest smptest3 to complete, timeout 600, ou
07/14 19:00:19 INFO |   aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:00:20 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:00:20 INFO | unittest:0052| Running vmexit
07/14 19:00:20 INFO |   kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:00:21 INFO | unittest:0096| Waiting for unittest vmexit to complete, timeout 600, outp
07/14 19:00:31 INFO |   aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:00:31 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:00:31 INFO | unittest:0052| Running access
07/14 19:00:31 INFO |   kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:00:32 INFO | unittest:0096| Waiting for unittest access to complete, timeout 600, outp
```

```

07/14 19:01:02 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:03 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:03 INFO | unittest:0052| Running emulator
07/14 19:01:03 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:05 INFO | unittest:0096| Waiting for unittest emulator to complete, timeout 600, ou
07/14 19:01:06 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:07 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:07 INFO | unittest:0052| Running hypercall
07/14 19:01:07 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:08 INFO | unittest:0096| Waiting for unittest hypercall to complete, timeout 600, o
07/14 19:01:08 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:09 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:09 INFO | unittest:0052| Running idt_test
07/14 19:01:09 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:10 INFO | unittest:0096| Waiting for unittest idt_test to complete, timeout 600, ou
07/14 19:01:10 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:11 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:11 INFO | unittest:0052| Running msr
07/14 19:01:11 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:12 INFO | unittest:0096| Waiting for unittest msr to complete, timeout 600, output
07/14 19:01:13 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:13 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:13 INFO | unittest:0052| Running port80
07/14 19:01:13 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:14 INFO | unittest:0096| Waiting for unittest port80 to complete, timeout 600, outp
07/14 19:01:31 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:32 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:32 INFO | unittest:0052| Running realmode
07/14 19:01:32 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:33 INFO | unittest:0096| Waiting for unittest realmode to complete, timeout 600, ou
07/14 19:01:33 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:01:34 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:01:34 INFO | unittest:0052| Running sieve
07/14 19:01:34 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:01:35 INFO | unittest:0096| Waiting for unittest sieve to complete, timeout 600, output
07/14 19:02:05 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:05 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:05 INFO | unittest:0052| Running tsc
07/14 19:02:05 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:02:06 INFO | unittest:0096| Waiting for unittest tsc to complete, timeout 600, output
07/14 19:02:06 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:07 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:07 INFO | unittest:0052| Running xsave
07/14 19:02:07 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vml' -nodefaults -vga std -monitor unix:'/tmp/monitor-
07/14 19:02:08 INFO | unittest:0096| Waiting for unittest xsave to complete, timeout 600, output
07/14 19:02:09 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:09 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:09 INFO | unittest:0052| Running rmap_chain
07/14 19:02:09 INFO | kvm_vm:0782| Running qemu command:

```

```

/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix: /tmp/monitor-
07/14 19:02:11 INFO | unittest:0096| Waiting for unittest rmap_chain to complete, timeout 600,
07/14 19:02:12 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:13 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:13 INFO | unittest:0052| Running svm
07/14 19:02:13 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix: /tmp/monitor-
07/14 19:02:13 INFO | aexpect:0783| (qemu) qemu: -enable-nesting: invalid option
07/14 19:02:13 INFO | aexpect:0783| (qemu) (Process terminated with status 1)
07/14 19:02:13 ERROR| unittest:0108| Exception happened during svm: VM creation command failed:
07/14 19:02:13 ERROR| unittest:0115| Not possible to collect logs
07/14 19:02:13 INFO | unittest:0052| Running svm-disabled
07/14 19:02:13 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix: /tmp/monitor-
07/14 19:02:14 INFO | unittest:0096| Waiting for unittest svm-disabled to complete, timeout 600
07/14 19:02:15 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:16 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:16 INFO | unittest:0052| Running kvmclock_test
07/14 19:02:16 INFO | kvm_vm:0782| Running qemu command:
/usr/local/autotest/tests/kvm/qemu -name 'vm1' -nodefaults -vga std -monitor unix: /tmp/monitor-
07/14 19:02:17 INFO | unittest:0096| Waiting for unittest kvmclock_test to complete, timeout 60
07/14 19:02:33 INFO | aexpect:0783| (qemu) (Process terminated with status 0)
07/14 19:02:34 INFO | unittest:0113| Unit test log collected and available under /usr/local/aut
07/14 19:02:34 ERROR| kvm:0094| Test failed: TestFail: Unit tests failed: apic svm

```

You might take a look at the `unittests.cfg` config file options to do some tweaking you might like, such as making the timeout to consider a unittest as failed smaller and other things.

Please give us feedback on whether this procedure was helpful - email me at lmr AT redhat DOT com.

virttest

virttest package

Subpackages

virttest.libvirt_xml package

Subpackages

virttest.libvirt_xml.devices package

Submodules

virttest.libvirt_xml.devices.address module Address device / device descriptor class

<http://libvirt.org/formatdomain.html#elementsAddress>

```

class virttest.libvirt_xml.devices.address.Address (type_name, virsh_instance=<module
                                                    'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.devices.base.TypedDeviceBase`

attrs

```
classmethod new_from_dict (attributes, virsh_instance=<module 'virttest.virsh' from  
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

```
classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from  
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

virttest.libvirt_xml.devices.base module Common base classes for devices

```
class virttest.libvirt_xml.devices.base.StubDeviceMeta (mcs, name, bases, dct)  
    Bases: type
```

Metaclass for generating stub Device classes where not fully implemented yet

warning_issued = False

```
class virttest.libvirt_xml.devices.base.TypedDeviceBase (device_tag, type_name,  
                                                           virsh_instance=<module  
                                                           'virttest.virsh' from  
                                                           '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                                                           test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.base.UntypedDeviceBase`

Base class implementing common functions for all device XML w/o a type attr.

```
classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from  
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Hides type_name from superclass new_from_element().

type_name

```
class virttest.libvirt_xml.devices.base.UntypedDeviceBase (device_tag,  
                                                           virsh_instance=<module  
                                                           'virttest.virsh' from  
                                                           '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                                                           test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Base class implementing common functions for all device XML w/o a type attr.

device_tag

from_element (element)

Stateful component to helper method for new_from_element.

```
classmethod new_from_dict (properties, virsh_instance=<module 'virttest.virsh' from  
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Create a new device XML instance from a dict-like object

```
classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from  
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Create a new device XML instance from an single ElementTree element

virttest.libvirt_xml.devices.channel module Classes to support XML for channel devices

<http://libvirt.org/formatdomain.html#elementCharSerial>

```
class virttest.libvirt_xml.devices.channel.Channel (type_name='unix',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.character.CharacterBase
    address
    alias
    source
    target
```

virttest.libvirt_xml.devices.character module Generic character device support for serial, parallel, channel, and console

<http://libvirt.org/formatdomain.html#elementCharSerial>

```
class virttest.libvirt_xml.devices.character.CharacterBase (device_tag, type_name,
                                                            virsh_instance=<module
                                                            'virttest.virsh'
                                                            from
                                                            '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                            test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase
    add_source (**attributes)
        Convenience method for appending a source from dictionary of attributes
    add_target (**attributes)
        Convenience method for appending a target from dictionary of attributes
    del_sources ()
        Remove the list of dictionaries containing each source's attributes.
    del_targets ()
        Remove the list of dictionaries containing each target's attributes.
    get_sources ()
        Return a list of dictionaries containing each source's attributes.
    get_targets ()
        Return a list of dictionaries containing each target's attributes.
    set_sources (value)
        Set all sources to the value list of dictionaries of source attributes.
    set_targets (value)
        Set all sources to the value list of dictionaries of target attributes.
    sources
    targets
    update_source (index, **attributes)
        Convenience method for merging values into a source's attributes
    update_target (index, **attributes)
        Convenience method for merging values into a target's attributes
```

virttest.libvirt_xml.devices.console module Console device support class(es)

<http://libvirt.org/formatdomain.html#elementCharSerial>

```
class virttest.libvirt_xml.devices.console.Console (type_name='pty',
                                                    virsh_instance=<module
                                                    'virttest.virsh'          from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.character.CharacterBase`

static marshal_from_sources (item, index, libvirtxml)

Convert a dict to console source attributes.

static marshal_to_sources (tag, attr_dict, index, libvirtxml)

Convert a source tag and attributes to a dict.

protocol_type

sources

target_port

target_type

virttest.libvirt_xml.devices.controller module controller device support class(es)

<http://libvirt.org/formatdomain.html#elementsControllers>

```
class virttest.libvirt_xml.devices.controller.Controller (type_name,
                                                          virsh_instance=<module
                                                          'virttest.virsh'          from
                                                          '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                          test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.base.TypedDeviceBase`

```
class Address (type_name,          virsh_instance=<module          'virttest.virsh'          from
          '/home/docs/checkouts/readthedocs.org/user_builds/virt-
          test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.base.TypedDeviceBase`

attrs

```
classmethod new_from_dict (attributes,    virsh_instance=<module    'virttest.virsh'    from
          '/home/docs/checkouts/readthedocs.org/user_builds/virt-
          test/checkouts/latest/virttest/virsh.pyc'>)
```

```
classmethod new_from_element (element,    virsh_instance=<module    'virttest.virsh'    from
          '/home/docs/checkouts/readthedocs.org/user_builds/virt-
          test/checkouts/latest/virttest/virsh.pyc'>)
```

`Controller.address`

`Controller.driver`

`Controller.index`

`Controller.model`

`Controller.new_controller_address` (***dargs*)

Return a new controller Address instance and set properties from dargs

`Controller.pcihole64`

`Controller.ports`

`Controller.type`

`Controller.vectors`

virttest.libvirt_xml.devices.disk module disk device support class(es)

<http://libvirt.org/formatdomain.html#elementsDisks>

```
class virttest.libvirt_xml.devices.disk.Disk (type_name='file',      virsh_instance=<module
                                                'virttest.virsh'          from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.base.TypedDeviceBase`

Disk device XML class

Properties:

device: string, how exposed to guest

rawio: string (yes/no), disk needs rawio capability

sgio: string, “filtered” or “unfiltered”

snapshot: string, “yes”, “no”, “internal” or “external”

wwn: string.

serial: string.

vendor: string.

product: string.

driver: dict, keys: name, type, cache, error_policy, io, ioeventfd, event_idx, copy_on_read, discard

target: dict, keys: dev, bus, tray

blockio: dict, keys: logical_block_size, physical_block_size

geometry: dict, keys: cyls, heads, secs, trans

address: `libvirt_xml.devices.Address` instance

boot: string, boot order number to use if not using boot in os element

readonly: bool, True/False

transient: bool, True/False

share: bool, True/False

mirror: bool, read-only, True if block copy started

ready: bool, read-only, True if disk ready for pivot

iotune: `libvirt_xml.devices.Disk.IOTune` instance

source: `libvirt_xml.devices.Disk.DiskSource` instance

encryption: `libvirt_xml.devices.Disk.Encryption` instance.

```
class Address (type_name,      virsh_instance=<module      'virttest.virsh'      from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.devices.base.TypedDeviceBase`

attrs

```
classmethod new_from_dict (attributes, virsh_instance=<module 'virttest.virsh' from
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                             test/checkouts/latest/virttest/virsh.pyc'>)

classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

class `Disk.Auth` (`virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>`, `auth_user=''`)
Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`
Auth device XML class
Properties:

- auth_user**: string, attribute of auth tag
- secret_type**: string, attribute of secret tag, sub-tag of the auth tag
- secret_uuid**: string, attribute of secret tag, sub-tag of the auth tag
- secret_usage**: string, attribute of secret tag, sub-tag of the auth tag

auth_user
secret_type
secret_usage
secret_uuid

```
class Disk.DiskSource (virsh_instance=<module 'virttest.virsh' from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`
Disk source device XML class
Properties:

attrs: Dictionary of attributes, qualifying the disk type
seclabels: list of `libvirt_xml.devices.seclabel.Seclabel` instances
hosts: list of dictionaries describing network host properties

attrs
config_file
hosts

static marshal_from_host (*item, index, libvirtxml*)
Convert a dictionary into a tag + attributes

static marshal_from_seclabel (*item, index, libvirtxml*)
Convert a `Seclabel` instance into tag + attributes

static marshal_to_host (*tag, attr_dict, index, libvirtxml*)
Convert a tag + attributes into a dictionary

static marshal_to_seclabel (*tag, attr_dict, index, libvirtxml*)
Convert a tag + attributes into a `Seclabel` instance

seclabels
snapshot_name

```
class Disk.Encryption (virsh_instance=<module 'virttest.virsh' from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase
    Encryption device XML class
    Properties:
    encryption: string.
    secret: dict, keys: type, uuid
    encryption
    secret

class Disk.IO Tune (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase
    IO Tune device XML class
    Properties:
    total_bytes_sec: str(int) read_bytes_sec: str(int) write_bytes_sec: str(int) total_iops_sec: str(int)
    read_iops_sec: str(int) write_iops_sec: str(int)
    read_bytes_sec
    read_iops_sec
    total_bytes_sec
    total_iops_sec
    write_bytes_sec
    write_iops_sec

Disk.address
Disk.auth
Disk.blockio
Disk.boot
Disk.device
Disk.driver
Disk.encryption
Disk.geometry
Disk.iotune
Disk.mirror
Disk.new_auth (**dargs)
    Return a new disk auth instance and set properties from dargs
Disk.new_disk_address (type_name='drive', **dargs)
    Return a new disk Address instance and set properties from dargs
Disk.new_disk_source (**dargs)
    Return a new disk source instance and set properties from dargs
```

`Disk.new_encryption (**dargs)`
Return a new disk encryption instance and set properties from dargs

`Disk.new_iotune (**dargs)`
Return a new disk IOTune instance and set properties from dargs

`Disk.product`

`Disk.rawio`

`Disk.readonly`

`Disk.ready`

`Disk.serial`

`Disk.sgio`

`Disk.share`

`Disk.snapshot`

`Disk.source`

`Disk.target`

`Disk.transient`

`Disk.vendor`

`Disk.wwn`

virttest.libvirt_xml.devices.emulator module Support for the pseudo ‘emulator’ device XML

<http://libvirt.org/formatdomain.html#elementsDevices>

```
class virttest.libvirt_xml.devices.emulator.Emulator (virsh_instance=<module
                                                    'virttest.virsh'          from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    path
```

virttest.libvirt_xml.devices.filesystem module filesystem device support class(es)

<http://libvirt.org/formatdomain.html#elementsFilesystems>

`virttest.libvirt_xml.devices.filesystem.Filesystem`

virttest.libvirt_xml.devices.graphics module graphics framebuffer device support class(es)

<http://libvirt.org/formatdomain.html#elementsGraphics>

```
class virttest.libvirt_xml.devices.graphics.Graphics (type_name='vnc',
                                                    virsh_instance=<module
                                                    'virttest.virsh'          from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase
    add_channel (**attributes)
        Convenience method for appending channel from dictionary of attributes
```

static add_graphic (*vm_name*, *passwd=None*, *graphic='vnc'*, *add_channel=False*)

Add spice ssl or vnc graphic with passwd

Parameters

- **vm_name** – name of vm
- **passwd** – password for graphic
- **graphic** – graphic type, spice or vnc
- **add_channel** – add channel for spice

add_listens (***attributes*)

Convenience method for appending listens from dictionary of attributes

autoport

static change_graphic_type_passwd (*vm_name*, *graphic*, *passwd=None*)

Change the graphic type name and passwd

Parameters

- **vm_name** – name of vm
- **graphic** – graphic type, spice or vnc
- **passwd** – password for graphic

channel

defaultMode

del_channel ()

Remove the list of dictionaries containing each channel's attributes

static del_graphic (*vm_name*)

Del original graphic device

Parameters **vm_name** – name of vm

del_listens ()

Remove the list of dictionaries containing each listen's attributes

get_channel ()

Return a list of dictionaries containing each channel's attributes

get_listens ()

Return a list of dictionaries containing each listen's attributes

image_compression

jpeg_compression

listen

listen_addr

listen_type

listens

passwd

playback_compression

port

set_channel (*value*)

Set all channel to the value list of dictionaries of channel attributes

set_listens (*value*)

Set all listens to the value list of dictionaries of listen attributes

tlsPort

zlib_compression

virttest.libvirt_xml.devices.hostdev module hostdev device support class(es)

<http://libvirt.org/formatdomain.html#elementsHostDev>

```
class virttest.libvirt_xml.devices.hostdev.Hostdev (type_name='hostdev',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.base.TypedDeviceBase*

```
class SourceAddress (virsh_instance=<module
                        'virttest.virsh'
                        from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

```
class UntypedAddress (virsh_instance=<module
                        'virttest.virsh'
                        from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.base.UntypedDeviceBase*

bus

domain

function

slot

Hostdev.SourceAddress.**new_untyped_address** (***dargs*)

Hostdev.SourceAddress.**untyped_address**

Hostdev.**boot_order**

Hostdev.**hostdev_type**

Hostdev.**managed**

Hostdev.**mode**

Hostdev.**new_source_address** (***dargs*)

Hostdev.**source_address**

virttest.libvirt_xml.devices.hub module hub device support class(es)

<http://libvirt.org/formatdomain.html#elementsHub>

```
class virttest.libvirt_xml.devices.hub.Hub (type_name,
                                              virsh_instance=<module
                                              'virttest.virsh'
                                              from
                                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.base.TypedDeviceBase*

```

class Address (type_name,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase

    attrs

    classmethod new_from_dict (attributes,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)

    classmethod new_from_element (element,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)

Hub.address

Hub.new_hub_address (type_name='usb', **dargs)
    Return a new hub Address instance and set properties from dargs

```

virttest.libvirt_xml.devices.input module input device support class(es)

<http://libvirt.org/formatdomain.html#elementsInput>

```

class virttest.libvirt_xml.devices.input.Input (type_name,          virsh_instance=<module
                    'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase

    class Address (type_name,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase

        attrs

        classmethod new_from_dict (attributes,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)

        classmethod new_from_element (element,          virsh_instance=<module          'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)

Input.address

Input.input_bus

Input.new_input_address (type_name='usb', **dargs)
    Return a new input Address instance and set properties from dargs

```

virttest.libvirt_xml.devices.interface module interface device support class(es)

<http://libvirt.org/formatdomain.html#elementsNICS> <http://libvirt.org/formatnwfilter.html#nwfilterconceptsvars>

```

class virttest.libvirt_xml.devices.interface.Interface (type_name,
                    virsh_instance=<module
                    'virttest.virsh'          from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase

```

```
class Address (type_name, virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase
attrs

classmethod new_from_dict (attributes, virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)

classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)

class Interface.Bandwidth (virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.base.LibvirtXMLBase
Interface bandwidth xml class.
Properties:
inbound: dict. Keys: average, peak, floor, burst
outbound: dict. Keys: average, peak, floor, burst
inbound
outbound

class Interface.Driver (virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.base.LibvirtXMLBase
Interface Driver xml class.
Properties:
driver: dict.
host: dict. Keys: csum, gso, tso4, tso6, ecn, ufo
guest: dict. Keys: csum, gso, tso4, tso6, ecn, ufo
driver_attr
driver_guest
driver_host

class Interface.Filterref (virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
Bases: virttest.libvirt_xml.base.LibvirtXMLBase
Interface filterref xml class.
Properties:
name: string. filter name
parameters: list. parameters element dict list
static marshal_from_parameter (item, index, libvirtxml)
    Convert a dictionary into a tag + attributes
```


static marshal_to_parameter (*tag, attr_dict, index, libvirtxml*)

Convert a tag + attributes into a dictionary

name

parameters

Interface.**address**

Interface.**backend**

Interface.**bandwidth**

Interface.**boot_order**

Interface.**driver**

Interface.**filterref**

Interface.**link_state**

Interface.**mac_address**

Interface.**model**

Interface.**new_bandwidth** (***dargs*)

Return a new interface bandwidth instance from dargs

Interface.**new_driver** (***dargs*)

Return a new interface driver instance from dargs

Interface.**new_filterref** (***dargs*)

Return a new interface filterref instance from dargs

Interface.**new_iface_address** (***dargs*)

Return a new interface Address instance and set properties from dargs

Interface.**source**

Interface.**target**

Interface.**virtualport_type**

virttest.libvirt_xml.devices.lease module lease device support class(es)

<http://libvirt.org/formatdomain.html#elementsLease>

virttest.libvirt_xml.devices.lease.Lease

virttest.libvirt_xml.devices.librarian module Module to hide underlying device xml handler class implementation

virttest.libvirt_xml.devices.librarian.get (*name*)

Returns named device xml element's handler class

Parameters **name** – the device name

Returns the named device xml element's handler class

virttest.libvirt_xml.devices.memballoon module memballoon device support class(es)

<http://libvirt.org/formatdomain.html#elementsMemBalloon>

```
class virttest.libvirt_xml.devices.memballoon.Memballoon (virsh_instance=<module
                                                    'virttest.virsh'          from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    model
    stats_period
```

virttest.libvirt_xml.devices.memory module memory device support class(es)

```
class virttest.libvirt_xml.devices.memory.Memory (virsh_instance=<module
                                                    'virttest.virsh'          from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    class Address (type_name, virsh_instance=<module 'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
        Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase
        attrs
        classmethod new_from_dict (attributes, virsh_instance=<module 'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
        classmethod new_from_element (element, virsh_instance=<module 'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    class Memory.Source (virsh_instance=<module 'virttest.virsh' from
                            '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                            test/checkouts/latest/virttest/virsh.pyc'>)
        Bases: virttest.libvirt_xml.base.LibvirtXMLBase
        Memory source xml class.
        Properties:
        pagesize: int.
        pagesize_unit, nodemask: string.
        nodemask
        pagesize
        pagesize_unit
    class Memory.Target (virsh_instance=<module 'virttest.virsh' from
                            '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                            test/checkouts/latest/virttest/virsh.pyc'>)
        Bases: virttest.libvirt_xml.base.LibvirtXMLBase
        Memory target xml class.
        Properties:
```

size, node: int.
size_unit: string.
node
size
size_unit
Memory.address
Memory.mem_model
Memory.new_mem_address (*type_name='dimm', **dargs*)
Return a new disk Address instance and set properties from dargs
Memory.source
Memory.target

virttest.libvirt_xml.devices.panic module panic device support class(es)

<http://libvirt.org/formatdomain.html#elementsPanic>

```
class virttest.libvirt_xml.devices.panic.Panic (virsh_instance=<module
                                                'virttest.virsh'
                                                from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    addr_bus
    addr_controller
    addr_iobase
    addr_port
    addr_type
```

virttest.libvirt_xml.devices.parallel module Parallel device support class(es)

<http://libvirt.org/formatdomain.html#elementCharSerial>

```
class virttest.libvirt_xml.devices.parallel.Parallel (type_name='pty',
                                                       virsh_instance=<module
                                                       'virttest.virsh'
                                                       from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                       test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.character.CharacterBase
```

virttest.libvirt_xml.devices.redirdev module redirdev device support class(es)

<http://libvirt.org/formatdomain.html#elementsRedir>

virttest.libvirt_xml.devices.redirdev.Redirdev

virttest.libvirt_xml.devices.rng module random number generator device support class(es)

<http://libvirt.org/formatdomain.html#elementsRng>

```
class virttest.libvirt_xml.devices.rng.Rng(virsh_instance=<module 'virttest.virsh' from
                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.base.UntypedDeviceBase*

```
class Backend(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Rng backend xml class.

Properties:

model: string. backend model

type: string. backend type

backend_dev

backend_model

backend_protocol

backend_type

static marshal_from_source(item, index, libvirtxml)

Convert a dictionary into a tag + attributes

static marshal_to_source(tag, attr_dict, index, libvirtxml)

Convert a tag + attributes into a dictionary

source

Rng.**backend**

Rng.**rate**

Rng.**rng_model**

virttest.libvirt_xml.devices.seclabel module seclabel device support class(es)

<http://libvirt.org/formatdomain.html#seclabel>

```
class virttest.libvirt_xml.devices.seclabel.Seclabel(type_name='dynamic',
                                                    virsh_instance=<module
                                                    'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.base.TypedDeviceBase*

Seclabel XML class

Properties:

model: string, security driver model

relabel: string, 'yes' or 'no'

baselabel: string, base label string

label: string, the sec label string

baselabel

label
model
relabel

virttest.libvirt_xml.devices.serial module Classes to support XML for serial devices

<http://libvirt.org/formatdomain.html#elementCharSerial>

```
class virttest.libvirt_xml.devices.serial.Serial(type_name='pty',
                                                  virsh_instance=<module
                                                  'virttest.virsh' from
                                                  '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                  test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.character.CharacterBase

    static marshal_from_sources(item, index, libvirtxml)
        Convert a dict to serial source attributes.

    static marshal_to_sources(tag, attr_dict, index, libvirtxml)
        Convert a source tag and attributes to a dict.

    protocol_type
    sources
    target_port
    target_type
```

virttest.libvirt_xml.devices.smartcard module smartcard device support class(es)

<http://libvirt.org/formatdomain.html#elementsSmartcard>

```
class virttest.libvirt_xml.devices.smartcard.Smartcard(type_name='spicevmc',
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase

    address
    address_controller
    address_slot
    address_type
    protocol
    protocol_type
    smartcard_mode
    smartcard_type
    source
    source_host
    source_mode
    source_service
```

virttest.libvirt_xml.devices.sound module sound device support class(es)

<http://libvirt.org/formatdomain.html#elementsSound>

```
class virttest.libvirt_xml.devices.sound.Sound (virsh_instance=<module
                                                'virttest.virsh'
                                                from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    address
    codec_type
    model_type
```

virttest.libvirt_xml.devices.video module video device support class(es)

<http://libvirt.org/formatdomain.html#elementsVideo>

```
class virttest.libvirt_xml.devices.video.Video (type_name, virsh_instance=<module
                                                'virttest.virsh'
                                                from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.TypedDeviceBase
    acceleration
    address
    model_heads
    model_ram
    model_type
    model_vram
    primary
```

virttest.libvirt_xml.devices.watchdog module watchdog device support class(es)

<http://libvirt.org/formatdomain.html#elementsWatchdog>

```
class virttest.libvirt_xml.devices.watchdog.Watchdog (virsh_instance=<module
                                                'virttest.virsh'
                                                from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.devices.base.UntypedDeviceBase
    action
    address
    model_type
```

Module contents

virttest.libvirt_xml.nwfilter_protocols package

Submodules

virttest.libvirt_xml.nwfilter_protocols.ah module ah protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMisc>

```
class virttest.libvirt_xml.nwfilter_protocols.ah.Ah (type_name='file',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase*

Create new Ah xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Ah.Attr instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Ah attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address sr-
cipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

srcipto

srcmacaddr

srcmacmask

state

Ah.attrs

Ah.get_attr()

Return ah attribute dict

Returns None if no ah in xml, dict of ah's attributes.

Ah.new_attr(dargs)**

Return a new Attr instance and set properties from dargs

Parameters dargs – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.ah_ipv6 module ah-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMiscv6>

```
class virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6(type_name='file',  
                                                             virsh_instance=<module  
                                                             'virttest.virsh' from  
                                                             '/home/docs/checkouts/readthedocs.org/user_  
                                                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Ah_ipv6 xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Ah_ipv6.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Ah_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address sr-
cipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset


```

    ipsetflags
    srcipaddr
    srcipfrom
    srcipmask
    srcipto
    srcmacaddr
    srcmacmask
    state
Ah_ipv6.attrs
Ah_ipv6.get_attr()
    Return ah-ipv6 attribute dict

    Returns None if no ah-ipv6 in xml, dict of ah-ipv6's attributes.
Ah_ipv6.new_attr(**dargs)
    Return a new Attr instance and set properties from dargs

    Parameters dargs – dict of attributes

    Returns new Attr instance

```

virttest.libvirt_xml.nwfilter_protocols.all module all protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMisc>

```

class virttest.libvirt_xml.nwfilter_protocols.all.All(type_name='file',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new All xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.All.Attr instance

```

class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

All attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address
srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp
dstipaddr
dstipfrom
dstipmask
dstipto
dstmacaddr
dstmacmask
ipset
ipsetflags
srcipaddr
srcipfrom
srcipmask
srcipto
srcmacaddr
srcmacmask
state

`All.attrs`

`All.get_attr()`

Return 'all' attribute dict

Returns None if no 'all' in xml, dict of all's attributes.

`All.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.all_ipv6 module all-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMiscv6>

```
class virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6(type_name='file',
                                                                virsh_instance=<module
                                                                'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new All_ipv6 xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.All_ipv6.Attr` instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

All_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
 dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of destination
 srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
 dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address
 srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
 dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destination IP address
 comment: string, text with max. 256 characters state: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

srcipto

srcmacaddr

srcmacmask

state

`All_ipv6.attrs`

`All_ipv6.get_attr()`

Return all-ipv6 attribute dict

Returns None if no all-ipv6 in xml, dict of all-ipv6's attributes.

`All_ipv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.arp module arp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoARP>

```
class virttest.libvirt_xml.nwfilter_protocols.arp.Arp(type_name='file',
                                                       virsh_instance=<module
                                                       'virttest.virsh'
                                                       from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                       test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Arp xml instances

Properties:

attrs: libvirt_xml.nwfilter_protocols.Arp.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Arp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacaddr: string, Mask applied to MAC address of
destination hwtype: string, Hardware type protocoltype: string, Protocol type opcode: string, Opcode arp-
srcmacaddr: string, Source MAC address in ARP/RARP packet arpdstmacaddr: string, Destination MAC
address in ARP/RARP packet arpsrcipaddr: string, Source IP address in ARP/RARP packet arpdstipaddr:
string, Destination IP address in ARP/RARP packet comment: string, text with max. 256 characters gra-
tuitous: string, boolean indicating whether to check for gratuitous ARP packet

arpdstipaddr

arpdstmacaddr

arpsrcipaddr

arpsrcmacaddr

comment

dstmacaddr

dstmacmask

gratuitous

hwtype

opcode

protocoltype

srcmacaddr

srcmacmask

Arp.**attrs**

Arp.**get_attr**()

Return arp attribute dict

Returns None if no arp in xml, dict of arp's attributes.

Arp.**new_attr**(**dargs)

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.base module Common base classes for filter rule protocols

```
class virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase(protocol_tag,
                                                                    type_name,
                                                                    virsh_instance=<module
                                                                    'virttest.virsh'
                                                                    from
                                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase`

Base class implementing common functions for all filter rule XML w/o a type attr.

```
classmethod new_from_element(element, virsh_instance=<module 'virttest.virsh' from
                                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Hides type_name from superclass new_from_element().

type_name

```
class virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase(protocol_tag,
                                                                    virsh_instance=<module
                                                                    'virttest.virsh'
                                                                    from
                                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Base class implementing common functions for all rule protocol XML w/o a type attr.

from_element (*element*)

Stateful component to helper method for new_from_element.

```
classmethod new_from_dict(properties, virsh_instance=<module 'virttest.virsh' from
                                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Create a new filter rule XML instance from a dict-like object

```
classmethod new_from_element(element, virsh_instance=<module 'virttest.virsh' from
                                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Create a new filter rule XML instance from an single ElementTree element

protocol_tag

virttest.libvirt_xml.nwfilter_protocols.esp module esp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMisc>

```
class virttest.libvirt_xml.nwfilter_protocols.esp.Esp(type_name='file',
                                                       virsh_instance=<module
                                                       'virttest.virsh'
                                                       from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Esp xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Esp.Attr instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

```
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase
```

Esp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address sr-
cipfrom: string, Start of range of source IP address scripto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

scripto

srcmacaddr

srcmacmask

state

Esp.attrs

Esp.get_attr()

Return esp attribute dict

Returns None if no esp in xml, dict of esp's attributes.

Esp.new_attr(dargs)**

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.esp_ipv6 module esp-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMiscv6>

```
class virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6(type_name='file',
                                                                virsh_instance=<module
                                                                'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase*

Create new Esp_ipv6 xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Esp_ipv6.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Esp_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address
srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

srcipto

srcmacaddr

srcmacmask

state

`Esp_ipv6.attrs`

`Esp_ipv6.get_attr()`

Return esp-ipv6 attribute dict

Returns None if no esp-ipv6 in xml, dict of esp-ipv6's attributes.

`Esp_ipv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.icmp module icmp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoICMP>

```
class virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp (type_name='file',
                                                         virsh_instance=<module
                                                         'virttest.virsh'          from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Icmp xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Icmp.Attr` instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Icmp attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcmacmask`: string, Mask applied to MAC address of sender
`dstmacaddr`: string, MAC address of destination `dstmacmask`: string, Mask applied to MAC address of destination
`srcipaddr`: string, Source IP address `srcipmask`: string, Mask applied to source IP address `dstipaddr`: string, Destination IP address `dstipmask`: string, Mask applied to destination IP address
`srcipfrom`: string, Start of range of source IP address `srcipto`: string, End of range of source IP address `dstipfrom`: string, Start of range of destination IP address `dstipto`: string, End of range of destination IP address
`type`: string, ICMP type code: string, ICMP code comment: string, text with max. 256 characters
`state`: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE
`ipset`: The name of an IPSet managed outside of libvirt
`ipsetflags`: flags for the IPSet; requires `ipset` attribute

`code`

`comment`

`dscp`

`dstipaddr`

`dstipfrom`

`dstipmask`

`dstipto`

`dstmacaddr`

`dstmacmask`


```

ipset
ipsetflags
srcipaddr
srcipfrom
srcipmask
srcipto
srcmacaddr
srcmacmask
state
type

```

`Icmp.attrs`

`Icmp.get_attr()`

Return icmp attribute dict

Returns None if no icmp in xml, dict of icmp's attributes.

`Icmp.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.icmpv6 module icmpv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoICMPv6>

```

class virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6(type_name='file',
                                                            virsh_instance=<module
                                                            'virttest.virsh' from
                                                            '/home/docs/checkouts/readthedocs.org/user_bui
                                                            test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Icmpv6 xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Icmpv6.Attr` instance

```

class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Icmpv6 attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcipaddr`: string, Source IPv6 address `srcipmask`: string, Mask applied to source IPv6 address `dstipaddr`: string, Destination IPv6 address `dstipmask`: string, Mask applied to destination IPv6 address `srcipfrom`: string, Start of range of source IP address `srcipto`: string, End of range of source IP address `dstipfrom`: string, Start of range of destination IP address `dstipto`: string, End of range of destination IP address `type`: string, ICMPv6 type code: string, ICMPv6 code comment: string, text with max. 256 characters `state`: string, comma separated list of NEW, ESTABLISHED, RELATED, INVALID or NONE `ipset`: The name of an IPSet managed outside of libvirt `ipsetflags`: flags for the IPSet; requires `ipset` attribute

`code`
`comment`
`dscp`
`dstipaddr`
`dstipfrom`
`dstipmask`
`dstipto`
`ipset`
`ipsetflags`
`srcipaddr`
`srcipfrom`
`srcipmask`
`srcipto`
`srcmacaddr`
`state`
`type`

`Icmpv6.attrs`

`Icmpv6.get_attr()`

Return icmpv6 attribute dict

Returns None if no icmpv6 in xml, dict of icmpv6's attributes.

`Icmpv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.igmp module igmp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMisc>

```
class virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp(type_name='file',
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Igmp xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Igmp.Attr` instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Igmp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address sr-
cipfrom: string, Start of range of source IP address scripto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

scripto

srcmacaddr

srcmacmask

state

`Igmp.attrs`

`Igmp.get_attr()`

Return igmp attribute dict

Returns None if no igmp in xml, dict of igmp's attributes.

`Igmp.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.ip module ipv4 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoIP>

```
class virttest.libvirt_xml.nwfilter_protocols.ip.Ip (type_name='file',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Ip xml instances

Properties:

attrs: `libvirt_xml.nwfilter_protocols.Ip.Attr` instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Ip attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacaddr: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address dsti-
paddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address ip_protocol:
string, Layer 4 protocol identifier srcportstart: string, Start of range of valid source ports; requires protocol
srcportend: string, End of range of valid source ports; requires protocol dstportstart: string, Start of range
of valid destination ports; requires protocol dstportend: string, End of range of valid destination ports;
requires protocol comment: string, text with max. 256 characters

comment

dscp

dstipaddr

dstipmask

dstmacaddr

dstmacmask

dstportend

dstportstart

ip_protocol

srcipaddr

srcipmask

srcmacaddr

srcmacmask

srcportend

srcportstart

`Ip.attrs`

`Ip.get_attr()`

Return ip attribute dict

Returns None if no ip in xml, dict of ip's attributes.

`Ip.new_attr (**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.ipv6 module ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoIPv6>

```
class virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6(type_name='file',
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Ipv6 xml instances

Properties:

attrs: `libvirt_xml.nwfilter_protocols.Ipv6.Attr` instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacaddr: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address dsti-
paddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address ip_protocol:
string, Layer 4 protocol identifier srcportstart: string, Start of range of valid source ports; requires protocol
srcportend: string, End of range of valid source ports; requires protocol dstportstart: string, Start of range
of valid destination ports; requires protocol dstportend: string, End of range of valid destination ports;
requires protocol comment: string, text with max. 256 characters

comment

dscp

dstipaddr

dstipmask

dstmacaddr

dstmacmask

dstportend

dstportstart

ip_protocol

srcipaddr

srcipmask

srcmacaddr

srcmacmask

srcportend
srcportstart
Ipv6.attrs
Ipv6.get_attr()
Return ipv6 attribute dict
Returns None if no ipv6 in xml, dict of ipv6's attributes.
Ipv6.new_attr(dargs)**
Return a new Attr instance and set properties from dargs
Parameters **dargs** – dict of attributes
Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.librarian module Module to hide underlying filter protocol xml handler class implementation

virttest.libvirt_xml.nwfilter_protocols.librarian.get(name)
Returns named filter protocol xml element's handler class
Parameters **name** – the filter protocol name
Returns named filter protocol xml element's handler class

virttest.libvirt_xml.nwfilter_protocols.mac module mac protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMAC>

class `virttest.libvirt_xml.nwfilter_protocols.mac.Mac` (*type_name='file',
virsh_instance=<module
'virttest.virsh' from
'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Mac xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Mac.Attr` instance

class `Attr` (*virsh_instance=<module 'virttest.virsh' from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Mac attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcmacmask`: string, Mask applied to MAC address of sender
`dstmacaddr`: string, MAC address of destination `dstmacmask`: string, Mask applied to MAC address of destination
`protocolid`: string, Layer 3 protocol ID `comment`: string, text with max. 256 characters

comment

dstmacaddr

dstmacmask

protocolid

srcmacaddr

srcmacmask

Mac.attrs

Mac.get_attr()
Return mac attribute dict

Returns None if no mac in xml, dict of mac's attributes.

Mac.new_attr(dargs)**
Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.rarp module rarp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoARP>

```
class virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp(type_name='file',
                                                       virsh_instance=<module
                                                       'virttest.virsh' from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                       test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Rarp xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Rarp.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Rarp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination hwtype: string, Hardware type protocoltype: string, Protocol type opcode: string, Opcode arp-
srcmacaddr: string, Source MAC address in ARP/RARP packet arpdstmacaddr: string, Destination MAC
address in ARP/RARP packet arpsrcipaddr: string, Source IP address in ARP/RARP packet arpdstipaddr:
string, Destination IP address in ARP/RARP packet comment: string, text with max. 256 characters gra-
tuitous: string, boolean indicating whether to check for gratuitous ARP packet

arpdstipaddr

arpdstmacaddr

arpsrcipaddr

arpsrcmacaddr

comment

dstmacaddr

dstmacmask

gratuitous

hwtype

opcode

protocoltype
srcmacaddr
srcmacmask

Rarp.attrs

Rarp.get_attr()

Return rarp attribute dict

Returns None if no rarp in xml, dict of rarp's attributes.

Rarp.new_attr(dargs)**

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.sctp module sctp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv4>

```
class virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp(type_name='file',
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Sctp xml instances

Properties:

attrs: libvirt_xml.nwfilter_protocols.Sctp.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Sctp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destination IP address srcportstart: string, Start of range of valid source ports; requires protocol srcportend: string, End of range of valid source ports; requires protocol dstportstart: string, Start of range of valid destination ports; requires protocol dstportend: string, End of range of valid destination ports; requires protocol comment: string, text with max. 256 characters state: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask


```

dstipto
dstportend
dstportstart
ipset
ipsetflags
srcipaddr
srcipfrom
srcipmask
srcipto
srcmacaddr
srcportend
srcportstart
state
Sctp.attrs
Sctp.get_attr()
    Return sctp attribute dict

    Returns None if no sctp in xml, dict of sctp's attributes.

Sctp.new_attr(**dargs)
    Return a new Attr instance and set properties from dargs

    Parameters dargs – dict of attributes

    Returns new Attr instance

```

virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6 module sctp-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv6>

```

class virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6(type_name='file',
                                                                    virsh_instance=<module
                                                                    'virttest.virsh'
                                                                    from
                                                                    '/home/docs/checkouts/readthedocs.org
                                                                    test/checkouts/latest/virttest/virsh.pyc'>
                                                                    )
    Bases: virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase

```

Create new Sctp_ipv6 xml instances

Properties:

attrs: libvirt_xml.nwfilter_protocols.Sctp_ipv6.Attr instance

```

class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase

```

Sctp_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address srcipfrom: string, Start of range of source IP address scripto: string, End of range of source IP address dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destination IP address srcportstart: string, Start of range of valid source ports; requires protocol srcportend: string, End of range of valid source ports; requires protocol dstportstart: string, Start of range of valid destination ports; requires protocol dstportend: string, End of range of valid destination ports; requires protocol comment: string, text with max. 256 characters state: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstportend

dstportstart

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

scripto

srcmacaddr

srcportend

srcportstart

state

`Sctp_ipv6.attrs`

`Sctp_ipv6.get_attr()`

Return sctp-ipv6 attribute dict

Returns None if no sctp-ipv6 in xml, dict of sctp-ipv6's attributes.

`Sctp_ipv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.stp module stp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoSTP>

```
class virttest.libvirt_xml.nwfilter_protocols.stp.Stp(type_name='file',
                                                    virsh_instance=<module
                                                    'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Stp xml instances

Properties: attrs: `libvirt_xml.nwfilter_protocols.Stp.Attr` instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Stp attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
 type: string, Bridge Protocol Data Unit (BPDU) type flags: string, BPDU flag root_priority: string, Root
 priority (range start) root_priority_hi: string, Root priority range end root_address: string, Root MAC
 address root_address_mask: string, BPDU sender MAC address root_cost: string, Root path cost (range
 start) root_cost_hi: string, Root path cost range end sender_priority: string, Sender priority (range start)
 sender_priority_hi: string, Sender priority range end sender_address: string, BPDU sender MAC address
 sender_address_mask: string, BPDU sender MAC address mask port: string, Port identifier (range start)
 port_hi: string, Port identifier range end msg_age: string, Message age timer (range start) msg_age_hi:
 string, Message age timer range end max_age: string, Maximum age timer (range start) max_age_hi:
 string, Maximum age timer range end hello_time: string, Hello time timer (range start) hello_time_hi:
 string, Hello time timer range end forward_delay: string, Forward delay (range start) forward_delay_hi:
 string, Forward delay range end comment: string, text with max. 256 characters

comment

flags

forward_delay

forward_delay_hi

hello_time

hello_time_hi

max_age

max_age_hi

msg_age

msg_age_hi

port

port_hi

root_address

root_address_mask

root_cost

root_cost_hi

root_priority

```
    root_priority_hi
    sender_address
    sender_address_mask
    sender_priority
    sender_priority_hi
    srcmacaddr
    srcmacmask
    type
```

`Stp.attrs`

`Stp.get_attr()`

Return stp attribute dict

Returns None if no stp in xml, dict of stp's attributes.

`Stp.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.tcp module tcp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv4>

```
class virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp (type_name='file',
                                                       virsh_instance=<module
                                                       'virttest.virsh'
                                                       from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                       test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Tcp xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Tcp.Attr` instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Tcp attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcipaddr`: string, Source IP address `srcipmask`: string, Mask applied to source IP address `dstipaddr`: string, Destination IP address `dstipmask`: string, Mask applied to destination IP address `srcipfrom`: string, Start of range of source IP address `srcipto`: string, End of range of source IP address `dstipfrom`: string, Start of range of destination IP address `dstipto`: string, End of range of destination IP address `srcportstart`: string, Start of range of valid source ports; requires protocol `srcportend`: string, End of range of valid source ports; requires protocol `dstportstart`: string, Start of range of valid destination ports; requires protocol `dstportend`: string, End of range of valid destination ports; requires protocol `comment`: string, text with max. 256 characters `state`: string, comma separated list of NEW,ESTABLISHED,RELATED, INVALID or NONE flags: string, TCP-only: format of mask/flags with mask and flags each being a comma separated list of SYN,ACK,URG,PSH,FIN,RST or NONE or

ALL ipset: The name of an IPSet managed outside of libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment
dscp
dstipaddr
dstipfrom
dstipmask
dstipto
dstportend
dstportstart
flags
ipset
ipsetflags
srcipaddr
srcipfrom
srcipmask
srcipto
srcmacaddr
srcportend
srcportstart
state

`Tcp.attrs`

`Tcp.get_attr()`

Return tcp attribute dict

Returns None if no tcp in xml, dict of tcp's attributes.

`Tcp.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6 module tcp-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv6>

```
class virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6(type_name='file',
                                                                virsh_instance=<module
                                                                'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user
                                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Tcp_ipv6 xml instances

Properties: `attrs: libvirt_xml.nwfilter_protocols.Tcp_ipv6.Attr` instance

class Attr (*virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Tcp_ipv6 attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcipaddr`: string, Source IP address `srcipmask`: string, Mask applied to source IP address `dstipaddr`: string, Destination IP address `dstipmask`: string, Mask applied to destination IP address `srcipfrom`: string, Start of range of source IP address `srcipto`: string, End of range of source IP address `dstipfrom`: string, Start of range of destination IP address `dstipto`: string, End of range of destination IP address `srcportstart`: string, Start of range of valid source ports; requires protocol `srcportend`: string, End of range of valid source ports; requires protocol `dstportstart`: string, Start of range of valid destination ports; requires protocol `dstportend`: string, End of range of valid destination ports; requires protocol `comment`: string, text with max. 256 characters `state`: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE flags: string, TCP-only: format of mask/flags with mask and flags each being a comma separated list of SYN,ACK,URG,PSH,FIN,RST or NONE or ALL `ipset`: The name of an IPSet managed outside of libvirt `ipsetflags`: flags for the IPSet; requires `ipset` attribute

`comment`

`dscp`

`dstipaddr`

`dstipfrom`

`dstipmask`

`dstipto`

`dstportend`

`dstportstart`

`flags`

`ipset`

`ipsetflags`

`srcipaddr`

`srcipfrom`

`srcipmask`

`srcipto`

`srcmacaddr`

`srcportend`

`srcportstart`

`state`

`Tcp_ipv6.attrs`

`Tcp_ipv6.get_attr()`

Return tcp-ipv6 attribute dict

Returns None if no tcp-ipv6 in xml, dict of tcp-ipv6's attributes.

`Tcp_ipv6.new_attr (**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.udp module udp protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv4>

```
class virttest.libvirt_xml.nwfilter_protocols.udp.Udp (type_name='file',
                                                       virsh_instance=<module
                                                       'virttest.virsh'
                                                       from
                                                       '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                       test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Udp xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Udp.Attr` instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Udp attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcipaddr`: string, Source IP address `srcipmask`: string, Mask applied to source IP address `dstipaddr`: string, Destination IP address `dstipmask`: string, Mask applied to destination IP address `srcipfrom`: string, Start of range of source IP address `srcipto`: string, End of range of source IP address `dstipfrom`: string, Start of range of destination IP address `dstipto`: string, End of range of destination IP address `srcportstart`: string, Start of range of valid source ports; requires protocol `srcportend`: string, End of range of valid source ports; requires protocol `dstportstart`: string, Start of range of valid destination ports; requires protocol `dstportend`: string, End of range of valid destination ports; requires protocol `comment`: string, text with max. 256 characters `state`: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE `ipset`: The name of an IPSet managed outside of libvirt `ipsetflags`: flags for the IPSet; requires `ipset` attribute

`comment`

`dscp`

`dstipaddr`

`dstipfrom`

`dstipmask`

`dstipto`

`dstportend`

`dstportstart`

`ipset`

`ipsetflags`

`srcipaddr`

```
srcipfrom
srcipmask
srcipto
srcmacaddr
srcportend
srcportstart
state
Udp.attrs
Udp.get_attr()
    Return udp attribute dict

    Returns None if no udp in xml, dict of udp's attributes.
Udp.new_attr(**dargs)
    Return a new Attr instance and set properties from dargs

    Parameters dargs – dict of attributes

    Returns new Attr instance
```

virttest.libvirt_xml.nwfilter_protocols.udp_ipv6 module udp-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoTCP-ipv6>

```
class virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6(type_name='file',
                                                                virsh_instance=<module
                                                                'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user_
                                                                builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Udp_ipv6 xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Udp_ipv6.Attr instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Udp_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destination IP address srcportstart: string, Start of range of valid source ports; requires protocol srcportend: string, End of range of valid source ports; requires protocol dstportstart: string, Start of range of valid destination ports; requires protocol dstportend: string, End of range of valid destination ports; requires protocol comment: string, text with max. 256 characters state: string, comma separated list of NEW,ESTABLISHED,RELATED,INVALID or NONE ipset: The name of an IPSet managed outside of libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment


```

dscp
dstipaddr
dstipfrom
dstipmask
dstipto
dstportend
dstportstart
ipset
ipsetflags
srcipaddr
srcipfrom
srcipmask
srcipto
srcmacaddr
srcportend
srcportstart
state

```

`Udp_ipv6.attrs`

`Udp_ipv6.get_attr()`

Return udp-ipv6 attribute dict

Returns None if no udp-ipv6 in xml, dict of udp-ipv6's attributes.

`Udp_ipv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters `dargs` – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.udplite module udplite protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMisc>

```

class virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite (type_name='file',
                                                             virsh_instance=<module
                                                             'virttest.virsh' from
                                                             '/home/docs/checkouts/readthedocs.org/user_
                                                             test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Udplite xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Udplite.Attr` instance

```

class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)

```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Udplite attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address
srcipfrom: string, Start of range of source IP address srcipto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED, INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

srcipto

srcmacaddr

srcmacmask

state

`Udplite.attrs`

`Udplite.get_attr()`

Return udplite attribute dict

Returns None if no udplite in xml, dict of udplite's attributes.

`Udplite.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6 module udplite-ipv6 protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoMiscv6>

```
class virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6 (type_name='file',  
                                                                    virsh_instance=<module  
                                                                    'virttest.virsh'  
                                                                    from  
                                                                    '/home/docs/checkouts/readthe  
                                                                    test/checkouts/latest/virttest/vir
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Udplite_ipv6 xml instances

Properties: attrs: libvirt_xml.nwfilter_protocols.Udplite_ipv6.Attr instance

```
class Attr (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Udplite_ipv6 attribute XML class

Properties:

srcmacaddr: string, MAC address of sender srcmacmask: string, Mask applied to MAC address of sender
dstmacaddr: string, MAC address of destination dstmacmask: string, Mask applied to MAC address of
destination srcipaddr: string, Source IP address srcipmask: string, Mask applied to source IP address
dstipaddr: string, Destination IP address dstipmask: string, Mask applied to destination IP address sr-
cipfrom: string, Start of range of source IP address scripto: string, End of range of source IP address
dstipfrom: string, Start of range of destination IP address dstipto: string, End of range of destina-
tion IP address comment: string, text with max. 256 characters state: string, comma separated list of
NEW,ESTABLISHED,RELATED, INVALID or NONE ipset: The name of an IPSet managed outside of
libvirt ipsetflags: flags for the IPSet; requires ipset attribute

comment

dscp

dstipaddr

dstipfrom

dstipmask

dstipto

dstmacaddr

dstmacmask

ipset

ipsetflags

srcipaddr

srcipfrom

srcipmask

scripto

srcmacaddr

srcmacmask

state

`Udplite_ipv6.attrs`

`Udplite_ipv6.get_attr()`

Return udplite-ipv6 attribute dict

Returns None if no udplite-ipv6 in xml, dict of udplite-ipv6's attributes.

`Udplite_ipv6.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

virttest.libvirt_xml.nwfilter_protocols.vlan module vlan protocol support class(es)

<http://libvirt.org/formatnwfilter.html#nwfelemsRulesProtoVLAN>

```
class virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan(type_name='file',
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase`

Create new Vlan xml instances

Properties: `attrs`: `libvirt_xml.nwfilter_protocols.Vlan.Attr` instance

```
class Attr(virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Vlan attribute XML class

Properties:

`srcmacaddr`: string, MAC address of sender `srcmacmask`: string, Mask applied to MAC address of sender
`dstmacaddr`: string, MAC address of destination `dstmacmask`: string, Mask applied to MAC address of destination
`vlanid`: string, VLAN ID `encap_protocol`: string, Encapsulated layer 3 protocol ID `comment`: string, text with max. 256 characters

comment

dstmacaddr

dstmacmask

encap_protocol

srcmacaddr

srcmacmask

vlanid

`Vlan.attrs`

`Vlan.get_attr()`

Return vlan attribute dict

Returns None if no vlan in xml, dict of vlan's attributes.

`Vlan.new_attr(**dargs)`

Return a new Attr instance and set properties from dargs

Parameters **dargs** – dict of attributes

Returns new Attr instance

Module contents

Submodules

virttest.libvirt_xml.accessors module Specializations of base.AccessorBase for particular XML manipulation types

class `virttest.libvirt_xml.accessors.AccessorBase` (*operation, property_name, libvirtxml, **dargs*)

Bases: `virttest.propcan.PropCanBase`

Base class for a callable operating on a LibvirtXMLBase subclass instance

element_by_parent (*parent_xpath, tag_name, create=True*)

Retrieve/create an element instance at parent_xpath/tag_name

Parameters

- **parent_xpath** – xpath of parent element
- **tag_name** – name of element under parent to retrieve/create
- **create** – True to create new element if not exist

Returns ElementTree.Element instance

Raise LibvirtXMLError: If element not exist & create=False

libvirtxml

operation

property_name

xmldatafile ()

Retrieve xmldatafile instance from libvirtxml instance

class `virttest.libvirt_xml.accessors.AccessorGeneratorBase` (*property_name, libvirtxml, forbidden=None, **dargs*)

Bases: `object`

Accessor method/class generator for specific property name

accessor_name (*operation*)

Return instance name for operation, defined by subclass (i.e. 'get_foo')

assign_callable (*operation, callable_inst*)

Set reference on objectified libvirtxml instance to callable_inst

static callable_name (*operation*)

Return class name for operation (i.e. 'Getter'), defined by subclass.

make_callable (*operation*)

Return an callable instance for operation

make_forbidden (*operation*)

Return a forbidden callable instance for operation

set_if_not_defined (*operation*)

Setup a callable instance for operation only if not already defined

```
class virttest.libvirt_xml.accessors.AllForbidden(property_name, libvirtxml)
    Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
    Class of forbidden accessor classes for those undefined on libvirtxml

class virttest.libvirt_xml.accessors.ForbiddenBase(operation, property_name, libvirtxml,
                                                    **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Raise LibvirtXMLAccessorError when called w/ or w/o a value arg.

class virttest.libvirt_xml.accessors.XMLAttribute(property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None, attribute=None)
    Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
    Class of accessor classes operating on an attribute of an element

class Delter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Remove attribute

    attribute
    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name

class XMLAttribute.Getter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Get attribute value

    attribute
    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name

class XMLAttribute.Setter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Set attribute value

    attribute
    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name
```

```
class virttest.libvirt_xml.accessors.XMLElementBool (property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None)
```

Bases: `virttest.libvirt_xml.accessors.AccessorGeneratorBase`

Class of accessor classes operating purely element existence

```
class Deltter (operation, property_name, libvirtxml, **dargs)
```

Bases: `virttest.libvirt_xml.accessors.AccessorBase`

Remove element and ignore if it doesn't exist (same as False)

libvirtxml

operation

parent_xpath

property_name

tag_name

```
class XMLElementBool.Getter (operation, property_name, libvirtxml, **dargs)
```

Bases: `virttest.libvirt_xml.accessors.AccessorBase`

Retrieve text on element

libvirtxml

operation

parent_xpath

property_name

tag_name

```
class XMLElementBool.Setter (operation, property_name, libvirtxml, **dargs)
```

Bases: `virttest.libvirt_xml.accessors.AccessorBase`

Create element when True, delete when false

libvirtxml

operation

parent_xpath

property_name

tag_name

`XMLElementBool.required_dargs = ('parent_xpath', 'tag_name')`

```
class virttest.libvirt_xml.accessors.XMLElementDict (property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None)
```

Bases: `virttest.libvirt_xml.accessors.AccessorGeneratorBase`

Class of accessor classes operating as a dictionary of attributes

```
class Deltter (operation, property_name, libvirtxml, **dargs)
```

Bases: `virttest.libvirt_xml.accessors.AccessorBase`

Remove element and ignore if it doesn't exist (same as False)

libvirtxml

operation

```
parent_xpath
property_name
tag_name

class XMLElementDict.Getter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Retrieve attributes on element

    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name

class XMLElementDict.Setter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Set attributes to value on element

    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name

class virttest.libvirt_xml.accessors.XMLElementInt(property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None, radix=10)
    Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
    Class of accessor classes operating on element.text as an integer

class Delter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Remove element and ignore if it doesn't exist (same as False)

    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name

class XMLElementInt.Getter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Retrieve text on element and convert to int

    libvirtxml
    operation
    parent_xpath
    property_name
```



```

    radix
    tag_name
class XMLElementInt.Setter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Set text on element after converting to int then to str
    libvirtxml
    operation
    parent_xpath
    property_name
    radix
    tag_name
XMLElementInt.required_dargs = ('parent_xpath', 'tag_name', 'radix')
class virttest.libvirt_xml.accessors.XMLElementList(property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    marshal_from=None, mar-
                                                    shal_to=None)
    Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
    Class of accessor classes operating on a list of child elements
    Other generators here have a hard-time dealing with XML that has multiple child-elements with the same tag.
    This class allows treating these structures as lists of arbitrary user-defined objects. User-defined marshal func-
    tions are called to perform the conversion to/from the format described in __init__.
class Delter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Remove ALL child elements for which marshal_to does NOT return None
    libvirtxml
    marshal_to
    operation
    parent_xpath
    property_name
class XMLElementList.Getter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Retrieve list of values as returned by the marshal_to callable
    libvirtxml
    marshal_to
    operation
    parent_xpath
    property_name
class XMLElementList.Setter(operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Set child elements as returned by the marshal_to callable

```

```
    libvirtxml
    marshal_from
    operation
    parent_xpath
    property_name

XMLElementList.required_dargs = ('parent_xpath', 'tag_name', 'marshal_from', 'marshal_to')
class virttest.libvirt_xml.accessors.XMLElementNest (property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None, subclass=None,
                                                    subclass_dargs=None)
Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
Class of accessor classes operating on a LibvirtXMLBase subclass
class Delter (operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Remove element and ignore if it doesn't exist (same as False)
    libvirtxml
    operation
    parent_xpath
    property_name
    tag_name
class XMLElementNest.Getter (operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Retrieve instance of subclass with it's xml set to rerooted xpath/tag
    libvirtxml
    operation
    parent_xpath
    property_name
    subclass
    subclass_dargs
    tag_name
class XMLElementNest.Setter (operation, property_name, libvirtxml, **dargs)
    Bases: virttest.libvirt_xml.accessors.AccessorBase
    Set attributes to value on element
    libvirtxml
    operation
    parent_xpath
    property_name
    subclass
    tag_name
```

```
XMLElementNest.required_dargs = ('parent_xpath', 'tag_name', 'subclass', 'subclass_dargs')
class virttest.libvirt_xml.accessors.XMLElementText (property_name, libvirtxml, forbid-
                                                    den=None, parent_xpath=None,
                                                    tag_name=None)
Bases: virttest.libvirt_xml.accessors.AccessorGeneratorBase
Class of accessor classes operating on element.text

class Delter (operation, property_name, libvirtxml, **dargs)
Bases: virttest.libvirt_xml.accessors.AccessorBase
Remove element and ignore if it doesn't exist (same as False)

libvirtxml
operation
parent_xpath
property_name
tag_name

class XMLElementText.Getter (operation, property_name, libvirtxml, **dargs)
Bases: virttest.libvirt_xml.accessors.AccessorBase
Retrieve text on element

libvirtxml
operation
parent_xpath
property_name
tag_name

class XMLElementText.Setter (operation, property_name, libvirtxml, **dargs)
Bases: virttest.libvirt_xml.accessors.AccessorBase
Set text to value on element

libvirtxml
operation
parent_xpath
property_name
tag_name

XMLElementText.required_dargs = ('parent_xpath', 'tag_name')
virttest.libvirt_xml.accessors.add_to_slots (*args)
Return list of AccessorBase.__all_slots__ + args

virttest.libvirt_xml.accessors.type_check (name, thing, expected)
Check that thing is expected subclass or instance, raise ValueError if not
```

virttest.libvirt_xml.base module

```
class virttest.libvirt_xml.base.LibvirtXMLBase (virsh_instance=<module
                                                'virttest.virsh'
                                                from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.propcan.PropCanBase*

Base class for common attributes/methods applying to all sub-classes

Properties:

xml: virtual XMLTreeFile instance

get: xml filename string

set: create new XMLTreeFile instance from string or filename

del: deletes property, closes & unlinks any temp. files

xmldatafile: XMLTreeFile instance

virsh: virsh module or Virsh class instance

set: validates and sets value

get: returns value

del: removes value

validates: virtual boolean, read-only, True/False from virt-xml-validate

copy()

Returns a copy of instance not sharing any references or modifications

del_validates()

Raises LibvirtXMLError

del_xmldatafile()

Remove all backing XML

get_section_string(xpath)

Returns the content of section in xml.

get_validates()

Accessor method for 'validates' property returns virt-xml-validate T/F

get_xml()

Accessor method for 'xml' property returns xmlTreeFile backup filename

get_xmldatafile()

Return the xmldatafile object backing this instance

restore()

Restore current xml content to original source content

set_validates(value)

Raises LibvirtXMLError

set_virsh(value)

Accessor method for virsh property, make sure it's right type

set_xml(value)

Accessor method for 'xml' property to load using xml_utils.XMLTreeFile

set_xmldatafile(value)

Point instance directly at an already initialized XMLTreeFile instance

validates

virsh

static virt_xml_validate (*filename, schema_name=None*)

Return CmdResult from running virt-xml-validate on backing XML

xml

xmllreefile

`virttest.libvirt_xml.base.load_xml_module` (*path, name, type_list*)

Returns named xml element's handler class

Parameters

- **path** – the xml module path
- **name** – the xml module name
- **type_list** – the supported type list of xml module names

Returns the named xml element's handler class

virttest.libvirt_xml.capability_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatcaps.html>

```
class virttest.libvirt_xml.capability_xml.CapabilityXML (virsh_instance=<module  
                                                         'virttest.virsh'           from  
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Handler of libvirt capabilities and nonspecific item operations.

Properties:

uuid: string of host uuid

guest_capabilities: dict, read-only

get: dict map from os type names to dict map from arch names

add_feature (*value*)

Add a feature Element to xml

Parameters **value** – The added feature name

arch

cells_topology

check_feature_name (*name*)

Check feature name valid or not.

Parameters **name** – The checked feature name

Returns True if check pass

cpu_count

cpu_topology

feature_list

get_cpu_count ()

Accessor method for cpu_count property (in __slots__)

get_feature (*num*)

Get a feature element from feature list by number

Returns Feature element

get_feature_list ()

Accessor method for feature_list property (in __slots__)

get_feature_name (*num*)

Get assigned feature name

Parameters *num* – Assigned feature number

Returns Assigned feature name

get_guest_capabilities ()

Accessor method for guest_capabilities property (in __slots__). Return a guest capabilities dict in following schema: {<os_type>: {<arch name>: {'wordsize': '', 'emulator': '', 'machine': [<machine name>, ...], 'domaini_<type>': {'emulator': ''}}}}

get_power_management_list ()

Accessor method for power_management_list property (in __slots__)

guest_capabilities

model

power_management_list

remove_feature (*num*)

Remove a assigned feature from xml

Parameters *num* – Assigned feature number

set_feature (*num*, *value*)

Set a assigned feature value to xml

Parameters

- *num* – Assigned feature number
- *value* – The feature name modified to

uuid

vendor

```
class virttest.libvirt_xml.capability_xml.CellXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Handler of cell element in libvirt capabilities.

Properties:

cell_id: string of node cell number id

memory: int, memory size

mem_unit: string of memory unit

pages: list of pages dict

sibling: list of sibling dict

cpus_num: string of cpus number

cpu: list of cpu dict

cell_id

cpu

cpus_num

static marshal_from_cpu (*item, index, libvirtxml*)
Convert a dict to cpu tag and attributes.

static marshal_from_pages (*item, index, libvirtxml*)
Convert a dict to pages tag and attributes.

static marshal_from_sibling (*item, index, libvirtxml*)
Convert a dict to sibling tag and attributes.

static marshal_to_cpu (*tag, attr_dict, index, libvirtxml*)
Convert a cpu tag and attributes to a dict.

static marshal_to_pages (*tag, attr_dict, index, libvirtxml, text*)
Convert a pages tag and attributes to a dict.

static marshal_to_sibling (*tag, attr_dict, index, libvirtxml*)
Convert a sibling tag and attributes to a dict.

mem_unit

memory

pages

sibling

```
class virttest.libvirt_xml.capability_xml.TopologyXML (virsh_instance=<module
    'virttest.virsh'
    from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Handler of cells topology element in libvirt capabilities.

Properties:

num: string of node cell numbers

cell: list of cpu dict

cell

get_cell ()

Return CellXML instances list

num

virttest.libvirt_xml.network_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatnetwork.html>

```
class virttest.libvirt_xml.network_xml.DNSXML (virsh_instance=<module 'virttest.virsh' from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

IP address block, optionally containing DHCP range information

Properties:

txt: Dict. keys: name, value

forwarder: List

srv: Dict. keys: service, protocol, domain, target, port, priority, weight

hosts: List of host name

class HostXML (*virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Hostname element of dns

host_ip

hostnames

static marshal_from_hostname (*item, index, libvirtxml*)

Convert a HostnameXML instance into a tag + attributes

static marshal_to_hostname (*tag, attr, index, libvirtxml, text*)

Convert a tag + attributes into a HostnameXML instance

class DNSXML.HostnameXML (*virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Hostname element of dns

hostname

DNSXML.dns_forward

DNSXML.forwarders

DNSXML.host

static DNSXML.marshal_from_forwarder (*item, index, libvirtxml*)

Convert a dictionary into a tag + attributes

static DNSXML.marshal_to_forwarder (*tag, attr_dict, index, libvirtxml*)

Convert a tag + attributes into a dictionary

DNSXML.new_host (***dargs*)

Return a new disk IOTune instance and set properties from dargs

DNSXML.srv

DNSXML.txt

class *virttest.libvirt_xml.network_xml.IPXML* (*address='192.168.122.1', netmask='255.255.255.0', virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

IP address block, optionally containing DHCP range information

Properties: *dhcp_ranges:* Dict. keys: start, end *host_attr:* host mac, name and ip information *address:* string

IP address *netmask:* string IP's netmask

address

dhcp_bootp

dhcp_ranges**family****hosts****static marshal_from_host** (*item, index, libvirtxml*)

Convert a dictionary into a tag + attributes

static marshal_to_host (*tag, attr_dict, index, libvirtxml*)

Convert a tag + attributes into a dictionary

netmask**prefix****tftp_root**

```
class virttest.libvirt_xml.network_xml.NetworkXML (network_name='default',
                                                    virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.network_xml.NetworkXMLBase`

Manipulators of a Virtual Network through it's XML definition.

create ()

Adds non-persistent / transient network to libvirt with net-create

debug_xml ()

Dump contents of XML file for debugging

define ()

Define network from self.xml.

exists ()

Return True if network already exists.

```
static get_uuid_by_name (network_name, virsh_instance=<module 'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Return Network's uuid by Network's name.

Parameters network_name – Network's name**Returns** Network's uuid

```
static new_all_networks_dict (virsh_instance=<module 'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Return a dictionary of names to NetworkXML instances for all networks

Parameters virsh – virsh module or instance to use**Returns** Dictionary of network name to NetworkXML instance

```
static new_from_net_dumpxml (network_name, virsh_instance=<module 'virttest.virsh'
                                                                from
                                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                                test/checkouts/latest/virttest/virsh.pyc'>, extra='')
```

Return new NetworkXML instance from virsh net-dumpxml command

Parameters

- **network_name** – Name of network to net-dumpxml

- **virsh_instance** – virsh module or instance to use

Returns New initialized NetworkXML instance

orbital_nuclear_strike()

It's the only way to really be sure. Remove all libvirt state

start()

Start network with self.virsh.

state_dict()

Return a dict containing states of active/autostart/persistent

Returns A dict contains active/autostart/persistent as keys and boolean as values or None if network doesn't exist.

sync (*state=None*)

Make the change of “self” take effect on network. Recover network to designated state if option state is set.

Parameters **state** – a boolean dict contains active/persistent/autostart as keys

undefine()

Undefine network witch name is self.name.

```
class virttest.libvirt_xml.network_xml.NetworkXMLBase (virsh_instance=<module
                                                         'virttest.virsh'
                                                         from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Accessor methods for NetworkXML class.

Properties:

name: string, operates on XML name tag

uuid: string, operates on uuid tag

mac: string, operates on address attribute of mac tag

ip: string operate on ip/dhcp ranges as IPXML instances

forward: dict, operates on forward tag

forward_interface: list, operates on forward/interface tag

nat_port: dict, operates on nat tag

bridge: dict, operates on bridge attributes

routes: list, operates on route tag.

virtualport_type: string, operates on ‘type’ attribute of virtualport tag.

bandwidth_inbound: dict, operates on inbound under bandwidth.

bandwidth_outbound: dict, operates on outbound under bandwidth.

portgroup: PortgroupXML instance to access portgroup tag.

domain_name: string, operates on name attribute of domain tag

dns: DNSXML instance to access dns tag.

defined: virtual boolean, callout to virsh methods

get: True if libvirt knows network name

set: True defines network, False undefines to libvirt
del: Undefines network to libvirt
active: virtual boolean, callout to virsh methods
get: True if network is active to libvirt
set: True activates network, False deactivates to libvirt
del: Deactivates network to libvirt
autostart: virtual boolean, callout to virsh methods
get: True if libvirt autostarts network with same name
set: True to set autostart, False to unset to libvirt
del: Unset autostart to libvirt
persistent: virtual boolean, callout to virsh methods
get: True if network was defined, False if only created.
set: Same as defined property
del: Same as defined property

active
autostart
bandwidth_inbound
bandwidth_outbound
bridge
defined
del_active()
 Accessor method for 'active' property, stops network
del_autostart()
 Accessor method for 'autostart' property, unsets autostart
del_defined()
 Accessor method for 'define' property, undefines network
del_ip()
del_persistent()
 Accessor method for 'define' property, undefines network
del_portgroup()
dns
domain_name
forward
forward_interface
get_active()
 Accessor method for 'active' property (True/False)
get_autostart()
 Accessor method for 'autostart' property, True if set

get_defined()
Accessor for 'define' property - does this name exist in network list

get_ip()

get_persistent()
Accessor method for 'persistent' property

get_portgroup()

ip

mac

static marshal_from_forward_iface (*item, index, libvirtxml*)
Convert a dictionary into a tag + attributes

static marshal_from_route (*item, index, libvirtxml*)
Convert a dictionary into a tag + attributes

static marshal_to_forward_iface (*tag, attr_dict, index, libvirtxml*)
Convert a tag + attributes into a dictionary

static marshal_to_route (*tag, attr_dict, index, libvirtxml*)
Convert a tag + attributes into a dictionary

name

nat_port

new_dns (***dargs*)
Return a new dns instance and set properties from dargs

persistent

portgroup

routes

set_active (*value*)
Accessor method for 'active' property, sets network active

set_autostart (*value*)
Accessor method for 'autostart' property, sets/unsets autostart

set_defined (*value*)
Accessor method for 'define' property, set True to define.

set_ip (*value*)

set_persistent (*value*)
Accessor method for 'define' property, set True to define.

set_portgroup (*value*)

uuid

virtualport_type

```
class virttest.libvirt_xml.network_xml.PortgroupXML (virsh_instance=<module
    'virttest.virsh'
    from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for PortgroupXML class in NetworkXML.

Properties:

name: string, operates on 'name' attribute of portgroup tag

default: string of yes or no, operates on 'default' attribute of portgroup tag

virtualport_type: string, operates on 'type' attribute of virtualport tag in portgroup.

bandwidth_inbound: dict, operates on inbound tag in bandwidth which is child of portgroup.

bandwidth_outbound: dict, operates on outbound tag in bandwidth which is child of portgroup.

vlan_tag: dict, operates on vlan tag of portgroup

bandwidth_inbound**bandwidth_outbound****default****name****virtualport_type****vlan_tag**

class `virttest.libvirt_xml.network_xml.RangeList` (*iterable=None*)

Bases: `list`

A list of start & end address tuples

append_to_element (*element*)

Adds range described by instance to ElementTree.element

virttest.libvirt_xml.nodedev_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatnode.html>

class `virttest.libvirt_xml.nodedev_xml.CAPXML` (*virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

The base class for capability.

static `get_key2filename_dict` ()

Return a dict which contain the key and the name of info file.

get_key2value_dict ()

Return a dict which contain the key and the value in capability xml.

get_sysfs_sub_path ()

return the sub path store the info of capability.

class `virttest.libvirt_xml.nodedev_xml.NetXML` (*virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.nodedev_xml.CAPXML`

class for capability whose type is net.

address**interface**

```
class virttest.libvirt_xml.nodedev_xml.NodedevXML (virsh_instance=<module
                                                    'virttest.virsh'           from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

Bases: virttest.libvirt_xml.nodedev_xml.NodedevXMLBase

class for Node device XML.

get_key2syspath_dict ()
    Get the dict which contains key and path. key: keys in nodedev xml need to check. syspath: the abs path
    for the file stores info for the key.

get_key2value_dict ()
    Get the dict which contain key and value in xml. key: keys in nodedev xml need to check. value: value in
    xml for the key.

static new_from_dumpxml (dev_name, virsh_instance=<module 'virttest.virsh' from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
    Get a instance of NodedevXML by dumpxml dev_name.

class virttest.libvirt_xml.nodedev_xml.NodedevXMLBase (virsh_instance=<module
                                                         'virttest.virsh'           from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                         test/checkouts/latest/virttest/virsh.pyc'>)

Bases: virttest.libvirt_xml.base.LibvirtXMLBase

Accessor methods for NodedevXML class.

cap
cap_type
del_cap ()
    Delete the capability from nodedev xml.

fabric_wwn
fc_type
get_cap ()
    Return the capability of nodedev_xml.

static get_cap_by_type (cap_type)
    Init a cap class for a specific type.

    Parameters cap_type – the type of capability.

    Returns instanse of the cap.

get_sysfs_path ()
    Get the abs path of the capability info.

get_sysfs_sub_path ()
    Get the sub sysfs path of the capability.

host
name
parent
set_cap (value)
    Set the capability by value.
```

sysfs_main_path

wwnn

wwpn

```
class virttest.libvirt_xml.nodedev_xml.PCIXML (virsh_instance=<module 'virttest.virsh' from
                                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nodedev_xml.CAPXML`

class for capability whose type is pci.

```
class Address (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Address of Virtual Function device.

bus

domain

function

slot

`PCIXML.bus`

`PCIXML.domain`

`PCIXML.function`

`PCIXML.get_address_dict()`

Return a dict contain the address.

static `PCIXML.get_key2filename_dict()`

return the dict key2filename. key: the keys in pcixml need to check. filename: the name of file stored info for this key.

`PCIXML.get_key2value_dict()`

return the dict key2value

key: the key in xml need to check. value: value in xml for this key.

`PCIXML.get_sysfs_sub_path()`

Return the sysfs_subdir in .

Example: pci_bus/0000:00/device/0000:00:00.0/

static `PCIXML.make_sysfs_sub_path(domain, bus, slot, function)`

Make sysfs_sub_path for pci by domain,bus,slot and function.

static `PCIXML.marshall_from_address(item, index, libvirtxml)`

Convert an Address instance into tag + attributes

static `PCIXML.marshall_to_address(tag, attr_dict, index, libvirtxml)`

Convert a tag + attributes into an Address instance

`PCIXML.numa_node`

`PCIXML.product_id`

`PCIXML.slot`

`PCIXML.vendor_id`

`PCIXML.virt_functions`

```
class virttest.libvirt_xml.nodedev_xml.StorageXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nodedev_xml.CAPXML`

class for capability whose type is storage.

block

bus

driver_type

```
class virttest.libvirt_xml.nodedev_xml.SystemXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nodedev_xml.CAPXML`

class for capability which type is system.

firm_release_date

firmversion

firmware_vendor

static get_key2filename_dict ()

Return a dict which contain the key and the name of info file for System node device.

get_key2value_dict ()

return the dict key2value

key: the key in xml need to check. value: value in xml for this key.

get_sysfs_sub_path ()

Return the sysfs_subdir.

hardware_serial

hardware_uuid

hardware_vendor

static make_sysfs_sub_path ()

return __sysfs_sub_path__ immediately.

product

virttest.libvirt_xml.nwfilter_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatnwfilter.html>

```
class virttest.libvirt_xml.nwfilter_xml.NwfilterRulesProtocol
```

Bases: `list`

List of protocol instances from classes handed out by librarian.get

append (value)

by_device_tag (tag)

extend (iterable)


```
class virttest.libvirt_xml.nwfilter_xml.NwfilterXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase`

Manipulators of a nwfilter through it's XML definition.

filter_chain

filter_name

filter_priority

filterrefs

get_all_protocols (*protocol=None*)

Put all type of protocol into a NwfilterRulesProtocol instance. Return all protocols class list if protocol as None, else return specific protocol type class list.

Parameters **protocol** – specific protocol type in rules

Returns NwfilterRulesProtocol instance list

get_all_rules ()

Return all rules dict with protocol attribute.

Returns all rules dict with key as rule index number

get_rules_dict (*filter_name*, *options=''*, *virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Return all rules dict with protocol attribute for given filter

Parameters

- **filter_name** – name or uuid of filter
- **options** – extra options

Returns all rules dictionary with index as key

static new_from_filter_dumpxml (*uuid*, *options=''*, *virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest/virsh.pyc'>*)

Return new NwfilterXML instance from virsh filter-dumpxml command

Parameters

- **uuid** – filter's uuid
- **virsh_instance** – virsh module or instance to use

Returns New initialized NwfilterXML instance

uuid

validates

virsh

xml

xmlltreefile

```
class virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase (virsh_instance=<module  
                                                         'virttest.virsh'           from  
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for NwfilterXML class.

Properties: filter_name: string, filter name filter_chain: string, filter name filter_priority: string, filter priority
 uuid: string, operates on uuid tag filterrefs: list, list of dictionaries describing filterref properties

add_rule (*value*)

Add new rule into filter

Parameters *value* – NwfilterXMLRules instance

del_rule (*rule_index=0*)

Delete rule with specific index

Parameters *rule_index* – rule's index number

filter_chain

filter_name

filter_priority

filterrefs

get_protocol_attr (*rule_index=0, protocol=None*)

Return protocol dict of specific rule index and protocol type

Parameters

- **rule_index** – rule's index number
- **protocol** – the specific protocol type in rules

Returns protocol attribute dict

get_rule (*rule_index=0, rule_protocol=None*)

Return NwfilterXMLRules instance for specific protocol and index

Parameters

- **rule_index** – rule's index number
- **rule_protocol** – the specific protocol type in rules

Returns New initialized NwfilterXMLRules instance

get_rule_index (*rule_protocol=None*)

Return rule index list for specific protocol

Parameters *rule_protocol* – the specific protocol type in rules

Returns rule index list

static marshal_from_filterref (*item, index, libvirtxml*)

Convert a dictionary into a tag + attributes

static marshal_to_filterref (*tag, attr_dict, index, libvirtxml*)

Convert a tag + attributes into a dictionary

set_rule (*value, rule_index=0*)

Delete rule with specific index and add new given value

Parameters

- **rule_index** – rule’s index number
- **value** – NwfilterXMLRules instance

uuid
validates
virsh
xml
xmlltreefile

```
class virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules (protocol=None,
                                                         virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Create new NwfilterXMLRules instance.

Properties: rule_action: string, rule action rule_direction: string, rule direction priority: string, rule priority
 statematch: string, rule statematch

backup_rule()
 Return backup rule instance

Returns the backup of rule instance

del_protocol()
 Delete protocol in rule xml

get_protocol (protocol=None)
 Return None if protocol is None, else return specific class instance

Parameters protocol – specific protocol type in rules

Returns specific protocol class instance from librarian.get

new_protocol (**dargs)
 Return a new rule protocol instance and set properties from dargs

rule_action
rule_direction
rule_priority
rule_statematch

virttest.libvirt_xml.pool_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatstorage.html#StoragePool>

```
class virttest.libvirt_xml.pool_xml.PoolXML (pool_type='dir', virsh_instance=<module
                                                         'virttest.virsh' from
                                                         '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.pool_xml.PoolXMLBase*

Manipulators of a libvirt Pool through it’s XML definition.

```
static backup_xml (name, virsh_instance=<module 'virttest.virsh' from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Backup the pool xml file.

```
debug_xml ()
```

Dump contents of XML file for debugging

```
static get_pool_details (name, virsh_instance=<module 'virttest.virsh' from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Return pool details by pool name.

Parameters **name** – pool name

Returns a dict which include a series of pool details

```
static get_type (name, virsh_instance=<module 'virttest.virsh' from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Return pool type by pool name

Parameters **name** – pool name

Returns pool type

```
static new_from_dumpxml (name, virsh_instance=<module 'virttest.virsh' from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Return new PoolXML instance from virsh pool-dumpxml command

Parameters

- **name** – Name of pool to pool-dumpxml
- **virsh_instance** – Virsh module or instance to use

Returns new initialized PoolXML instance

```
pool_define ()
```

Define pool with virsh from this instance

```
static pool_rename (name, new_name, uuid=None, virsh_instance=<module 'virttest.virsh'
                    from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Rename a pool from pool XML. :param name: Original pool name. :param new_name: new name of pool. :param uuid: new pool uuid, if None libvirt will generate automatically. :return: True/False or raise LibvirtXMLError

```
pool_undefine ()
```

Undefine pool with libvirt retaining XML in instance

```
class virttest.libvirt_xml.pool_xml.PoolXMLBase (virsh_instance=<module
                    'virttest.virsh' from
                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for PoolXML class.

Properties:

pool_type: string, pool type

name: string, pool name

uuid: string, pool uuid
capacity: integer, pool total capacity
allocation: integer, pool allocated capacity
available: integer, pool available capacity
source: PoolSourceXML instance
target: string, target path of pool

allocation

available

capacity

del_source()

get_source()

group

mode

name

owner

pool_type

set_source(value)

source

target_path

uuid

```
class virttest.libvirt_xml.pool_xml.SourceXML(virsh_instance=<module 'virttest.virsh' from
                                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Source block in pool xml, optionally containing different elements and attributes which dependent on pool type.

adp_name

adp_parent

adp_type

adp_wwnn

adp_wwpn

auth_type

auth_username

device_path

dir_path

format_type

host_name

hosts

static marshal_from_host (*item, index, libvirtxml*)

Convert a dictionary into a tag + attributes

static marshal_to_host (*tag, attr_dict, index, libvirtxml*)

Convert a tag + attributes into a dictionary

secret_usage

secret_uuid

vg_name

virttest.libvirt_xml.secret_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatsecret.html>

```
class virttest.libvirt_xml.secret_xml.SecretXML (ephemeral='yes',          private='no',
                                                  virsh_instance=<module
                                                  'virttest.virsh'          from
                                                  '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                  test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.secret_xml.SecretXMLBase`

Manipulators of a secret through it's XML definition.

```
static get_secret_details_by_uuid (uuid, virsh_instance=<module 'virttest.virsh' from
                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Return secret XML by secret's uuid

Parameters **uuid** – secret's uuid

Returns secret XML dictionary

```
static new_from_secret_dumpxml (uuid, virsh_instance=<module 'virttest.virsh' from
                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Return new SecretXML instance from virsh secret-dumpxml command

Parameters

- **uuid** – secret's uuid
- **virsh_instance** – virsh module or instance to use

Returns New initialized SecretXML instance

```
class virttest.libvirt_xml.secret_xml.SecretXMLBase (virsh_instance=<module
                                                  'virttest.virsh'          from
                                                  '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                  test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for SecretXML class.

Properties:

secret_ephemeral: yes or no, operates on XML secret tag

secret_private: yes or no, operates on XML secret tag

description: string, operates on description tag

uuid: string, operates on uuid tag

auth_type: string, secret authentication type, operates on auth tag

auth_username: string, secret authentication username, operates on auth tag

usage: string, operates on usage tag

target: string, sub-tag of the usage tag, operates on target tag

volume: the volume file path, sub-tag of the usage tag, operates on volume tag

auth_type

auth_username

description

secret_ephemeral

secret_private

target

usage

usage_name

uuid

volume

virttest.libvirt_xml.snapshot_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatnapshot.html>

```
class virttest.libvirt_xml.snapshot_xml.SnapshotXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase*

Manipulators of a snapshot through it's XML definition.

```
class SnapDiskXML (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.devices.disk.Disk*

Manipulators disk xml in snapshot xml definition. Most properties are inherit from parent class Disk.

Properties:

disk_name: string, operates on disk name under disk tag

address

auth

blockio

boot

device

disk_name

driver

encryption

geometry

iotune

mirror
product
rawio
readonly
ready
serial
sgio
share
snapshot
source
target
transient
vendor
wwn

`SnapshotXML.del_disks()`
Remove all disks

static `SnapshotXML.new_from_snapshot_dumpxml(name, snap_name, virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>)`
Return new SnapshotXML instance from virsh snapshot-dumpxml command

Parameters

- **name** – vm’s name
- **snap_name** – snapshot name
- **uuid** – snapshot’s uuid
- **virsh_instance** – virsh module or instance to use

Returns New initialized SnapshotXML instance

`SnapshotXML.set_disks(value_list)`
Define disks based on contents of SnapDiskXML instance list

Parameters **value_list** – SnapDiskXML instance list

class `virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase` (`virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virttest/checkouts/latest/virttest/virsh.pyc'>`)

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for SnapshotXML class.

Properties:

snap_name: string, operates on snapshot name tag
description: string, operates on snapshot description tag

mem_snap_type: string, operates snapshot type under memory tag, 'internal', 'external' or 'no'

mem_file: string, operates snapshot file path under memory tag

creation_time: string, operates on creationTime tag

state: string, operates snapshot state tag

parent_name: string, parent snapshot name tag under parent tag

creation_time

description

mem_file

mem_snap_type

parent_name

snap_name

state

virttest.libvirt_xml.sysinfo_xml module Module simplifying manipulation of sysinfo XML

```
class virttest.libvirt_xml.sysinfo_xml.SysinfoXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

Handler of libvirt sysinfo xml.

get_all_processors ()

Get all processors dict with entry name as key.

Returns all processors dict with entry name as key

virttest.libvirt_xml.vm_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatdomain.html>

```
class virttest.libvirt_xml.vm_xml.VMCPUTuneXML (virsh_instance=<module
                                                    'virttest.virsh'
                                                    from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: *virttest.libvirt_xml.base.LibvirtXMLBase*

CPU tuning tag XML class

Elements: vcpupins: list of dict - vcpu, cpuset emulatorpin: attribute - cpuset shares: int period: int quota: int
emulator_period: int emulator_quota: int

emulator_period

emulator_quota

emulatorpin

static marshal_from_vcpupins (item, index, libvirtxml)

Convert a dict to vcpupin tag and attributes.

static marshal_to_vcpupins (tag, attr_dict, index, libvirtxml)

Convert a vcpupin tag and attributes to a dict.

period

quota

shares

vcpupins

```
class virttest.libvirt_xml.vm_xml.VMCPUXML (virsh_instance=<module 'virttest.virsh' from
                                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Higher-level manipulations related to VM's XML(CPU)

add_feature (*name*, *policy*='')

Add a feature element to xml

Parameters

- **name** – New feature name
- **policy** – New feature policy

static check_feature_name (*value*)

Check feature name valid or not.

Parameters **value** – Feature name

Returns True if check pass

fallback

feature_list

get_feature (*num*)

Get a feature element from feature list by number

Returns Feature element

get_feature_list ()

Accessor method for feature_list property (in __slots__)

get_feature_name (*num*)

Get feature name

Parameters **num** – Number in feature list

Returns Feature name

get_feature_policy (*num*)

Get feature policy

Parameters **num** – Number in feature list

Returns Feature policy

static marshal_from_cell (*item*, *index*, *libvirtxml*)

Convert a dict to cell tag and attributes.

static marshal_to_cell (*tag*, *attr_dict*, *index*, *libvirtxml*)

Convert a cell tag and attributes to a dict.

match

mode

model

numa_cell

remove_feature (*num*)

Remove a feature from xml

Parameters **num** – Number in feature list

set_feature (*num*, *name*='', *policy*='')

Set feature name (and policy) to xml

Parameters

- **num** – Number in feature list
- **name** – New feature name
- **policy** – New feature policy

topology

vendor

```
class virttest.libvirt_xml.vm_xml.VMClockXML (virsh_instance=<module 'virttest.virsh' from
                                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>,
                                             offset='utc')
```

Bases: `virttest.libvirt_xml.vm_xml.VMXML`

Higher-level manipulations related to VM's XML(Clock)

```
class TimerXML (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>, timer_name='tsc')
```

Bases: `virttest.libvirt_xml.vm_xml.VMXML`

Timer element of clock

catchup_limit

catchup_slew

catchup_threshold

frequency

mode

name

present

tickpolicy

track

update (*attr_dict*)

`VMClockXML.adjustment`

```
VMClockXML.from_dumpxml (vm_name, virsh_instance=<module 'virttest.virsh' from
                                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Helper to load xml from domain.

static `VMClockXML.marshall_from_timer` (*item*, *index*, *libvirtxml*)

Convert a TimerXML instance into tag + attributes

static `VMClockXML.marshall_to_timer` (*tag*, *attr_dict*, *index*, *libvirtxml*)

Convert a tag + attributes to a TimerXML instance

VMClockXML.**offset**

VMClockXML.**timers**

VMClockXML.**timezone**

```
class virttest.libvirt_xml.vm_xml.VMFeaturesXML (virsh_instance=<module
                                         'virttest.virsh'                      from
                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Class to access <features> tag of domain XML.

Elements: feature_list list of top level element hyperv_relaxed: attribute - state hyperv_vapic: attribute - state
hyperv_spinlocks: attributes - state, retries kvm_hidden: attribute - state pvspinlock: attribute - state

add_feature (name, attr_name='', attr_value='')

Add a feature element to xml

Params name Feature name

feature_list

get_feature_list ()

Return all features(top level elements) in xml

has_feature (name)

Return true if the given feature exist in xml

hyperv_relaxed_state

hyperv_spinlocks_retries

hyperv_spinlocks_state

hyperv_vapic_state

kvm_hidden_state

pvspinlock_state

remove_feature (name)

Remove a feature element from xml

Params name Feature name

```
class virttest.libvirt_xml.vm_xml.VMHugepagesXML (virsh_instance=<module
                                         'virttest.virsh'                      from
                                         '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.vm_xml.VMXML`

hugepages element

```
class PageXML (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                         test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.vm_xml.VMXML`

Page element of hugepages

nodeset

size

unit

```

    update (attr_dict)

static VMHugepagesXML.marshal_from_page (item, index, libvirtxml)
    Convert a PageXML instance into tag + attributes

static VMHugepagesXML.marshal_to_page (tag, attr_dict, index, libvirtxml)
    Convert a tag + attributes to a PageXML instance

VMHugepagesXML.pages

class virttest.libvirt_xml.vm_xml.VMMemBackingXML (virsh_instance=<module
                                                    'virttest.virsh'      from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

Bases: virttest.libvirt_xml.vm_xml.VMXML
memoryBacking tag XML class

Elements: hugepages nosharepages locked

hugepages

locked

nosharepages

class virttest.libvirt_xml.vm_xml.VMMemTuneXML (virsh_instance=<module
                                                    'virttest.virsh'      from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

Bases: virttest.libvirt_xml.base.LibvirtXMLBase
Memory Tuning tag XML class

Element: hard_limit: int hard_limit_unit: attribute soft_limit: int soft_limit_unit: attribute swap_hard_limit:
        int swap_limit_unit: attribute min_guarantee: int min_guarantee_unit: attribute

hard_limit

hard_limit_unit

min_guarantee

min_guarantee_unit

soft_limit

soft_limit_unit

swap_hard_limit

swap_limit_unit

class virttest.libvirt_xml.vm_xml.VMOSXML (virsh_instance=<module      'virttest.virsh'      from
                                                    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                    test/checkouts/latest/virttest/virsh.pyc'>)

Bases: virttest.libvirt_xml.base.LibvirtXMLBase

Class to access <os> tag of domain XML.

Elements: type: text attributes - arch, machine loader: path boots: list attributes - dev bootmenu: attributes
        - enable_smbios: attributes - mode bios: attributes - useserial, rebootTimeout init: text bootloader: text
        bootloader_args: text kernel: text initrd: text cmdline: text dtb: text

TODO: initargs: list

arch

```

`bios_reboot_timeout`

`bios_useserial`

`bootloader`

`bootloader_args`

`bootmenu_enable`

`boots`

`cmdline`

`dtb`

`init`

`initargs`

`initrd`

`kernel`

`loader`

`machine`

static `marshal_from_boots` (*item, index, libvirtxml*)

Convert a string to boot tag and attributes.

static `marshal_to_boots` (*tag, attr_dict, index, libvirtxml*)

Convert a boot tag and attributes to a string.

`smbios_mode`

`type`

class `virttest.libvirt_xml.vm_xml.VMPMXML` (*virsh_instance=<module 'virttest.virsh' from
'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

VM power management tag XML class

Elements: suspend-to-disk: attribute - enabled suspend-to-mem: attribute - enabled

disk_enabled

mem_enabled

class `virttest.libvirt_xml.vm_xml.VMXML` (*hypervisor_type='kvm',
virsh_instance=<module 'virttest.virsh' from
'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'>*)

Bases: `virttest.libvirt_xml.vm_xml.VMXMLBase`

Higher-level manipulations related to VM's XML or guest/host state

add_device (*value*)

Add a device into VMXML.

Parameters *value* – instance of device in `libvirt_xml/devices/`

add_hostdev (*source_address, mode='subsystem', hostdev_type='pci', managed='yes',
boot_order=None*)

Add a hostdev device to guest.

Parameters *source_address* – A dict include slot, function, bus, domain

static add_security_info (*vmxml*, *passwd*)

Add passwd for graphic

Parameters

- **vmxml** – instance of VMXML
- **passwd** – Password you want to set

static check_cpu_mode (*mode*)

Check input cpu mode invalid or not.

Parameters **mode** – the mode of cpu: 'host-model'...

static check_disk_exist (*vm_name*, *disk_src*, *virsh_instance*=<module *'virttest.virsh'*
from *'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'*>)

Check if given disk exist in VM.

Parameters

- **vm_name** – Domain name.
- **disk_src** – Domain disk source path or target dev.

Returns True/False

static check_disk_type (*vm_name*, *disk_src*, *disk_type*, *virsh_instance*=<module *'virttest.virsh'*
from *'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'*>)

Check if disk type is correct in VM

Parameters

- **vm_name** – Domain name.
- **disk_src** – Domain disk source path
- **disk_type** – Domain disk type

Returns True/False

define ()

Define VM with virsh from this instance

del_device (*value*, *by_tag*=False)

Remove a device from VMXML

Parameters

- **value** – instance of device in libvirt_xml/devices/ or device tag
- **by_tag** – Boolean value. True for delete device by tag name

static del_memoryBacking_tag (*vm_name*, *virsh_instance*=<module *'virttest.virsh'* from
*'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'*>)

Remove the memoryBacking tag from a domain

get_agent_channels ()

Get all qemu guest agent channels

static get_blkdevio_params (*vm_name*, *options*='', *virsh_instance*=<module *'virttest.virsh'*
from *'/home/docs/checkouts/readthedocs.org/user_builds/virt-
test/checkouts/latest/virttest/virsh.pyc'*>)

Return VM's block I/O tuning setting from XML definition

```
static get_blkio_params (vm_name, options='', virsh_instance=<module 'virttest.virsh'
                        from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Return VM's block I/O setting from XML definition

```
static get_device_class (type_name)
```

Return class that handles type_name devices, or raise exception.

```
static get_disk_address (vm_name, disk_target, virsh_instance=<module 'virttest.virsh'
                        from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Get disk address in VM

Parameters

- **vm_name** – Domain name.
- **disk_target** – Domain disk target

Returns disk address

```
get_disk_all ()
```

Return VM's disk from XML definition, None if not set

```
static get_disk_attr (vm_name, target, tag, attr, virsh_instance=<module 'virttest.virsh'
                        from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Get value of disk tag attribute for a given target dev.

```
static get_disk_blk (vm_name, virsh_instance=<module 'virttest.virsh' from
                    'home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Get block device of a defined VM's disks.

Parameters **vm_name** – Name of defined vm.

```
static get_disk_count (vm_name, virsh_instance=<module 'virttest.virsh' from
                    'home/docs/checkouts/readthedocs.org/user_builds/virt-
                    test/checkouts/latest/virttest/virsh.pyc'>)
```

Get count of VM's disks.

Parameters **vm_name** – Name of defined vm.

```
static get_disk_serial (vm_name, disk_target, virsh_instance=<module 'virttest.virsh'
                        from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Get disk serial in VM

Parameters

- **vm_name** – Domain name.
- **disk_target** – Domain disk target

Returns disk serial

```
static get_disk_source (vm_name, option='', virsh_instance=<module 'virttest.virsh'
                        from 'home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Get block device of a defined VM's disks.

Parameters

- **vm_name** – Name of defined vm.
- **option** – extra option.


```
static get_first_mac_by_name (vm_name, virsh_instance=<module 'virttest.virsh' from
                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Convenience method for getting first mac of a defined VM

Param `vm_name`: Name of defined vm to get mac

```
get_graphics_devices (type_name='')
```

Get all graphics devices or desired type graphics devices

Parameters `type_name` – graphic type, vnc or spice

```
get_iface_all ()
```

Get a dict with interface's mac and node.

```
static get_iface_by_mac (vm_name, mac, virsh_instance=<module 'virttest.virsh'
                      from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                      test/checkouts/latest/virttest/virsh.pyc'>)
```

Get the interface if mac is matched.

Parameters

- **vm_name** – Name of defined vm.
- **mac** – a mac address.

Returns return a dict include main interface's features

```
static get_iface_dev (vm_name, virsh_instance=<module 'virttest.virsh' from
                  '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                  test/checkouts/latest/virttest/virsh.pyc'>)
```

Return VM's interface device from XML definition, None if not set

```
static get_iftune_params (vm_name, options='', virsh_instance=<module 'virttest.virsh'
                      from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                      test/checkouts/latest/virttest/virsh.pyc'>)
```

Return VM's interface tuning setting from XML definition

```
get_net_all ()
```

Return VM's net from XML definition, None if not set

```
static get_net_dev (vm_name)
```

Get net device of a defined VM's nets.

Parameters `vm_name` – Name of defined vm.

```
static get_numa_memnode_params (vm_name, virsh_instance=<module 'virttest.virsh' from
                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Return VM's numa memnode setting from XML definition

```
static get_numa_memory_params (vm_name, virsh_instance=<module 'virttest.virsh' from
                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                              test/checkouts/latest/virttest/virsh.pyc'>)
```

Return VM's numa memory setting from XML definition

```
get_primary_serial ()
```

Get a dict with primary serial features.

```
static new_from_dumpxml (vm_name, options='', virsh_instance=<module 'virttest.virsh'
                      from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                      test/checkouts/latest/virttest/virsh.pyc'>)
```

Return new VMXML instance from virsh dumpxml command

Parameters

- **vm_name** – Name of VM to dumpxml
- **virsh_instance** – virsh module or instance to use

Returns New initialized VMXML instance

```
static new_from_inactive_dumpxml (vm_name, options='', virsh_instance=<module  
                                'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Return new VMXML instance of inactive domain from virsh dumpxml command

Parameters

- **vm_name** – Name of VM to dumpxml
- **options** – virsh dumpxml command's options
- **virsh_instance** – virsh module or instance to use

Returns New initialized VMXML instance

remove_agent_channels ()

Delete all channels for guest agent

remove_all_boots ()

Remove all OS boots

remove_all_device_by_type (device_type)

Remove all devices of a given type.

Parameters type – Type name for devices should be removed.

remove_all_graphics ()

Remove all graphics devices.

set_agent_channel (src_path=None, tgt_name='org.qemu.guest_agent.0', ignore_exist=False)

Add a channel for guest agent if non exists.

Parameters

- **src_path** – Source path of the channel
- **tgt_name** – Target name of the channel
- **ignore_exist** – Whether add a channel even if another already exists.

static set_cpu_mode (vm_name, mode='host-model')

Set cpu's mode of VM.

Parameters

- **vm_name** – Name of defined vm to set cpu mode.
- **mode** – the mode of cpu: 'host-model' ...

```
static set_memoryBacking_tag (vm_name, hpgs=True, nosp=False, locked=False,  
                              virsh_instance=<module 'virttest.virsh' from  
                              '/home/docs/checkouts/readthedocs.org/user_builds/virt-  
                              test/checkouts/latest/virttest/virsh.pyc'>)
```

let the guest using hugepages.

static set_multiqueues (vm_name, queues, index=0)

Set multiqueues for interface.

Parameters

- **queues** – the count of queues for interface

- **index** – the index of interface

```
static set_pm_suspend (vm_name, mem='yes', disk='yes', virsh_instance=<module
                        'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Add/set pm suspend Support

Params vm_name Name of defined vm

Params mem Enable suspend to memory

Params disk Enable suspend to disk

```
static set_primary_serial (vm_name, dev_type, port, path=None,
                           virsh_instance=<module 'virttest.virsh'
                           from
                           '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                           test/checkouts/latest/virttest/virsh.pyc'>)
```

Set primary serial's features of vm_name.

Parameters

- **vm_name** – Name of defined vm to set primary serial.
- **dev_type** – the type of serial:pty, file...
- **port** – the port of serial
- **path** – the path of serial, it is not necessary for pty

```
static set_vm_vcpus (vm_name, value, current=None, virsh_instance=<module 'virttest.virsh'
                           from
                           '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                           test/checkouts/latest/virttest/virsh.pyc'>)
```

Convenience method for updating 'vcpu' and 'current' attribute property of a defined VM

Parameters

- **vm_name** – Name of defined vm to change vcpu elemnet data
- **value** – New data value, None to delete.
- **current** – New current value, None will not change current value

sync (options=None)

Rebuild VM with the config file.

undefine (options=None)

Undefine this VM with libvirt retaining XML in instance

```
static vm_rename (vm, new_name, uuid=None, virsh_instance=<module 'virttest.virsh'
                        from
                        '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                        test/checkouts/latest/virttest/virsh.pyc'>)
```

Rename a vm from its XML.

Parameters

- **vm** – VM class type instance
- **new_name** – new name of vm
- **uuid** – new_vm's uuid, if None libvirt will generate.

Returns a new VM instance

```
class virttest.libvirt_xml.vm_xml.VMXMLBase (virsh_instance=<module 'virttest.virsh' from
                                                '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                                test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for VMXML class properties (items in `__slots__`)

Properties:

hypervisor_type: string, hypervisor type name get: return domain's type attribute value set: change domain type attribute value del: raise `xcepts.LibvirtXMLError`

vm_name: string, name of the vm get: return text value of name tag set: set text value of name tag del: raise `xcepts.LibvirtXMLError`

uuid: string, uuid string for vm get: return text value of uuid tag set: set text value for (new) uuid tag (unvalidated) del: remove uuid tag

vcpu, max_mem, current_mem, iothreads: integers get: returns integer set: set integer del: removes tag

dumpcore: string, control guest OS memory dump get: return text value set: set 'on' or 'off' for guest OS memory dump del: removes tag

numa_memory: dictionary get: return dictionary of numatune/memory attributes set: set numatune/memory attributes from dictionary del: remove numatune/memory tag

numa_memnode: list dict of memnode attributes cellid, mode and nodeset get: return list of dictionary with numatune/memnode attributes set: set multiple numatune/memnode attributes from dictionary list del: remove numatune/memnode tag

on_poweroff: string, action to take when the guest requests a poweroff get: returns text value of on_poweroff tag set: set test of on_poweroff tag del: remove on_poweroff tag

on_reboot: string, action to take when the guest requests a reboot get: returns text value of on_reboot tag set: set test of on_reboot tag del: remove on_reboot tag

on_crash: string, action to take when the guest crashes get: returns text value of on_crash tag set: set test of on_crash tag del: remove on_crash tag

devices: VMXMLDevices (list-like) get: returns VMXMLDevices instance for all devices set: Define all devices from VMXMLDevices instance del: remove all devices

cputune: VMCPUTuneXML get: return VMCPUTuneXML instance for the domain. set: Define cputune tag from a VMCPUTuneXML instance. del: remove cputune tag

cpu: VMCPUXML get: return VMCPUXML instance for the domain. set: Define cpu tag from a VMCPUXML instance. del: remove cpu tag

current_vcpu: string, 'current' attribute of vcpu tag get: return a string for 'current' attribute of vcpu set: change 'current' attribute of vcpu del: remove 'current' attribute of vcpu

placement: string, 'placement' attribute of vcpu tag get: return a string for 'placement' attribute of vcpu set: change 'placement' attribute of vcpu del: remove 'placement' attribute of vcpu

cpuset: string, 'cpuset' attribute of vcpu tag get: return a string for 'cpuset' attribute of vcpu set: change 'cpuset' attribute of vcpu del: remove 'cpuset' attribute of vcpu

emulatorpin: string, cpuset value (see man virsh: cpulist) get: return text value of cputune/emulatorpin attributes set: set cputune/emulatorpin attributes from string del: remove cputune/emulatorpin tag

features: VMFeaturesXML get: return VMFeaturesXML instances for the domain. set: define features tag from a VMFeaturesXML instances. del: remove features tag

mem_backing: VMMemBackingXML get: return VMMemBackingXML instances for the domain. set: define memoryBacking tag from a VMMemBackingXML instances. del: remove memoryBacking tag

max_mem_unit: string, 'unit' attribute of memory get: return text value of memory unit attribute set: set memory unit attribute del: remove memory unit attribute

current_mem_unit: string, 'unit' attribute of memory get: return text value of current_memory unit attribute set: set current_memory unit attribute del: remove current_memory unit attribute

memtune: VMMemTuneXML get: return VMMemTuneXML instance for the domain. set: Define memtune tag from a VMCPUTuneXML instance. del: remove memtune tag

cpu

cpuset

cputune

current_mem

current_mem_unit

current_vcpu

del_controller (*controller_type=None*)
Delete controllers according controller type

Returns None if deleting all controllers

del_devices ()
Remove all devices

del_seclabel ()
Remove the seclabel tag from a domain

devices

dumpcore

features

get_devices (*device_type=None*)
Put all nodes of devices into a VMXMLDevices instance.

get_seclabel ()
Return seclabel + child attribute dict list or raise LibvirtXML error

Returns None if no seclabel in xml, list contains dict of seclabel's attributs and children.

hypervisor_type

iothreads

static marshal_from_memnode (*item, index, libvirtxml*)
Convert a dict to memnode tag and attributes.

static marshal_to_memnode (*tag, attr_dict, index, libvirtxml*)
Convert a memnode tag and attributes to a dict.

max_mem

max_mem_rt

max_mem_rt_slots

max_mem_rt_unit

max_mem_unit

mb

memtune

numa_memnode

numa_memory

on_crash

on_poweroff

on_reboot

os

placement

pm

seclabel

set_controller (*controller_list*)

Set controller of vm. Create new controllers use xmltreefile from given controller_list.

set_devices (*value*)

Define devices based on contents of VMXMLDevices instance

set_seclabel (*seclabel_dict_list*)

Set seclabel of vm. Delete all seclabels if seclabel exists, create new seclabels use dict values from given seclabel_dict_list in xmltreefile.

uuid

vcpu

vm_name

class `virttest.libvirt_xml.vm_xml.VMXMLDevices`

Bases: `list`

List of device instances from classes handed out by `librarian.get()`

append (*value*)

by_device_tag (*tag*)

extend (*iterable*)

virttest.libvirt_xml.vol_xml module Module simplifying manipulation of XML described at <http://libvirt.org/formatstorage.html#StorageVol>

```
class virttest.libvirt_xml.vol_xml.VolXML (vol_name='default', virsh_instance=<module
    'virttest.virsh'
    from
    '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.vol_xml.VolXMLBase`

Manipulators of a Virtual Vol through it's XML definition.

```
class Encryption (virsh_instance=<module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
    test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Encryption volume XML class

Properties:

format: string.

secret: dict, keys: type, uuid

format

secret

```
VolXML.create(pool_name, virsh_instance=<module 'virttest.virsh' from
               '/home/docs/checkouts/readthedocs.org/user_builds/virt-
               test/checkouts/latest/virttest/virsh.pyc'>)
```

Create volume with virsh from this instance

```
static VolXML.get_vol_details_by_name(vol_name, pool_name,
                                     virsh_instance=<module 'virttest.virsh' from
                                     '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                     test/checkouts/latest/virttest/virsh.pyc'>)
```

Return volume xml dictionary by Vol's uuid or name.

Parameters **vol_name** – Vol's name

Returns volume xml dictionary

```
VolXML.new_encryption(**dargs)
```

Return a new volume encryption instance and set properties from dargs

```
static VolXML.new_from_vol_dumpxml(vol_name, pool_name, virsh_instance=<module
                                   'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                   test/checkouts/latest/virttest/virsh.pyc'>)
```

Return new VolXML instance from virsh vol-dumpxml command

Parameters

- **vol_name** – Name of vol to vol-dumpxml
- **virsh_instance** – virsh module or instance to use

Returns New initialized VolXML instance

```
static VolXML.new_vol(**dargs)
```

Return a new VolXML instance and set properties from dargs

Parameters **dargs** – param dictionary

Returns new VolXML instance

```
class virttest.libvirt_xml.vol_xml.VolXMLBase(virsh_instance=<module 'virttest.virsh' from
                                               '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                               test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `virttest.libvirt_xml.base.LibvirtXMLBase`

Accessor methods for VolXML class.

Properties: **name:** string, operates on XML name tag **key:** string, operates on key tag **capacity:** integer, operates on capacity attribute of capacity tag **allocation:** integer, operates on allocation attribute of allocation tag **format:** string, operates on type attribute of format tag **path:** string, operates on path attribute of path tag **owner:** integer, operates on owner attribute of owner tag **group:** integer, operates on group attribute of group tag **mode:** string, operates on mode attribute of mode tag **label:** string, operates on label attribute of label tag **compat:** string, operates on compat attribute of label tag **lazy_refcounts:** bool, True/False **encryption:** VolXMLBase.Encryption instance. **capacity_unit:** string, operates on unit attribute of capacity tag

allocation

capacity

capacity_unit

compat

encryption
format
group
key
label
lazy_refcounts
mode
name
owner
path

virttest.libvirt_xml.xcepts module Module of common exceptions used in libvirt_xml package

exception `virttest.libvirt_xml.xcepts.LibvirtXMLAccessorError` (*details=''*)

Bases: `virttest.libvirt_xml.xcepts.LibvirtXMLError`

LibvirtXMLError related to an accessor generator class/method

exception `virttest.libvirt_xml.xcepts.LibvirtXMLError` (*details=''*)

Bases: `exceptions.Exception`

Error originating within libvirt_xml module

exception `virttest.libvirt_xml.xcepts.LibvirtXMLForbiddenError` (*details=''*)

Bases: `virttest.libvirt_xml.xcepts.LibvirtXMLError`

LibvirtXMLError raised when operating on a property is prohibited

exception `virttest.libvirt_xml.xcepts.LibvirtXMLNotFoundError` (*details=''*)

Bases: `virttest.libvirt_xml.xcepts.LibvirtXMLError`

LibvirtXMLError related when an element cannot be found

Module contents Intermediate module for working with XML-related virsh functions/methods.

The intention of this module is to hide the details of working with XML from test module code. Helper methods are all high-level and not conducive to direct use in error-testing. However, access to a virsh instance is available.

All classes defined here should inherit from LibvirtXMLBase and utilize the property-like interface provided by `utils_misc.PropCanBase` to manipulate XML from the 'xml' property. Please refer to the `xml_utils` module documentation for more information on working with `XMLTreeFile` instances. Please see the `virsh` and `utils_misc` modules for information on working with `Virsh` and `PropCanBase` classes.

All properties defined in `__slots__` are intended for test-module manipulation. External calling of accessor methods isn't forbidden, but discouraged. Instead, test modules should use normal reference, assignment, and delete operations on instance properties as if they were attributes. It's up to the test if it uses the dict-like or instance-attribute interface.

Internally, accessor methods (`get_*`(), `set_*`(), & `del_*`()) should always use `__dict_get__()`, `__dict_set__()`, and/or `__dict_del__()` to manipulate properties (otherwise infinite recursion can occur). In some cases, where class or instance attributes are needed (outside of `__slots__`) they must be accessed via the `__super_set__()`, `__super_get__()`, and/or `__super_del__()` methods. None of the `__super_*`() or the `__dict_*`() methods are intended for use by test-modules.

Errors originating beneath this module (e.g. w/in `virsh` or `libvirt_vm`) should not be caught (so caller can test for them). Errors detected within this module should raise `LibvirtXMLError` or a subclass.

virttest.qemu_devices package**Submodules**

virttest.qemu_devices.qbuses module Autotest representations of qemu buses.

These classes emulates the usual qemu buses behaviors in order to create or match the autotest params into qemu qdev structure.

copyright 2012-2013 Red Hat Inc.

class `virttest.qemu_devices.qbuses.QAHCIbus` (*busid, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QBusUnitBus`

AHCI bus (ich9-ahci, ahci)

class `virttest.qemu_devices.qbuses.QBusUnitBus` (*busid, bus_type, lengths, aobject=None, atype=None*)

Bases: `virttest.qemu_devices.qbuses.QDenseBus`

Implementation of bus-unit bus (ahci, ide)

class `virttest.qemu_devices.qbuses.QDenseBus` (*bus_item, addr_spec, busid, bus_type=None, aobject=None, atype=None*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

Dense bus representation. The only difference from SparseBus is the output string format. DenseBus iterates over all addresses and show free slots too. SparseBus on the other hand prints always the device address.

class `virttest.qemu_devices.qbuses.QDriveBus` (*busid, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

QDrive bus representation (single slot, drive=...)

get_free_slot (*addr_pattern*)

Use only drive as slot

class `virttest.qemu_devices.qbuses.QFloppyBus` (*busid, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QDenseBus`

Floppy bus (-global isa-fdc.drive?=\$drive)

class `virttest.qemu_devices.qbuses.QIDEBus` (*busid, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QBusUnitBus`

IDE bus (piix3-ide)

class `virttest.qemu_devices.qbuses.QNoAddrCustomBus` (*bus_item, addr_spec, busid, bus_type=None, aobject=None, atype=None*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

This is the opposite of QStrictCustomBus. Even when addr is set it's not updated in the device's params.

class `virttest.qemu_devices.qbuses.QOldFloppyBus` (*busid, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QDenseBus`

Floppy bus (-drive index=n)

class `virttest.qemu_devices.qbuses.QPCIBus` (*busid, bus_type, aobject=None, length=32, first_port=0*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

PCI Bus representation (bus&addr, uses hex digits)

class `virttest.qemu_devices.qbuses.QPCISwitchBus` (*busid, bus_type, downstream_type, aobject=None*)

Bases: `virttest.qemu_devices.qbuses.QPCIBus`

PCI Switch bus representation (creates downstream device while inserting a device).

add_downstream_port (*addr*)

Add downstream port of the certain address

class `virttest.qemu_devices.qbuses.QSCSIBus` (*busid, bus_type, addr_spec, aobject=None, atype=None*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

SCSI bus representation (bus + 2 leves, don't iterate over lun by default)

class `virttest.qemu_devices.qbuses.QSparseBus` (*bus_item, addr_spec, busid, bus_type=None, aobject=None, atype=None*)

Bases: `object`

Universal bus representation object.

It creates an abstraction of the way how buses works in qemu. Additionally it can store incorrect records (out-of-range addr, multiple devs, ...). Everything with `bad*` prefix means it concerns the bad records (`badbus`).

You can insert and remove device to certain address, address ranges or let the bus assign first free address. The order of `addr_spec` does matter since the last item is incremented first.

There are 3 different address representation used:

stor_addr stored address representation '`$first-$second-...-$ZZZ`'

addr internal address representation [`$first, $second, ..., $ZZZ`]

device_addr qemu address stored into separate device params (`bus, port`) `device{$param1:$first, $param2:$second, ..., $paramZZZ, $ZZZ}`

Note When you insert a device, it's properties might be updated (`addr,..`)

get (*item*)

Parameters *item* – autotest id or QObject-like object

Returns First matching object from this bus or None

get_device ()

Get device in which this bus is present

get_free_slot (*addr_pattern*)

Finds unoccupied address

Parameters *addr_pattern* – Address pattern (full qualified or with Nones)

Returns First free address when found, (free or reserved for this dev) None when no free address is found, (all occupied) False in case of incorrect address (oor)

insert (*device, strict_mode=False*)

Insert device into this bus representation.

Parameters

- **device** – `qdevices.QBaseDevice` device
- **strict_mode** – Use strict mode (set optional params)

Returns list of added devices on success, string indicating the failure on failure.

match_bus (*bus_spec*, *type_test=True*)

Check if the bus matches the bus_specification. :param bus_spec: Bus specification :type bus_spec: dict
:param type_test: Match only type :type type_test: bool :return: True when the bus matches the specification :rtype: bool

remove (*device*)

Remove device from this bus :param device: qdevices.QBaseDevice device :return: True when removed, False when the device wasn't found

reserve (*addr*)

Reserve the slot :param addr: Desired address :type addr: internal [addr1, addr2, ..] or stor format "addr1-addr2-.."

set_device (*device*)

Set the device in which this bus belongs

str_long ()

long string representation

str_short ()

short string representation

class virttest.qemu_devices.qbuses.QStrictCustomBus (*bus_item*, *addr_spec*, *busid*,
bus_type=None, *aobject=None*,
atype=None, *first_port=None*)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

Similar to QSparseBus. The address starts with 1 and addr is always set

class virttest.qemu_devices.qbuses.QUSBBus (*length*, *busid*, *bus_type*, *aobject=None*,
port_prefix=None)

Bases: `virttest.qemu_devices.qbuses.QSparseBus`

USB bus representation including usb-hub handling.

virttest.qemu_devices.qcontainer module Autotest qdev-structure representation.

This is the main class which represent qdev-structure. It allows to create, interact and verify the qemu qdev structure.

copyright 2012-2013 Red Hat Inc.

class virttest.qemu_devices.qcontainer.DevContainer (*qemu_binary*, *vm-*
name, *strict_mode='no'*,
workaround_qemu_qmp_crash='no',
allow_hotplugged_vm='yes')

Bases: `object`

Device container class

cdroms_define_by_params (*name*, *image_params*, *media=None*, *index=None*, *image_boot=None*,
image_bootindex=None)

Wrapper for creating cdrom and related hbas from autotest image params.

Note To skip the argument use None, to disable it use False

Note Strictly bool options accept "yes", "on" and True ("no"...))

Note Options starting with '_' are optional and used only when strict_mode is True

Parameters

- **name** – Name of the new disk
- **params** – Disk params (params.object_params(name))

cmdline (*dynamic=True*)

Creates cmdline arguments for creating all defined devices :return: cmdline of all devices (without qemu-cmd itself)

execute_qemu (*options, timeout=5*)

Execute this qemu and return the stdout+stderr output. :param options: additional qemu options :type options: string :param timeout: execution timeout :type timeout: int :return: Output of the qemu :rtype: string

get (*item*)

Parameters **item** – autotest id or QObject-like object

Returns First matching object defined in this QDevContainer or None

get_buses (*bus_spec, type_test=False*)

Parameters

- **bus_spec** (*dict*) – Bus specification (dictionary)
- **atype** (*bool*) – Match qemu and atype params

Returns All matching buses

Return type List of QSparseBus

get_by_params (*filt*)

Return list of matching devices :param filt: filter { 'param': 'value', ... } :type filt: dict

get_by_properties (*filt*)

Return list of matching devices :param filt: filter { 'property': 'value', ... } :type filt: dict

get_by_qid (*qid*)

Parameters **qid** – qemu id

Returns List of items with matching qemu id

get_first_free_bus (*bus_spec, addr*)

Parameters

- **bus_spec** – Bus specification (dictionary)
- **addr** – Desired address

Returns First matching bus with free desired address (the latest added matching bus)

get_help_text ()

Returns Full output of “qemu -help”

get_state ()

Get the current state (0 = synchronized with VM)

has_device (*device*)

Parameters **device** – Desired device

Returns Is the desired device supported by current qemu?

has_hmp_cmd (*cmd*)

Parameters **cmd** – Desired command

Returns Is the desired command supported by this qemu’s human monitor?

has_option (*option*)

Parameters *option* – Desired option

Returns Is the desired option supported by current qemu?

has_qmp_cmd (*cmd*)

Parameters *cmd* – Desired command

Returns Is the desired command supported by this qemu’s QMP monitor?

hook_fill_scsi_hbas (*params*)

This hook creates dummy scsi hba per 7 -drive ‘scsi’ devices.

hotplug_verified ()

This function should be used after you verify, that hotplug was successful. For each hotplug call, hotplug_verified have to be executed in order to mark VM as clear.

Warning If you can’t verify, that hotplug was successful, don’t use this function! You could screw-up following tests.

idx_of_next_named_bus (*bus_pattern*)

Parameters *bus_pattern* – Bus name prefix without %s and tailing digit

Returns Name of the next bus (integer is appended and incremented until there is no existing bus).

images_define_by_params (*name, image_params, media=None, index=None, image_boot=None, image_bootindex=None*)

Wrapper for creating disks and related hbas from autotest image params.

Note To skip the argument use None, to disable it use False

Note Strictly bool options accept “yes”, “on” and True (“no”...)

Note Options starting with ‘_’ are optional and used only when strict_mode is True

Parameters

- **name** – Name of the new disk
- **params** – Disk params (params.object_params(name))

images_define_by_variables (*name, filename, index=None, fmt=None, cache=None, werror=None, error=None, serial=None, snapshot=None, boot=None, blkdebug=None, bus=None, unit=None, port=None, bootindex=None, removable=None, min_io_size=None, opt_io_size=None, physical_block_size=None, logical_block_size=None, readonly=None, scsiid=None, lun=None, aio=None, strict_mode=None, media=None, imgfmt=None, pci_addr=None, scsi_hba=None, x_data_plane=None, blk_extra_params=None, scsi=None, pci_bus='pci.0', drv_extra_params=None, num_queues=None, bus_extra_params=None, force_fmt=None*)

Creates related devices by variables :note: To skip the argument use None, to disable it use False :note: Strictly bool options accept “yes”, “on” and True (“no”...) :param name: Autotest name of this disk :param filename: Path to the disk file :param index: drive index (used for generating names) :param fmt: drive subsystem type (ide, scsi, virtio, usb2, ...) :param force_fmt: Force to use specific drive format :param cache: disk cache (none, writethrough, writeback) :param werror: What to do when write error occurs (stop, ...) :param error: What to do when read error occurs (stop, ...) :param serial: drive serial number (\$string) :param snapshot: use snapshot? (\$bool) :param boot: is bootable? (\$bool) :param blkdebug: use blkdebug (None, blkdebug_filename) :param bus: 1st level of disk location (index of bus) (\$int) :param unit: 2nd level of disk location (unit/scsiid/...) (\$int) :param port: 3rd level of disk location (port/lun/...)

(*\$int*) :param bootindex: device boot priority (*\$int*) :param removable: can the drive be removed? (*\$bool*) :param min_io_size: Min allowed io size :param opt_io_size: Optimal io size :param physical_block_size: set physical_block_size (*\$int*) :param logical_block_size: set logical_block_size (*\$int*) :param readonly: set the drive readonly (*\$bool*) :param scsiid: Deprecated 2nd level of disk location (&unit) :param lun: Deprecated 3rd level of disk location (&port) :param aio: set the type of async IO (native, threads, ..) :param strict_mode: enforce optional parameters (address, ...) (*\$bool*) :param media: type of the media (disk, cdrom, ...) :param imgfmt: image format (qcow2, raw, ...) :param pci_addr: drive pci address (*\$int*) :param scsi_hba: Custom scsi HBA :param num_queues: performace option for virtio-scsi-pci :param bus_extra_params: options want to add to virtio-scsi-pci bus

insert (*devices, strict_mode=None*)

Inserts devices into this VM representation :param devices: List of `qdevices.QBaseDevice` devices :raise `DeviceError`: On failure. The representation remains unchanged.

list_missing_named_buses (*bus_pattern, bus_type, bus_count*)

Parameters

- **bus_pattern** – Bus name pattern with `1x%s` for `idx` or `%s` is appended in the end. ('mybuses' or 'my%sbus').
- **bus_type** – Type of the bus.
- **bus_count** – Desired number of buses.

Returns List of buses, which are missing in range(`bus_count`)

machine_by_params (*params=None*)

Choose the used machine and set the default devices accordingly :param params: VM params :return: List of added devices (including default buses)

pcic_by_params (*name, params*)

Creates pci controller/switch/... based on params

Parameters

- **name** – Autotest name
- **params** – PCI controller params

Note x3130 creates x3130-upstream bus + xio3130-downstream port for each inserted device.

Warning x3130-upstream device creates only x3130-upstream device and you are responsible for creating the downstream ports.

remove (*device, recursive=True*)

Remove device from this representation :param device: autotest id or `QObject`-like object :param recursive: remove children recursively :return: None on success, -1 when the device is not present

reset_state ()

Mark representation as completely clean, without hotplugged devices.

set_clean ()

Decrease VM dirtiness (synchronized with VM)

set_dirty ()

Increase VM dirtiness (not synchronized with VM)

simple_hotplug (*device, monitor*)

Function hotplug device to devices representation. If verification is supported by hotplugged device and result of verification is `True` then it calls `set_clean`. Otherwise it don't call `set_clean` because devices representatio don't know if device is added correctly.

Parameters

- **device** (*string*, *qdevices.QDevice*.) – Device which should be unplugged.
- **monitor** (*qemu_monitor.Monitor*) – Monitor from vm.

Returns tuple(monitor.cmd(), verify_hotplug output)

simple_unplug (*device*, *monitor*)

Function unplug device to devices representation. If verification is supported by unplugged device and result of verification is True then it calls set_clean. Otherwise it don't call set_clean because devices representatio don't know if device is added correctly.

Parameters

- **device** (*string*, *qdevices.QDevice*.) – Device which should be unplugged.
- **monitor** (*qemu_monitor.Monitor*) – Monitor from vm.

Returns tuple(monitor.cmd(), verify_unplug output)

str_bus_long ()

Long representation of all buses

str_bus_short ()

Short representation of all buses

str_long ()

Long string representation of all devices

str_short ()

Short string representation of all devices

usb_by_params (*usb_name*, *params*)

Wrapper for creating usb devices from autotest params. :param usb_name: Name of the usb :param params: USB device's params :return: QDev device

usb_by_variables (*usb_name*, *usb_type*, *controller_type*, *bus=None*, *port=None*)

Creates usb-devices by variables. :param usb_name: usb name :param usb_type: usb type (usb-tablet, usb-serial, ...) :param controller_type: type of the controller (uhci, ehci, xhci, ...) :param bus: the bus name (my_bus.0, ...) :param port: port specifiaction (4, 4.1.2, ...) :return: QDev device

usbc_by_params (*usb_name*, *params*)

Wrapper for creating usb bus from autotest usb params. :param usb_name: Name of the usb bus :param params: USB params (params.object_params(usb_name)) :return: List of QDev devices

usbc_by_variables (*usb_id*, *usb_type*, *multifunction=False*, *masterbus=None*, *firstport=None*, *freq=None*, *max_ports=6*, *pci_addr=None*, *pci_bus='pci.0'*)

Creates usb-controller devices by variables :param usb_id: Usb bus name :param usb_type: Usb bus type :param multifunction: Is the bus multifunction :param masterbus: Is this bus master? :param firstport: Offset of the first port :param freq: Bus frequency :param max_ports: How many ports this bus have [6] :param pci_addr: Desired PCI address :return: List of QDev devices

wash_the_device_out (*device*)

Removes any traces of the device from representation. :param device: qdevices.QBaseDevice device

virttest.qemu_devices.qdevices module Autotest representation of qemu devices.

These classes implements various features in order to simulate, verify or interact with qemu qdev structure.

copyright 2012-2013 Red Hat Inc.

```
class virttest.qemu_devices.qdevices.QBaseDevice (dev_type='QBaseDevice',  
                                                  params=None, aobject=None, par-  
ent_bus=None, child_bus=None)
```

Bases: `object`

Base class of qemu objects

add_child_bus (*bus*)

Add child bus :param bus: Bus, which this device contains :type bus: QSparseBus-like

cmdline ()

Returns cmdline command to define this device

cmdline_nd ()

Command line without dynamic params.

Returns cmdline command to define this device without dynamic parameters

get_aid ()

Returns per VM unique autotest_id

get_children ()

Returns List of all children (recursive)

get_param (*option, default=None*)

Returns object param

get_qid ()

Returns qemu_id

hotplug (*monitor*)

Returns the output of monitor.cmd() hotplug command

hotplug_hmp ()

Returns the hotplug monitor command

hotplug_qmp ()

Returns tuple(hotplug qemu command, arguments)

rm_child_bus (*bus*)

removes child bus :param bus: Bus, which this device contains :type bus: QSparseBus-like

set_aid (*aid*)

Parameters *aid* – new autotest id for this device

set_param (*option, value, option_type=None, dynamic=False*)

Set device param using qemu notation (“on”, “off” instead of bool...) :param option: which option’s value to set :param value: new value :param option_type: type of the option (bool) :param dynamic: if true value is changed to DYN for not_dynamic compare

str_long ()

Full representation, multi-line with all params

str_short ()

Short representation (aid, qid, alternative, type)

unplug (*monitor*)

Returns the output of monitor.cmd() unplug command

unplug_hmp()

Returns the unplug monitor command

unplug_hook()

Modification prior to unplug can be made here

unplug_qmp()

Returns tuple(unplug qemu command, arguments)

unplug_unhook()

Roll back the modification made before unplug

verify_hotplug(out, monitor)

Parameters

- **out** – Output of the hotplug command
- **monitor** – Monitor used for hotplug

Returns True when successful, False when unsuccessful, string/None when can't decide.

verify_unplug(out, monitor)

Parameters

- **out** – Output of the unplug command
- **monitor** – Monitor used for unplug

class virttest.qemu_devices.qdevices.**QCustomDevice**(*dev_type, params=None, aobject=None, parent_bus=None, child_bus=None, backend=None*)

Bases: *virttest.qemu_devices.qdevices.QBaseDevice*

Representation of the '-\$option \$param1=\$value1,\$param2...' qemu object. This representation handles only cmdline.

cmdline()

Returns cmdline command to define this device

cmdline_nd()

Command line without dynamic parameters.

Returns cmdline command to define this device without dynamic parameters.

class virttest.qemu_devices.qdevices.**QDevice**(*driver=None, params=None, aobject=None, parent_bus=None, child_bus=None*)

Bases: *virttest.qemu_devices.qdevices.QCustomDevice*

Representation of the '-device' qemu object. It supports all methods. :note: Use driver format in full form - 'driver' = '...' (usb-ehci, ide-hd)

get_children()

Device bus should be removed too

hotplug_hmp()

Returns the hotplug monitor command

hotplug_hmp_nd()

Returns the hotplug monitor command without dynamic parameters

hotplug_qmp()

Returns the hotplug monitor command

hotplug_qmp_nd()

Returns the hotplug monitor command without dynamic parameters

unplug_hmp()

Returns the unplug monitor command

unplug_qmp()

Returns the unplug monitor command

verify_hotplug(out, monitor)

verify_unplug(out, monitor)

class `virttest.qemu_devices.qdevices.QDrive(aobject, use_device=True)`

Bases: `virttest.qemu_devices.qdevices.QCustomDevice`

Representation of the ‘-drive’ qemu object without hotplug support.

set_param(option, value, option_type=None)

Set device param using qemu notation (“on”, “off” instead of bool...) It restricts setting of the ‘id’ param as it’s automatically created. :param option: which option’s value to set :param value: new value :param option_type: type of the option (bool)

class `virttest.qemu_devices.qdevices.QFloppy(unit=None, drive=None, aobject=None, parent_bus=None, child_bus=None)`

Bases: `virttest.qemu_devices.qdevices.QGlobal`

Imitation of qemu floppy disk defined by -global isa-fdc.drive?=\$drive

set_param(option, value, option_type=None)

drive and unit params have to be ‘translated’ as value and property.

class `virttest.qemu_devices.qdevices.QGlobal(driver, prop, value, aobject=None, parent_bus=None, child_bus=None)`

Bases: `virttest.qemu_devices.qdevices.QBaseDevice`

Representation of qemu global setting (-global driver.property=value)

cmdline()

class `virttest.qemu_devices.qdevices.QHPDrive(aobject)`

Bases: `virttest.qemu_devices.qdevices.QDrive`

Representation of the ‘-drive’ qemu object with hotplug support.

get_children()

Device bus should be removed too

hotplug_hmp()

Returns the hotplug monitor command

unplug_hmp()

Returns the unplug monitor command

unplug_hook()

Devices from this bus are not removed, only ‘drive’ is set to None.

unplug_unhook()

Set back the previous ‘drive’ (unsafe, using the last value)

verify_hotplug(out, monitor)

verify_unplug (*out, monitor*)

class `virttest.qemu_devices.qdevices.QOldDrive` (*aobject, use_device=True*)

Bases: `virttest.qemu_devices.qdevices.QDrive`

This is a variant for -drive without 'addr' support

set_param (*option, value, option_type=None*)

Ignore addr parameters as they are not supported by old qemus

class `virttest.qemu_devices.qdevices.QRHDrive` (*aobject*)

Bases: `virttest.qemu_devices.qdevices.QDrive`

Representation of the '-drive' qemu object with RedHat hotplug support.

get_children ()

Device bus should be removed too

hotplug_hmp ()

Returns the hotplug monitor command

hotplug_qmp ()

Returns the hotplug monitor command

unplug_hmp ()

Returns the unplug monitor command

unplug_hook ()

Devices from this bus are not removed, only 'drive' is set to None.

unplug_qmp ()

Returns the unplug monitor command

unplug_unhook ()

Set back the previous 'drive' (unsafe, using the last value)

class `virttest.qemu_devices.qdevices.QStringDevice` (*dev_type='dummy', params=None, aobject=None, parent_bus=None, child_bus=None, cmdline=', cmdline_nd=None*)

Bases: `virttest.qemu_devices.qdevices.QBaseDevice`

General device which allows to specify methods by fixed or parametrizable strings in this format:

```
"%(type)s,id=%(id)s,addr=%(addr)s"
```

params will be used to subst % () s

cmdline ()

Returns cmdline command to define this device

cmdline_nd ()

Command line without dynamic parameters.

Returns cmdline command to define this device without dynamic parameters.

virttest.qemu_devices.utils module Shared classes and functions (exceptions, ...)

copyright 2013 Red Hat Inc.

exception `virttest.qemu_devices.utils.DeviceError`

Bases: `exceptions.Exception`

General device exception

exception `virttest.qemu_devices.utils.DeviceHotplugError` (*device, reason, vmdev*)

Bases: `virttest.qemu_devices.utils.DeviceInsertError`

Fail to hotplug device

exception `virttest.qemu_devices.utils.DeviceInsertError` (*device, reason, vmdev*)

Bases: `virttest.qemu_devices.utils.DeviceError`

Fail to insert device

exception `virttest.qemu_devices.utils.DeviceRemoveError` (*device, reason, vmdev*)

Bases: `virttest.qemu_devices.utils.DeviceInsertError`

Fail to remove device

exception `virttest.qemu_devices.utils.DeviceUnplugError` (*device, reason, vmdev*)

Bases: `virttest.qemu_devices.utils.DeviceHotplugError`

Fail to unplug device

`virttest.qemu_devices.utils.none_or_int` (*value*)

Helper fcton which returns None or int()

Module contents

`virttest.remote_commander` package

Submodules

virttest.remote_commander.messenger module Created on Dec 6, 2013

author jzupka

class `virttest.remote_commander.messenger.DataWrapper`

Bases: `object`

Basic implementation of IOWrapper for stdio.

decode (*data*)

Decodes the data which was read.

Returns decoded data.

encode (*data*)

Encode data.

Returns encoded data.

class `virttest.remote_commander.messenger.DataWrapperBase64`

Bases: `virttest.remote_commander.messenger.DataWrapper`

Basic implementation of IOWrapper for stdio.

decode (*data*)

encode (*data*)

class `virttest.remote_commander.messenger.IOWrapper` (*obj*)

Bases: `object`

Class encapsulates io operation to be more consist in different implementations. (stdio, sockets, etc..)

close()

fileno()

Function should return file descriptor number. If object should be used for standard io operation.

Returns File number.

read (*max_len*, *timeout=None*)

Read function should be reinplemented as blocking reading from data source when timeout is None and nonblocking for timeout is not None. Implementation example StdIWrapper.

Params **max_len** Max len of readed data.

Parameters **timeout** (*float*) – Timeout of reading operation.

Returns Readed data.

write (*data*)

Write function should be implemented for object used for writing.

Parameters **data** (*str.*) – Data to write.

class `virttest.remote_commander.messenger.Messenger` (*stdin*, *stdout*)

Bases: `object`

Class could be used for communication between two python process connected by communication canal wrapped by IOWrapper class. Pickling is used for communication and thus it is possible to communicate every picleable object.

close()

flush_stdin()

Flush all input data from communication interface.

format_msg (*data*)

Format message where first 10 char is length of message and rest is pickled message.

read_msg (*timeout=None*)

Read data from com interface.

Parameters **timeout** (*float*) – timeout for reading data.

Returns (True, data) when reading is successful. (False, None) when other side is closed. (None, None) when reading is timeouted.

write_msg (*data*)

Write formatted message to communication interface.

exception `virttest.remote_commander.messenger.MessengerError` (*msg*)

Bases: `exceptions.Exception`

class `virttest.remote_commander.messenger.StdIOWrapper` (*obj*)

Bases: `virttest.remote_commander.messenger.IOWrapper`,
`virttest.remote_commander.messenger.DataWrapper`

Basic implementation of IOWrapper for stdio.

close()

fileno()

class `virttest.remote_commander.messenger.StdIOWrapperIn` (*obj*)

Bases: `virttest.remote_commander.messenger.StdIOWrapper`

Basic implementation of IOWrapper for stdin

read (*max_len*, *timeout=None*)

class `virttest.remote_commander.messenger.StdIOWrapperInBase64` (*obj*)

Bases: `virttest.remote_commander.messenger.StdIOWrapperIn`,
`virttest.remote_commander.messenger.DataWrapperBase64`

Basic implementation of IOWrapper for stdin

class `virttest.remote_commander.messenger.StdIOWrapperOut` (*obj*)

Bases: `virttest.remote_commander.messenger.StdIOWrapper`

Basic implementation of IOWrapper for stdout

write (*data*)

class `virttest.remote_commander.messenger.StdIOWrapperOutBase64` (*obj*)

Bases: `virttest.remote_commander.messenger.StdIOWrapperOut`,
`virttest.remote_commander.messenger.DataWrapperBase64`

Basic implementation of IOWrapper for stdout

virttest.remote_commander.remote_interface module Created on Dec 11, 2013

author jzupka

class `virttest.remote_commander.remote_interface.BaseCmd` (*func_cmd*, **args*, ***kargs*)

Bases: `virttest.remote_commander.remote_interface.CmdMessage`

Class used for moving information about commands between master and slave.

args

cmd_hash

func

is_async ()

Returns True if command is async else False

is_finished ()

Returns True if command is finished else False

kargs

nh_stderr

nh_stdin

nh_stdout

results

single_cmd_id = 0

update (*basecmd*)

Sync local class with class moved over the messenger.

Parameters **basecmd** (`BaseCmd`) – basecmd from which should be sync data to this instance

update_cmd_hash (*basecmd*)

```
class virttest.remote_commander.remote_interface.CmdMessage(cmd_id)
    Bases: object
    Base cmd message class

    cmd_id

    isCmdMsg()

exception virttest.remote_commander.remote_interface.CmdTraceBack(msg)
    Bases: exceptions.Exception
    Represent back-trace used for error tracing on remote side.

exception virttest.remote_commander.remote_interface.CommanderError(msg)
    Bases: virttest.remote_commander.remote_interface.MessengerError
    Represent error in Commnader

exception virttest.remote_commander.remote_interface.MessengerError(msg)
    Bases: exceptions.Exception
    Represented error in messenger.

class virttest.remote_commander.remote_interface.Stderr(msg, cmd_id=None)
    Bases: virttest.remote_commander.remote_interface.StdStream
    Represent message from stderr string data from remote client

class virttest.remote_commander.remote_interface.Stdout(msg, cmd_id=None)
    Bases: virttest.remote_commander.remote_interface.StdStream
    Represent message from stdout string data from remote client

class virttest.remote_commander.remote_interface.StdStream(msg, cmd_id=None)
    Bases: virttest.remote_commander.remote_interface.CmdMessage
    Represent message string data from remote client

    msg

virttest.remote_commander.remote_master module   Created on Dec 6, 2013

    author jzupka

class virttest.remote_commander.remote_master.CmdEncapsulation(master, obj_name,
                                                                name)
    Bases: object
    Class parse command name cmd.nohup.shell -> ["nohup", "shell"]

class virttest.remote_commander.remote_master.CmdMaster(commander, name, *args,
                                                         **kargs)
    Bases: object
    Representation of BaseCmd on master side.

    basecmd
        Property basecmd getter

    getbasecmd()
        Property basecmd getter

    getstderr()
        Property stderr getter
```

getstdout ()

Property stdout getter

send_stdin (*msg*)

Send data to stdin

set_commander (*commander*)

For nohup commands it allows connect cmd to new created commander.

setbasecmd (*value*)

Property basecmd setter _results_cnt identify if value was change from last reading.

setstderr (*value*)

Property stderr setter _stderr_cnt identify if value was change from last reading.

setstdout (*value*)

Property stdout setter _stdout_cnt identify if value was change from last reading.

stderr

Property stderr getter

stdout

Property stdout getter

wait ()

Wait until command return results.

wait_response (*timeout=None*)

Wait until command return any cmd.

exception `virttest.remote_commander.remote_master.CmdTimeout` (*msg*)

Bases: `virttest.remote_commander.remote_interface.MessengerError`

Raised when waiting for cmd exceeds time define by timeout.

class `virttest.remote_commander.remote_master.Commander`

Bases: `object`

Commander representation for transfer over network.

class `virttest.remote_commander.remote_master.CommanderMaster` (*stdin, stdout, debug=False*)

Bases: `virttest.remote_commander.messenger.Messenger`

Class commander master is responsible for communication with commander slave. It invoke commands to slave part and receive messages from them. For communication is used only stdin and stdout which are streams from slave part.

close ()

cmd (*cmd, timeout=60*)

Invoke command on client side.

listen_cmds (*cmd*)

Manage basecmds from slave side.

listen_errors (*cmd*)

Listen for errors raised from slave part of commander.

listen_messenger (*timeout=60*)

Wait for msg from slave side and take care about them.

listen_streams (*cmd*)

Listen on all streams included in Commander commands.

wait (*cmd*, *timeout=60*)

Wait until command return results.

wait_response (*cmd*, *timeout=60*)

Wait until command return any cmd.

`virttest.remote_commander.remote_master.getsource(obj)`

`virttest.remote_commander.remote_master.wait_timeout(timeout)`

virttest.remote_commander.remote_runner module Created on Dec 6, 2013

author jzupka

class `virttest.remote_commander.remote_runner.CmdFinish` (*parent=False*)

Bases: `object`

Class used for communication with child process. This class

pid

class `virttest.remote_commander.remote_runner.CmdSlave` (*baseCmd*)

Bases: `object`

Representation of BaseCmd on slave side.

close_pipes ()

Close command communication pipe.

finish (*commander*)

Remove cmd from commander commands on finish of process.

parse_func_name (*func_name*, *commander*)

Parse name sended from master.

format: ["manage|async|nohup| ", "fnnam1", "fnnam2", ...]

Parameters

- **func_name** – Function name
- **commander** – Where to execute the command (remote or local)

recover_fds ()

Helper function for reconnect to daemon/nohup process.

recover_paths ()

Helper function for reconnect to daemon/nohup process.

work ()

Wait for message from running child process

class `virttest.remote_commander.remote_runner.CommanderSlave` (*stdin*, *stdout*, *o_stdout*,
o_stderr)

Bases: `virttest.remote_commander.messenger.Messenger`

Class commander slace is responsible for communication with commander master. It invoke commands to slave part and receive messages from them. For communication is used only stdin and stdout which are streams from slave part.

cmd_loop ()

Wait for commands from master and receive results and outputs from commands.

class `virttest.remote_commander.remote_runner.CommanderSlaveCmds` (*stdin*, *stdout*, *o_stdout*, *o_stderr*)

Bases: `virttest.remote_commander.remote_runner.CommanderSlave`

Class extends `CommanderSlave` and adds to them special commands like shell process, interactive python, `send_msg` to cmd.

add_function (*f_code*)

Adds function to client code.

Parameters *f_code* (*str.*) – Code of function.

copy_file (*name*, *path*, *content*)

Really naive implementation of copying files. Should be used only for short files.

exit ()

Method for killing command slave.

import_src (*name*, *path*=None)

Import file to running python session.

interactive ()

Starts interactive python.

register_cmd (*basecmd*, *basecmd_cmd_id*)

Second side of `set_commander cmd` from master. It register existing cmd to `CommandSlave` dict.

Parameters

- **basecmd** (*BaseCmd*) – cmd which should be added to `CommandSlave` dict
- **basecmd_cmd_id** (*int*) – number under which should be stored

send_msg (*msg*, *cmd_id*)

Send msg to cmd with `id == cmd_id`

Parameters

- **msg** (*str*) – message passed to cmd over the `stdin`
- **cmd_id** – id of cmd.

shell (*cmd*)

Starts shell process. Stdout is automatically copyed to `basecmd.stdout`

Parameters *cmd* – Command which should be started.

Returns `basecmd` with return code of cmd.

`virttest.remote_commander.remote_runner.clean_tmp_dir` (*path*)

Clean up directory.

`virttest.remote_commander.remote_runner.close_unused_fds` (*fds*)

Close all file descriptors which are not necessary anymore.

Parameters *fds* (*list []*) – file descriptors

`virttest.remote_commander.remote_runner.create_process_cmd` ()

Create child process without clean process data thanks that it is possible call function and classes from child process.

`virttest.remote_commander.remote_runner.daemonize` (*pipe_root_path*='tmp')

Init daemon.

Parameters *pipe_root_path* – path to directory for pipe.

Returns [True if child, stdin_path, stdout_path, stderr_path]

`virttest.remote_commander.remote_runner.gen_tmp_dir(root_path)`

Try to create tmp dir with special name.

`virttest.remote_commander.remote_runner.remote_agent(in_stream_cls,
out_stream_cls)`

Connect file descriptors to right pipe and start slave command loop. When something happend it raise exception which could be caught by cmd master.

Params in_stream_cls Class encapsulated input stream.

Params out_stream_cls Class encapsulated output stream.

`virttest.remote_commander.remote_runner.sort_fds_event(fds)`

Module contents

virttest.staging package

Subpackages

virttest.staging.backports package

Subpackages

virttest.staging.backports.collections package

Submodules

virttest.staging.backports.collections.OrderedDict module Backport of OrderedDict() class that runs on Python 2.4, 2.5, 2.6, 2.7 and pypy. Passes Python2.7's test suite and incorporates all the latest updates.

Obtained from: <http://code.activestate.com/recipes/576693-ordered-dictionary-for-py24/>

class `virttest.staging.backports.collections.OrderedDict.OrderedDict(*args,
**kwargs)`

Bases: `dict`

Dictionary that remembers insertion order

<http://code.activestate.com/recipes/576693-ordered-dictionary-for-py24/> :codeauthor: Raymond Hettinger :license: MIT

clear () → None. Remove all items from od.

copy () → a shallow copy of od

classmethod fromkeys (S[, v]) → New ordered dictionary with keys from S and values equal to v (which defaults to None).

items () → list of (key, value) pairs in od

iteritems ()
od.iteritems -> an iterator over the (key, value) items in od

iterkeys () → an iterator over the keys in od

itervalues ()

od.itervalues -> an iterator over the values in od

keys () -> list of keys in od

pop (k[, d]) -> v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

popitem () -> (k, v), return and remove a (key, value) pair.

Pairs are returned in LIFO order if last is true or FIFO order if false.

setdefault (k[, d]) -> od.get(k,d), also set od[k]=d if k not in od

update (E, **F) -> None. Update od from dict/iterable E and F.

If E is a dict instance, does: for k in E: od[k] = E[k] If E has a .keys() method, does: for k in E.keys(): od[k] = E[k] Or if E is an iterable of items, does: for k, v in E: od[k] = v In either case, this is followed by: for k, v in F.items(): od[k] = v

values () -> list of values in od

viewitems () -> a set-like object providing a view on od's items

viewkeys () -> a set-like object providing a view on od's keys

viewvalues () -> an object providing a view on od's values

virttest.staging.backports.collections.defaultdict module Backport of the defaultdict module, obtained from: <http://code.activestate.com/recipes/523034-emulate-collectionsdefaultdict/>

class virttest.staging.backports.collections.defaultdict.**defaultdict** (default_factory=None, *a, **kw)

Bases: dict

collections.defaultdict is a handy shortcut added in Python 2.5 which can be emulated in older versions of Python. This recipe tries to backport defaultdict exactly and aims to be safe to subclass and extend without worrying if the base class is in C or is being emulated.

<http://code.activestate.com/recipes/523034-emulate-collectionsdefaultdict/> :codeauthor: Jason Kirtland :license: PSF

Changes: * replaced self.items() with self.iteritems() to fix Pickle bug as recommended by Aaron Lav * reformat with autopep8

copy ()

virttest.staging.backports.collections.namedtuple module This module contains a backport for collections.namedtuple obtained from <http://code.activestate.com/recipes/500261-named-tuples/>

virttest.staging.backports.collections.namedtuple.**namedtuple** (typename, field_names, verbose=False, rename=False)

Returns a new subclass of tuple with named fields.

```
>>> Point = namedtuple('Point', 'x y')
>>> Point.__doc__                # docstring for the new class
'Point(x, y)'
>>> p = Point(11, y=22)          # instantiate with positional args or keywords
>>> p[0] + p[1]                  # indexable like a plain tuple
33
```

```

>>> x, y = p                                # unpack like a regular tuple
>>> x, y
(11, 22)
>>> p.x + p.y                                # fields also accessible by name
33
>>> d = p._asdict()                          # convert to a dictionary
>>> d['x']
11
>>> Point(**d)                               # convert from a dictionary
Point(x=11, y=22)
>>> p._replace(x=100)                        # _replace() is like str.replace() but targets named fields
Point(x=100, y=22)

```

<http://code.activestate.com/recipes/500261-named-tuples/> :codeauthor: Raymond Hettinger :license: PSF

Changes: * autopep8 reformatting

Module contents

virttest.staging.backports.simplejson package

Submodules

virttest.staging.backports.simplejson.decoder module Implementation of JSONDecoder

```

class virttest.staging.backports.simplejson.decoder.JSONDecoder(encoding=None,
                                                                ob-
                                                                ject_hook=None,
                                                                parse_float=None,
                                                                parse_int=None,
                                                                parse_constant=None,
                                                                strict=True, ob-
                                                                ject_pairs_hook=None)

```

Bases: `object`

Simple JSON <<http://json.org>> decoder

Performs the following translations in decoding by default:

JSON	Python
object	dict
array	list
string	unicode
number (int)	int, long
number (real)	float
true	True
false	False
null	None

It also understands NaN, Infinity, and -Infinity as their corresponding float values, which is outside the JSON spec.

decode (*s*, *_w*=<built-in method match of *_sre.SRE_Pattern* object>)

Return the Python representation of *s* (a `str` or `unicode` instance containing a JSON document)

raw_decode(*s*, *idx=0*)

Decode a JSON document from *s* (a `str` or `unicode` beginning with a JSON document) and return a 2-tuple of the Python representation and the index in *s* where the document ended.

This can be used to decode a JSON document from a string that may have extraneous data at the end.

virttest.staging.backports.simplejson.encoder module Implementation of JSONEncoder

```
class virttest.staging.backports.simplejson.encoder.JSONEncoder(skipkeys=False,
                                                                en-
                                                                sure_ascii=True,
                                                                check_circular=True,
                                                                allow_nan=True,
                                                                sort_keys=False,
                                                                indent=None,
                                                                separators=None,
                                                                encoding='utf-
                                                                8', default=None,
                                                                use_decimal=False)
```

Bases: `object`

Extensible JSON <<http://json.org>> encoder for Python data structures.

Supports the following objects and types by default:

Python	JSON
dict	object
list, tuple	array
str, unicode	string
int, long, float	number
True	true
False	false
None	null

To extend this to recognize other objects, subclass and implement a `.default()` method with another method that returns a serializable object for *o* if possible, otherwise it should call the superclass implementation (to raise `TypeError`).

default(*o*)

Implement this method in a subclass such that it returns a serializable object for *o*, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement `default` like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    return JSONEncoder.default(self, o)
```

encode(*o*)

Return a JSON string representation of a Python data structure.

```
>>> from simplejson import JSONEncoder
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

item_separator = ‘,’

iterencode (*o*, *_one_shot=False*)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

key_separator = ‘:’

```
class virttest.staging.backports.simplejson.encoder.JSONEncoderForHTML(skipkeys=False,
    ensure_ascii=True,
    check_circular=True,
    allow_nan=True,
    sort_keys=False,
    indent=None,
    separators=None,
    encoding='utf-8', default=None,
    use_decimal=False)
```

Bases: `virttest.staging.backports.simplejson.encoder.JSONEncoder`

An encoder that produces JSON safe to embed in HTML.

To embed JSON content in, say, a script tag on a web page, the characters &, < and > should be escaped. They cannot be escaped with the usual entities (e.g. &) because they are not expanded within <script> tags.

encode (*o*)

iterencode (*o*, *_one_shot=False*)

`virttest.staging.backports.simplejson.encoder.encode_basestring` (*s*)

Return a JSON representation of a Python string

`virttest.staging.backports.simplejson.encoder.encode_basestring_ascii` (*s*)

Return an ASCII-only JSON representation of a Python string

`virttest.staging.backports.simplejson.encoder.py_encode_basestring_ascii` (*s*)

Return an ASCII-only JSON representation of a Python string

virttest.staging.backports.simplejson.ordered_dict module Drop-in replacement for `collections.OrderedDict` by Raymond Hettinger

<http://code.activestate.com/recipes/576693/>

```
class virttest.staging.backports.simplejson.ordered_dict.OrderedDict(*args,
    **kwargs)
```

Bases: `dict`, `UserDict.DictMixin`

clear ()

copy ()

classmethod fromkeys (*iterable*, *value=None*)

```
items()
iteritems()
iterkeys()
itervalues()
keys()
pop(key, *args)
popitem(last=True)
setdefault(key, default=None)
update(other=None, **kwargs)
values()
```

virttest.staging.backports.simplejson.scanner module JSON token scanner

`virttest.staging.backports.simplejson.scanner.make_scanner(context)`

virttest.staging.backports.simplejson.tool module

Module contents JSON (JavaScript Object Notation) <<http://json.org>> is a subset of JavaScript syntax (ECMA-262 3rd edition) used as a lightweight data interchange format.

`simplejson` exposes an API familiar to users of the standard library `marshal` and `pickle` modules. It is the externally maintained version of the `json` library contained in Python 2.6, but maintains compatibility with Python 2.4 and Python 2.5 and (currently) has significant performance advantages, even without using the optional C extension for speedups.

Encoding basic Python object hierarchies:

```
>>> import simplejson as json
>>> json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])
'["foo", {"bar": ["baz", null, 1.0, 2]}]'
>>> print json.dumps("\foo\bar")
"\foo\bar"
>>> print json.dumps(u'\u1234')
"\u1234"
>>> print json.dumps('\')
"\"
>>> print json.dumps({"c": 0, "b": 0, "a": 0}, sort_keys=True)
{"a": 0, "b": 0, "c": 0}
>>> from StringIO import StringIO
>>> io = StringIO()
>>> json.dump(['streaming API'], io)
>>> io.getvalue()
'["streaming API"]'
```

Compact encoding:

```
>>> import simplejson as json
>>> json.dumps([1,2,3,{ '4': 5, '6': 7}], separators=(',', ':'))
'[1,2,3,{"4":5,"6":7}]'
```

Pretty printing:


```
>>> import simplejson as json
>>> s = json.dumps({'4': 5, '6': 7}, sort_keys=True, indent='    ')
>>> print '\n'.join([l.rstrip() for l in s.splitlines()])
{
    "4": 5,
    "6": 7
}
```

Decoding JSON:

```
>>> import simplejson as json
>>> obj = [u'foo', {u'bar': [u'baz', None, 1.0, 2]}]
>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]') == obj
True
>>> json.loads('"\"foo\"bar\"') == u'"foo\x08ar'
True
>>> from StringIO import StringIO
>>> io = StringIO('["streaming API"]')
>>> json.load(io)[0] == 'streaming API'
True
```

Specializing JSON object decoding:

```
>>> import simplejson as json
>>> def as_complex(dct):
...     if '__complex__' in dct:
...         return complex(dct['real'], dct['imag'])
...     return dct
...
>>> json.loads('{"__complex__": true, "real": 1, "imag": 2}',
...     object_hook=as_complex)
(1+2j)
>>> from decimal import Decimal
>>> json.loads('1.1', parse_float=Decimal) == Decimal('1.1')
True
```

Specializing JSON object encoding:

```
>>> import simplejson as json
>>> def encode_complex(obj):
...     if isinstance(obj, complex):
...         return [obj.real, obj.imag]
...     raise TypeError(repr(o) + " is not JSON serializable")
...
>>> json.dumps(2 + 1j, default=encode_complex)
'[2.0, 1.0]'
>>> json.JSONEncoder(default=encode_complex).encode(2 + 1j)
'[2.0, 1.0]'
>>> ''.join(json.JSONEncoder(default=encode_complex).iterencode(2 + 1j))
'[2.0, 1.0]'
```

Using simplejson.tool from the shell to validate and pretty-print:

```
$ echo '{"json": "obj"}' | python -m simplejson.tool
{
    "json": "obj"
}
$ echo '{ 1.2:3.4}' | python -m simplejson.tool
Expecting property name: line 1 column 2 (char 2)
```

```
virttest.staging.backports.simplejson.dump(obj, fp, skipkeys=False, ensure_ascii=True,
                                             check_circular=True, allow_nan=True,
                                             cls=None, indent=None, separa-
                                             tors=None, encoding='utf-8', default=None,
                                             use_decimal=False, **kw)
```

Serialize `obj` as a JSON formatted stream to `fp` (a `.write()`-supporting file-like object).

If `skipkeys` is true then dict keys that are not basic types (`str`, `unicode`, `int`, `long`, `float`, `bool`, `None`) will be skipped instead of raising a `TypeError`.

If `ensure_ascii` is false, then the some chunks written to `fp` may be `unicode` instances, subject to normal Python `str` to `unicode` coercion rules. Unless `fp.write()` explicitly understands `unicode` (as in `codecs.getwriter()`) this is likely to cause an error.

If `check_circular` is false, then the circular reference check for container types will be skipped and a circular reference will result in an `OverflowError` (or worse).

If `allow_nan` is false, then it will be a `ValueError` to serialize out of range float values (`nan`, `inf`, `-inf`) in strict compliance of the JSON specification, instead of using the JavaScript equivalents (`NaN`, `Infinity`, `-Infinity`).

If `indent` is a string, then JSON array elements and object members will be pretty-printed with a newline followed by that string repeated for each level of nesting. `None` (the default) selects the most compact representation without any newlines. For backwards compatibility with versions of `simplejson` earlier than 2.1.0, an integer is also accepted and is converted to a string with that many spaces.

If `separators` is an (`item_separator`, `dict_separator`) tuple then it will be used instead of the default (`' , ' , ' : ' '`) separators. (`' , ' , ' : ' '`) is the most compact JSON representation.

`encoding` is the character encoding for `str` instances, default is UTF-8.

`default(obj)` is a function that should return a serializable version of `obj` or raise `TypeError`. The default simply raises `TypeError`.

If `use_decimal` is true (default: `False`) then `decimal.Decimal` will be natively serialized to JSON with full precision.

To use a custom `JSONEncoder` subclass (e.g. one that overrides the `.default()` method to serialize additional types), specify it with the `cls` kwarg.

```
virttest.staging.backports.simplejson.dumps(obj, skipkeys=False, ensure_ascii=True,
                                              check_circular=True, allow_nan=True,
                                              cls=None, indent=None, separa-
                                              tors=None, encoding='utf-8', default=None,
                                              use_decimal=False, **kw)
```

Serialize `obj` to a JSON formatted `str`.

If `skipkeys` is false then dict keys that are not basic types (`str`, `unicode`, `int`, `long`, `float`, `bool`, `None`) will be skipped instead of raising a `TypeError`.

If `ensure_ascii` is false, then the return value will be a `unicode` instance subject to normal Python `str` to `unicode` coercion rules instead of being escaped to an ASCII `str`.

If `check_circular` is false, then the circular reference check for container types will be skipped and a circular reference will result in an `OverflowError` (or worse).

If `allow_nan` is false, then it will be a `ValueError` to serialize out of range float values (`nan`, `inf`, `-inf`) in strict compliance of the JSON specification, instead of using the JavaScript equivalents (`NaN`, `Infinity`, `-Infinity`).

If `indent` is a string, then JSON array elements and object members will be pretty-printed with a newline followed by that string repeated for each level of nesting. `None` (the default) selects the most compact repre-

sentation without any newlines. For backwards compatibility with versions of simplejson earlier than 2.1.0, an integer is also accepted and is converted to a string with that many spaces.

If `separators` is an `(item_separator, dict_separator)` tuple then it will be used instead of the default `(' ', ' ', ' ': '')` separators. `(' ', ' ', ' ': '')` is the most compact JSON representation.

`encoding` is the character encoding for `str` instances, default is UTF-8.

`default(obj)` is a function that should return a serializable version of `obj` or raise `TypeError`. The default simply raises `TypeError`.

If `use_decimal` is true (default: `False`) then `decimal.Decimal` will be natively serialized to JSON with full precision.

To use a custom `JSONEncoder` subclass (e.g. one that overrides the `.default()` method to serialize additional types), specify it with the `cls` kwarg.

```
virttest.staging.backports.simplejson.load(fp, encoding=None, cls=None, ob-
                                         ject_hook=None, parse_float=None,
                                         parse_int=None, parse_constant=None, ob-
                                         ject_pairs_hook=None, use_decimal=False,
                                         **kw)
```

Deserialize `fp` (a `.read()`-supporting file-like object containing a JSON document) to a Python object.

`encoding` determines the encoding used to interpret any `str` objects decoded by this instance (`'utf-8'` by default). It has no effect when decoding `unicode` objects.

Note that currently only encodings that are a superset of ASCII work, strings of other encodings should be passed in as `unicode`.

`object_hook`, if specified, will be called with the result of every JSON object decoded and its return value will be used in place of the given `dict`. This can be used to provide custom deserializations (e.g. to support JSON-RPC class hinting).

`object_pairs_hook` is an optional function that will be called with the result of any object literal decode with an ordered list of pairs. The return value of `object_pairs_hook` will be used instead of the `dict`. This feature can be used to implement custom decoders that rely on the order that the key and value pairs are decoded (for example, `collections.OrderedDict()` will remember the order of insertion). If `object_hook` is also defined, the `object_pairs_hook` takes priority.

`parse_float`, if specified, will be called with the string of every JSON float to be decoded. By default, this is equivalent to `float(num_str)`. This can be used to use another datatype or parser for JSON floats (e.g. `decimal.Decimal`).

`parse_int`, if specified, will be called with the string of every JSON int to be decoded. By default, this is equivalent to `int(num_str)`. This can be used to use another datatype or parser for JSON integers (e.g. `float`).

`parse_constant`, if specified, will be called with one of the following strings: `'-Infinity'`, `'Infinity'`, `'NaN'`. This can be used to raise an exception if invalid JSON numbers are encountered.

If `use_decimal` is true (default: `False`) then it implies `parse_float=decimal.Decimal` for parity with `dump`.

To use a custom `JSONDecoder` subclass, specify it with the `cls` kwarg.

```
virttest.staging.backports.simplejson.loads(s, encoding=None, cls=None, ob-
                                         ject_hook=None, parse_float=None,
                                         parse_int=None, parse_constant=None, ob-
                                         ject_pairs_hook=None, use_decimal=False,
                                         **kw)
```

Deserialize `s` (a `str` or `unicode` instance containing a JSON document) to a Python object.

encoding determines the encoding used to interpret any `str` objects decoded by this instance (`'utf-8'` by default). It has no effect when decoding `unicode` objects.

Note that currently only encodings that are a superset of ASCII work, strings of other encodings should be passed in as `unicode`.

object_hook, if specified, will be called with the result of every JSON object decoded and its return value will be used in place of the given `dict`. This can be used to provide custom deserializations (e.g. to support JSON-RPC class hinting).

object_pairs_hook is an optional function that will be called with the result of any object literal decode with an ordered list of pairs. The return value of *object_pairs_hook* will be used instead of the `dict`. This feature can be used to implement custom decoders that rely on the order that the key and value pairs are decoded (for example, `collections.OrderedDict()` will remember the order of insertion). If *object_hook* is also defined, the *object_pairs_hook* takes priority.

parse_float, if specified, will be called with the string of every JSON float to be decoded. By default, this is equivalent to `float(num_str)`. This can be used to use another datatype or parser for JSON floats (e.g. `decimal.Decimal`).

parse_int, if specified, will be called with the string of every JSON int to be decoded. By default, this is equivalent to `int(num_str)`. This can be used to use another datatype or parser for JSON integers (e.g. `float`).

parse_constant, if specified, will be called with one of the following strings: `'-Infinity'`, `'Infinity'`, `'NaN'`. This can be used to raise an exception if invalid JSON numbers are encountered.

If *use_decimal* is true (default: `False`) then it implies *parse_float*=`decimal.Decimal` for parity with `dump`.

To use a custom `JSONDecoder` subclass, specify it with the `cls` kwarg.

```
class virttest.staging.backports.simplejson.JSONDecoder(encoding=None,          ob-
                                                         ject_hook=None,
                                                         parse_float=None,
                                                         parse_int=None,
                                                         parse_constant=None,
                                                         strict=True,          ob-
                                                         ject_pairs_hook=None)
```

Bases: `object`

Simple JSON <<http://json.org>> decoder

Performs the following translations in decoding by default:

JSON	Python
object	dict
array	list
string	unicode
number (int)	int, long
number (real)	float
true	True
false	False
null	None

It also understands `NaN`, `Infinity`, and `-Infinity` as their corresponding `float` values, which is outside the JSON spec.

decode (*s*, *_w*=<built-in method match of `_sre.SRE_Pattern` object>)

Return the Python representation of *s* (a `str` or `unicode` instance containing a JSON document)

raw_decode (*s*, *idx=0*)

Decode a JSON document from *s* (a `str` or `unicode` beginning with a JSON document) and return a 2-tuple of the Python representation and the index in *s* where the document ended.

This can be used to decode a JSON document from a string that may have extraneous data at the end.

exception `virttest.staging.backports.simplejson.JSONDecodeError` (*msg*, *doc*, *pos*, *end=None*)

Bases: `exceptions.ValueError`

Subclass of `ValueError` with the following additional properties:

msg: The unformatted error message *doc*: The JSON document being parsed *pos*: The start index of *doc* where parsing failed *end*: The end index of *doc* where parsing failed (may be `None`) *lineno*: The line corresponding to *pos* *colno*: The column corresponding to *pos* *endlineno*: The line corresponding to *end* (may be `None`) *endcolno*: The column corresponding to *end* (may be `None`)

class `virttest.staging.backports.simplejson.JSONEncoder` (*skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *encoding='utf-8'*, *default=None*, *use_decimal=False*)

Bases: `object`

Extensible JSON <<http://json.org>> encoder for Python data structures.

Supports the following objects and types by default:

Python	JSON
dict	object
list, tuple	array
str, unicode	string
int, long, float	number
True	true
False	false
None	null

To extend this to recognize other objects, subclass and implement a `.default()` method with another method that returns a serializable object for *o* if possible, otherwise it should call the superclass implementation (to raise `TypeError`).

default (*o*)

Implement this method in a subclass such that it returns a serializable object for *o*, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement `default` like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    return JSONEncoder.default(self, o)
```

encode (*o*)

Return a JSON string representation of a Python data structure.

```
>>> from simplejson import JSONEncoder
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

item_separator = ‘, ‘

iterencode (*o*, *_one_shot=False*)

Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

key_separator = ‘: ‘

class `virttest.staging.backports.simplejson.OrderedDict` (**args*, ***kwargs*)

Bases: `dict`

Dictionary that remembers insertion order

clear () → None. Remove all items from od.

copy () → a shallow copy of od

classmethod fromkeys (*S* [, *v*]) → New ordered dictionary with keys from *S*.

If not specified, the value defaults to None.

items () → list of (key, value) pairs in od

iteritems ()

od.iteritems -> an iterator over the (key, value) pairs in od

iterkeys () → an iterator over the keys in od

itervalues ()

od.itervalues -> an iterator over the values in od

keys () → list of keys in od

pop (*k* [, *d*]) → *v*, remove specified key and return the corresponding value. If key is not found, *d* is returned if given, otherwise `KeyError` is raised.

popitem () → (*k*, *v*), return and remove a (key, value) pair.

Pairs are returned in LIFO order if last is true or FIFO order if false.

setdefault (*k* [, *d*]) → od.get(*k*,*d*), also set od[*k*]=*d* if *k* not in od

update ([*E*], ***F*) → None. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a `.keys()` method, does: for *k* in *E*: *D*[*k*] = *E*[*k*] If *E* present and lacks `.keys()` method, does: for (*k*, *v*) in *E*: *D*[*k*] = *v* In either case, this is followed by: for *k*, *v* in *F*.items(): *D*[*k*] = *v*

values () → list of values in od

viewitems () → a set-like object providing a view on od's items

viewkeys () → a set-like object providing a view on od's keys

viewvalues () → an object providing a view on od's values

Module contents This module contains backported functions that are not present in Python 2.4 but are standard in more recent versions.

`virttest.staging.backports.all` (*iterable*)

From <http://stackoverflow.com/questions/3785433/python-backports-for-some-methods> :codeauthor: Tim Pietzcker <http://stackoverflow.com/users/20670/tim-pietzcker> licensed under cc-wiki with attribution required

`virttest.staging.backports.any` (*iterable*)

From <http://stackoverflow.com/questions/3785433/python-backports-for-some-methods> :codeauthor: Tim Pietzcker <http://stackoverflow.com/users/20670/tim-pietzcker> licensed under cc-wiki with attribution required

`virttest.staging.backports.bin` (*number*)

Adapted from <http://code.activestate.com/recipes/576847/> :codeauthor: Vishal Sapre :license: MIT

A foolishly simple look-up method of getting binary string from an integer This happens to be faster than all other ways!!!

`virttest.staging.backports.next` (**args*)

Retrieve the next item from the iterator by calling its next() method. If default is given, it is returned if the iterator is exhausted, otherwise StopIteration is raised. New in version 2.6.

Parameters

- **iterator** (*iterator*) – the iterator
- **default** (*object*) – the value to return if the iterator raises StopIteration

Returns The object returned by iterator.next()

Return type *object*

Submodules

virttest.staging.lv_utils module Utilities to create logical volumes or take snapshots of existing ones.

author Plamen Dimitrov

copyright Intra2net AG 2012

license GPL v2

param vg_name Name of the volume group.

param lv_name Name of the logical volume.

param lv_size Size of the logical volume as string in the form “#G” (for example 30G).

param lv_snapshot_name Name of the snapshot with origin the logical volume.

param lv_snapshot_size Size of the snapshot with origin the logical volume also as “#G”.

param ramdisk_vg_size Size of the ramdisk virtual group.

param ramdisk_basedir Base directory for the ramdisk sparse file.

param ramdisk_sparse_filename Name of the ramdisk sparse file.

Sample ramdisk params:

```
ramdisk_vg_size = "40000"
ramdisk_basedir = "/tmp"
ramdisk_sparse_filename = "virtual_hdd"
```

Sample general params:

```
vg_name='autotest_vg',
lv_name='autotest_lv',
lv_size='1G',
lv_snapshot_name='autotest_sn',
lv_snapshot_size='1G'
```

The ramdisk volume group size is in MB.

`virttest.staging.lv_utils.lv_check(vg_name, lv_name)`

Check whether provided logical volume exists.

`virttest.staging.lv_utils.vg_check(vg_name)`

Check whether provided volume group exists.

`virttest.staging.lv_utils.vg_list()`

List available volume groups.

`virttest.staging.lv_utils.vg_ramdisk_cleanup(ramdisk_filename, vg_name, loop_device, vg_ramdisk_dir)`

Inline cleanup function in case of test error.

virttest.staging.service module

class `virttest.staging.service.Factory`

Bases: `object`

Class to create different kinds of ServiceManager. The all interfaces to create manager are staticmethod, so we do not have to create an instance of factory when create manager.

•GenericServiceManager:

- Interface: `create_generic_service()`

- **Description: Object to manage the all services(ldap, sshd and so on).** You can list the all services by `GenericServiceManager.list()`. And you can operate any service by passing the service name, such as `GenericServiceManager.start("sshd")`.

Example: # Get the system service manager `service_manager = Factory.create_generic_service()`

 # Starting service/unit "sshd" `service_manager.start("sshd")`

 # Getting a list of available units `units = service_manager.list()`

•SpecificServiceManager:

- interface: `create_specific_service(service_name)`

- **description: Object to manage specific service(such as sshd).** You can not operate the other services nor list the all information on this host.

 # Get the specific service manager for sshd `sshd = Factory.create_specific_service("sshd")`
 `sshd.start()` `sshd.stop()`

After all, there is an unified interface to create both of them, `create_service(service_name=None)`.

If we pass a `service_name` to it, it will return a `SpecificServiceManager`, otherwise, it will return `GenericServiceManager`.

class `FactoryHelper` (*run=<function run>*)

Bases: `object`

Internal class to help create service manager.

Provide some functions to auto detect system type. And auto create `command_generator` and `result_parser`.

get_generic_service_command_generator()

Lazy initializer for `ServiceCommandGenerator` using the auto-detect init command.

Returns `ServiceCommandGenerator` for the current init command.

Return type `_ServiceCommandGenerator`

get_generic_service_manager_type()

Get the `ServiceManager` type using the auto-detect init command.

Returns Subclass type of `_GenericServiceManager` from the current init command.

Return type `_SysVInitServiceManager` or `_SystemdServiceManager`.

get_generic_service_result_parser()

Get the `ServiceResultParser` using the auto-detect init command.

Returns `ServiceResultParser` fro the current init command.

Return type `_ServiceResultParser`

get_name_of_init()

Internal function to determine what executable is PID 1, :return: executable name for PID 1, aka init
:rtype: str

get_specific_service_command_generator()

Create a class that will create partial functions that generate commands for the current init command.

```
lldpad = SpecificServiceManager("lldpad",
                                auto_create_specific_service_command_generator())
lldpad.start()
lldpad.stop()
```

Returns A `ServiceCommandGenerator` for the auto-detected init command.

Return type `_ServiceCommandGenerator`

get_specific_service_result_parser()

Create a class that will create partial functions that generate `result_parser` for the current init command.

Returns A `ServiceResultParser` for the auto-detected init command.

Return type `_ServiceResultParser`

static `Factory.create_generic_service(run=<function run>)`

Detect which init program is being used, init or systemd and return a class with methods to start/stop services.

```
# Get the system service manager
service_manager = Factory.create_generic_service()

# Stating service/unit "sshd"
service_manager.start("sshd")

# Getting a list of available units
units = service_manager.list()

# Disabling and stopping a list of services
services_to_disable = ['ntpd', 'httpd']
for s in services_to_disable:
    service_manager.disable(s)
    service_manager.stop(s)
```

Returns `SysVInitServiceManager` or `SystemdServiceManager`

Return type `_GenericServiceManager`

static `Factory.create_service` (*service_name=None*, *run=<function run>*)
Unified interface for generic and specific service manager.

Returns `_SpecificServiceManager` if *service_name* is not `None`, `_GenericServiceManager` if *service_name* is `None`.

static `Factory.create_specific_service` (*service_name*, *run=<function run>*)
Get the specific service manager for sshd
`sshd = Factory.create_specific_service("sshd")`
`sshd.start()`
`sshd.stop()` `sshd.reload()` `sshd.restart()` `sshd.condrestart()` `sshd.status()` `sshd.enable()` `sshd.disable()`
`sshd.is_enabled()`

Parameters *service_name* (*str*) – systemd unit or init.d service to manager

Returns `SpecificServiceManager` that has start/stop methods

Return type `_SpecificServiceManager`

`virttest.staging.service.convert_systemd_target_to_runlevel` (*target*)
Convert systemd target to runlevel.

Parameters *target* (*str*) – systemd target

Returns `sys_v` runlevel

Return type *str*

Raises `ValueError` – when systemd target is unknown

`virttest.staging.service.convert_sysv_runlevel` (*level*)
Convert runlevel to systemd target.

Parameters *level* (*str* or *int*) – `sys_v` runlevel

Returns systemd target

Return type *str*

Raises `ValueError` – when runlevel is unknown

`virttest.staging.service.raw_status_parser` (*cmdResult=None*)
Just return the result of service sub-command.

`virttest.staging.service.systemd_command_generator` (*command*)
Generate list of command line argument strings for `systemctl`. One argument per string for compatibility
Popen

WARNING: If `systemctl` detects that it is running on a tty it will use color, pipe to `$PAGER`, change column sizes and not truncate unit names. Use `--no-pager` to suppress pager output, or set `PAGER=cat` in the environment. You may need to take other steps to suppress color output. See https://bugzilla.redhat.com/show_bug.cgi?id=713567

Parameters *command* (*str*) – start, stop, restart, etc.

Returns list of command and arguments to pass to `utils.run` or similar functions

Return type *list*

`virttest.staging.service.systemd_list_parser` (*cmdResult=None*)
Parse method for service sub-command list.

:return in form of dict-like, including service name, status and so on

For example:

```
{ "sshd": "enabled",
  "vsftpd": "disabled",
  "systemd-sysctl": "static",
  ...
}
```

`virttest.staging.service.systemd_result_parser` (*command*)

Parse results for systemd style commands.

Parameters *command* – service sub-command(string).

Returns depends on sub-command.

`virttest.staging.service.systemd_status_parser` (*cmdResult=None*)

Parse method for service sub-command status.

:return True : if status is active(running). :return False : if status is stopped. :return None : if status is un-loaded.

`virttest.staging.service.sysvinit_command_generator` (*command*)

Generate lists of command arguments for sys_v style inits.

Parameters *command* (*str*) – start,stop,restart, etc.

Returns list of commands to pass to `utils.run` or similar function

Return type *list*

`virttest.staging.service.sysvinit_list_parser` (*cmdResult=None*)

Parse method for service sub-command list.

:return in form of dict-like, including service name, status and so on

For example:

```
{ "sshd": {0: 'off', 1: 'off', 2: 'off', ..., 6: 'off'},
  "vsftpd": {0: 'off', 1: 'off', 2: 'off', ..., 6: 'off'},
  "xinetd": {'discard-dgram': 'off', 'rsync': 'on', ...},
  ...
}
```

`virttest.staging.service.sysvinit_result_parser` (*command*)

Parse results for sys_v style commands.

Parameters *command* – service sub-command(string).

Returns depends on sub-command.

`virttest.staging.service.sysvinit_status_parser` (*cmdResult=None*)

Parse method for service sub-command status.

:return True : if status is running or active. :return False : if status is stopped. :return None : if status is unrecognized.

virttest.staging.utils_cgroup module Helpers for cgroup testing.

copyright 2011 Red Hat Inc.

author Lukas Doktor <ldoktor@redhat.com>

class `virttest.staging.utils_cgroup.CgconfigService`

Bases: `object`

Cgconfig service class.

cgconfig_condrestart ()
Condrestart cgconfig service

cgconfig_is_running ()
Check cgconfig service status

cgconfig_restart ()
Restart cgconfig service

cgconfig_start ()
Start cgconfig service

cgconfig_stop ()
Stop cgconfig service

class `virttest.staging.utils_cgroup.Cgroup` (*module*, *_client*)
Bases: `object`

Cgroup handling class.

cgclassify_cgroup (*pid*, *cgroup*)
Classify pid into cgroup

Parameters

- **pid** – pid of the process
- **cgroup** – cgroup name

cgdelete_all_cgroups ()
Delete all cgroups in the module

cgdelete_cgroup (*cgroup*, *recursive=False*)
Delete desired cgroup.

Params cgroup desired cgroup

Params force If true, sub cgroup can be deleted with parent cgroup

cgexec (*cgroup*, *cmd*, *args=''*)
Execute command in desired cgroup

Parameters

- **cgroup** – Desired cgroup
- **cmd** – Executed command
- **args** – Executed command's parameters

cgset_property (*prop*, *value*, *pwd=None*, *check=True*, *checkprop=None*)
Sets the property value by cgset command

Parameters

- **prop** – property name (file)
- **value** – desired value
- **pwd** – cgroup directory
- **check** – check the value after setup / override checking value
- **checkprop** – override prop when checking the value

get_all_cgroups ()
Get all sub cgroups in this controller

get_cgroup_index (*cgroup*)

Get cgroup's index in cgroups

Parameters *cgroup* – cgroup name

Returns index of cgroup

get_cgroup_name (*pwd=None*)

Get cgroup's name

Parameters *pwd* – cgroup name

Returns cgroup's name

get_pids (*pwd=None*)

Get all pids in cgroup

Params *pwd*: cgroup directory

Returns all pids(list)

get_property (*prop, pwd=None*)

Gets the property value :param prop: property name (file) :param pwd: cgroup directory :return: [] values or None when FAILED

initialize (*modules*)

Initializes object for use.

Parameters *modules* – Array of all available cgroup modules.

is_cgroup (*pid, pwd*)

Checks if the 'pid' process is in 'pwd' cgroup :param pid: pid of the process :param pwd: cgroup directory :return: 0 when is 'pwd' member

is_root_cgroup (*pid*)

Checks if the 'pid' process is in root cgroup (WO cgroup) :param pid: pid of the process :return: 0 when is 'root' member

mk_cgroup (*pwd=None, cgroup=None*)

Creates new temporary cgroup :param pwd: where to create this cgroup (default: self.root) :param cgroup: desired cgroup name :return: last cgroup index

mk_cgroup_cgcreate (*pwd=None, cgroup=None*)

Make a cgroup by executing the cgcreate command

Params *cgroup*: name of the cgroup to be created

Returns last cgroup index

rm_cgroup (*pwd*)

Removes cgroup.

Parameters *pwd* – cgroup directory.

set_cgroup (*pid, pwd=None*)

Sets cgroup membership :param pid: pid of the process :param pwd: cgroup directory

set_property (*prop, value, pwd=None, check=True, checkprop=None*)

Sets the property value :param prop: property name (file) :param value: desired value :param pwd: cgroup directory :param check: check the value after setup / override checking value :param checkprop: override prop when checking the value

set_property_h (*prop, value, pwd=None, check=True, checkprop=None*)

Sets the one-line property value concerning the K,M,G postfix :param prop: property name (file) :param

value: desired value :param pwd: cgroup directory :param check: check the value after setup / override checking value :param checkprop: override prop when checking the value

set_root_cgroup (*pid*)

Resets the cgroup membership (sets to root) :param pid: pid of the process :return: 0 when PASSED

smoke_test ()

Smoke test Module independent basic tests

test (*cmd*)

Executes cgroup_client.py with cmd parameter.

Parameters **cmd** – command to be executed

Returns subprocess.Popen() process

class virttest.staging.utils_cgroup.**CgroupModules** (*mountdir=None*)

Bases: `object`

Handles the list of different cgroup filesystems.

get_pwd (*module*)

Returns the mount directory of 'module' :param module: desired module (memory, ...) :return: mount directory of 'module' or None

init (*_modules*)

Checks the mounted modules and if necessary mounts them into tmp mountdir.

Parameters **_modules** – Desired modules.'memory','cpu,cpuset'...

Returns Number of initialized modules.

virttest.staging.utils_cgroup.**all_cgroup_delete** ()

Clear all cgroups in system

virttest.staging.utils_cgroup.**get_all_controllers** ()

Get all controllers used in system

Returns all used controllers(controller_list)

virttest.staging.utils_cgroup.**get_cgroup_mountpoint** (*controller,*
mount_file='/proc/mounts')

Get desired controller's mountpoint

Parameters

- **controller** – Desired controller
- **mount_file** – Name of file contains mounting information, in most cases this are not need to be set.

Returns controller's mountpoint

Raise TestError when contoller doesn't exist in mount table

virttest.staging.utils_cgroup.**get_load_per_cpu** (*_stats=None*)

Gather load per cpu from /proc/stat :param _stats: previous values :return: list of diff/absolute values of CPU times [SUM, CPU1, CPU2, ...]

virttest.staging.utils_cgroup.**resolve_task_cgroup_path** (*pid, controller*)

Resolving cgroup mount path of a particular task

Params **pid** : process id of a task for which the cgroup path required

Params **controller**: takes one of the controller names in controller list

Returns resolved path for cgroup controllers of a given pid

virttest.staging.utils_koji module**class** `virttest.staging.utils_koji.KojiClient` (*cmd=None*)Bases: `object`

Establishes a connection with the build system, either koji or brew.

This class provides convenience methods to retrieve information on packages and the packages themselves hosted on the build system. Packages should be specified in the KojiPkgSpec syntax.

CMD_LOOKUP_ORDER = ['/usr/bin/brew', '/usr/bin/koji']**CONFIG_MAP** = {'/usr/bin/brew': '/etc/brewkoji.conf', '/usr/bin/koji': '/etc/koji.conf'}**RETRY_STEP** = 3**RETRY_TIMEOUT** = 30**get_default_command**()

Looks up for koji or brew “binaries” on the system

Systems with plain koji usually don't have a brew cmd, while systems with koji, have *both* koji and brew utilities. So we look for brew first, and if found, we consider that the system is configured for brew. If not, we consider this is a system with plain koji.

Returns either koji or brew command line executable path, or None

get_pkg_base_url()

Gets the base url for packages in Koji

get_pkg_info(*pkg*)

Returns information from Koji on the package

Parameters **pkg** (`KojiPkgSpec`) – information about the package, as a KojiPkgSpec instance

Returns information from Koji about the specified package

get_pkg_rpm_file_names(*pkg, arch=None*)

Gets the file names for the RPM packages specified in pkg

Parameters

- **pkg** (`KojiPkgSpec`) – a package specification
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_pkg_rpm_info(*pkg, arch=None*)

Returns a list of information on the RPM packages found on koji

Parameters

- **pkg** (`KojiPkgSpec`) – a package specification
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_pkg_rpm_names(*pkg, arch=None*)

Gets the names for the RPM packages specified in pkg

Parameters

- **pkg** (`KojiPkgSpec`) – a package specification
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_pkg_urls (*pkg*, *arch=None*)

Gets the urls for the packages specified in *pkg*

Parameters

- **pkg** (*KojiPkgSpec*) – a package specification
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_pkgs (*pkg*, *dst_dir*, *arch=None*)

Download the packages

Parameters

- **pkg** (*KojiPkgSpec*) – a package specification
- **dst_dir** (*string*) – the destination directory, where the downloaded packages will be saved on
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_scratch_base_url ()

Gets the base url for scratch builds in Koji

get_scratch_pkg_urls (*pkg*, *arch=None*)

Gets the urls for the scratch packages specified in *pkg*

Parameters

- **pkg** (*KojiScratchPkgSpec*) – a scratch package specification
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_scratch_pkgs (*pkg*, *dst_dir*, *arch=None*)

Download the packages from a scratch build

Parameters

- **pkg** (*KojiScratchPkgSpec*) – a scratch package specification
- **dst_dir** (*string*) – the destination directory, where the downloaded packages will be saved on
- **arch** (*string*) – packages built for this architecture, but also including architecture independent (noarch) packages

get_session_options ()

Filter only options necessary for setting up a cobbler client session

Returns only the options used for session setup

is_command_valid ()

Checks if the currently set koji command is valid

Returns True or False

is_config_valid ()

Checks if the currently set koji configuration is valid

Returns True or False

is_pkg_spec_build_valid (*pkg*)

Checks if build is valid on Koji

Parameters `pkg` – a `Pkg` instance

is_pkg_spec_tag_valid (`pkg`)
Checks if tag is valid on Koji

Parameters `pkg` (`KojiPkgSpec`) – a package specification

is_pkg_valid (`pkg`)
Checks if this package is altogether valid on Koji

This verifies if the build or tag specified in the package specification actually exist on the Koji server

Returns True or False

read_config (`check_is_valid=True`)
Reads options from the Koji configuration file

By default it checks if the koji configuration is valid

Parameters `check_valid` (`boolean`) – whether to include a check on the configuration

Raise `ValueError`

Returns None

class `virttest.staging.utils_koji.KojiDirIndexParser`

Bases: `HTMLParser.HTMLParser`

Parser for HTML directory index pages, specialized to look for RPM links

handle_starttag (`tag`, `attrs`)
Handle tags during the parsing

This just looks for links ('a' tags) for files ending in .rpm

exception `virttest.staging.utils_koji.KojiDownloadError` (`url`, `timeout`, `last_error`)
Bases: `exceptions IOError`

class `virttest.staging.utils_koji.KojiPkgSpec` (`text=''`, `tag=None`, `build=None`, `package=None`, `subpackages=[]`)

Bases: `object`

A package specification syntax parser for Koji

This holds information on either tag or build, and packages to be fetched from koji and possibly installed (features external do this class).

New objects can be created either by providing information in the textual format or by using the actual parameters for tag, build, package and sub-packages. The textual format is useful for command line interfaces and configuration files, while using parameters is better for using this in a programmatic fashion.

The following sets of examples are interchangeable. Specifying all packages part of build number 1000:

```
>>> from kvm_utils import KojiPkgSpec
>>> pkg = KojiPkgSpec('1000')
```

```
>>> pkg = KojiPkgSpec(build=1000)
```

Specifying only a subset of packages of build number 1000:

```
>>> pkg = KojiPkgSpec('1000:kernel, kernel-devel')
```

```
>>> pkg = KojiPkgSpec(build=1000,
                      subpackages=['kernel', 'kernel-devel'])
```

Specifying the latest build for the 'kernel' package tagged with 'dist-f14':

```
>>> pkg = KojiPkgSpec('dist-f14:kernel')
```

```
>>> pkg = KojiPkgSpec(tag='dist-f14', package='kernel')
```

Specifying the ‘kernel’ package using the default tag:

```
>>> kvm_utils.set_default_koji_tag('dist-f14')
>>> pkg = KojiPkgSpec('kernel')
```

```
>>> pkg = KojiPkgSpec(package='kernel')
```

Specifying the ‘kernel’ package using the default tag:

```
>>> kvm_utils.set_default_koji_tag('dist-f14')
>>> pkg = KojiPkgSpec('kernel')
```

```
>>> pkg = KojiPkgSpec(package='kernel')
```

If you do not specify a default tag, and give a package name without an explicit tag, your package specification is considered invalid:

```
>>> print kvm_utils.get_default_koji_tag()
None
>>> print kvm_utils.KojiPkgSpec('kernel').is_valid()
False
```

```
>>> print kvm_utils.KojiPkgSpec(package='kernel').is_valid()
False
```

SEP = ‘:’

describe()

Describe this package specification, in a human friendly way

Returns package specification description

describe_invalid()

Describes why this is not valid, in a human friendly way

is_valid()

Checks if this package specification is valid.

Being valid means that it has enough and not conflicting information. It does not validate that the packages specified actually exist on the Koji server.

Returns True or False

parse(text)

Parses a textual representation of a package specification

Parameters **text** (*string*) – textual representation of a package in koji

to_text()

Return the textual representation of this package spec

The output should be consumable by parse() and produce the same package specification.

We find that it’s acceptable to put the currently set default tag as the package explicit tag in the textual definition for completeness.

Returns package specification in a textual representation

```
class virttest.staging.utils_koji.KojiScratchPkgSpec (text='', user=None, task=None,
                                                    subpackages=[])
```

Bases: `object`

A package specification syntax parser for Koji scratch builds

This holds information on user, task and subpackages to be fetched from koji and possibly installed (features external do this class).

New objects can be created either by providing information in the textual format or by using the actual parameters for user, task and subpackages. The textual format is useful for command line interfaces and configuration files, while using parameters is better for using this in a programmatic fashion.

This package definition has a special behaviour: if no subpackages are specified, all packages of the chosen architecture (plus noarch packages) will match.

The following sets of examples are interchangeable. Specifying all packages from a scratch build (whose task id is 1000) sent by user jdoe:

```
>>> from kvm_utils import KojiScratchPkgSpec
>>> pkg = KojiScratchPkgSpec('jdoe:1000')
```

```
>>> pkg = KojiScratchPkgSpec(user=jdoe, task=1000)
```

Specifying some packages from a scratch build whose task id is 1000, sent by user jdoe:

```
>>> pkg = KojiScratchPkgSpec('jdoe:1000:kernel, kernel-devel')
```

```
>>> pkg = KojiScratchPkgSpec(user=jdoe, task=1000,
                              subpackages=['kernel', 'kernel-devel'])
```

SEP = ':'

parse (*text*)

Parses a textual representation of a package specification

Parameters *text* (*string*) – textual representation of a package in koji

```
class virttest.staging.utils_koji.RPMFileNameInfo (filename)
```

Simple parser for RPM based on information present on the filename itself

get_arch ()

Returns just the architecture as present on the RPM filename

get_filename_without_arch ()

Returns the filename without the architecture

This also excludes the RPM suffix, that is, removes the leading arch and RPM suffix.

get_filename_without_suffix ()

Returns the filename without the default RPM suffix

get_nvr_info ()

Returns a dictionary with the name, version and release components

If koji is not installed, this returns None

```
virttest.staging.utils_koji.get_default_koji_tag()
```

```
virttest.staging.utils_koji.set_default_koji_tag(tag)
```

Sets the default tag that will be used

virttest.staging.utils_memory module

`virttest.staging.utils_memory.drop_caches()`

Writes back all dirty pages to disk and clears all the caches.

`virttest.staging.utils_memory.freememtotal()`

`virttest.staging.utils_memory.get_buddy_info(chunk_sizes, nodes='all', zones='all')`

Get the fragmentation status of the host. It use the same method to get the page size in buddyinfo. $2^{\text{chunk_size}} * \text{page_size}$ The chunk_sizes can be string make up by all orders that you want to check splited with blank or a mathematical expression with '>', '<' or '='. For example: The input of chunk_size could be: "0 2 4" And the return will be: {'0': 3, '2': 286, '4': 687} if you are using expression: ">=9" the return will be: {'9': 63, '10': 225}

Parameters

- **chunk_size** (*string*) – The order number shows in buddyinfo. This is not the real page size.
- **nodes** (*string*) – The numa node that you want to check. Default value is all
- **zones** (*string*) – The memory zone that you want to check. Default value is all

Returns A dict using the chunk_size as the keys

Return type `dict`

`virttest.staging.utils_memory.get_huge_page_size()`

`virttest.staging.utils_memory.get_num_anon_huge_pages(pid=0)`

`virttest.staging.utils_memory.get_num_huge_pages()`

`virttest.staging.utils_memory.get_num_huge_pages_free()`

`virttest.staging.utils_memory.get_num_huge_pages_rsvd()`

`virttest.staging.utils_memory.get_transparent_hugepage()`

`virttest.staging.utils_memory.memtotal()`

`virttest.staging.utils_memory.node_size()`

`virttest.staging.utils_memory.numa_nodes()`

`virttest.staging.utils_memory.read_from_meminfo(key)`

`virttest.staging.utils_memory.read_from_numa_maps(pid, key)`

Get the process numa related info from numa_maps. This function only use to get the numbers like anon=1.

Parameters

- **pid** (*String*) – Process id
- **key** (*String*) – The item you want to check from numa_maps

Returns A dict using the address as the keys

Return type `dict`

`virttest.staging.utils_memory.read_from_numastat(pid, key)`

Get the process numastat from numastat output.

`virttest.staging.utils_memory.read_from_smaps(pid, key)`

Get specific item value from the smaps of a process include all sections.

Parameters

- **pid** (*String*) – Process id

- **key** (*String*) – The item you want to check from smaps

Returns The value of the item in kb

Return type `int`

```
virttest.staging.utils_memory.read_from_vmstat(key)
```

Get specific item value from vmstat

Parameters **key** (*String*) – The item you want to check from vmstat

Returns The value of the item

Return type `int`

```
virttest.staging.utils_memory.rounded_memtotal()
```

```
virttest.staging.utils_memory.set_num_huge_pages(num)
```

```
virttest.staging.utils_memory.set_transparent_hugepage(sflag)
```

sflag only can be set always, madvise or never.

Module contents

virttest.tests package

Submodules

virttest.tests.unattended_install module

```
class virttest.tests.unattended_install.RemoteInstall(path, ip, port, filename)
```

Bases: `object`

Represents a install http server that we can master according to our needs.

```
close()
```

```
get_answer_file_path(filename)
```

```
get_url()
```

```
class virttest.tests.unattended_install.UnattendedInstallConfig(test, params, vm)
```

Bases: `object`

Creates a floppy disk image that will contain a config file for unattended OS install. The parameters to the script are retrieved from environment variables.

```
answer_kickstart(answer_path)
```

Replace KVM_TEST_CDKEY (in the unattended file) with the cdkey provided for this test and replace the KVM_TEST_MEDIUM with the tree url or nfs address provided for this test.

Returns Answer file contents

```
answer_suse_xml(answer_path)
```

```
answer_windows_ini(answer_path)
```

```
answer_windows_xml(answer_path)
```

```
get_driver_hardware_id(*args, **kwargs)
```

Get windows driver's hardware id from inf files.

Parameters

- **dirver** – Configurable driver name.

- **run_cmd** – Use hardware id in windows cmd command or not.

Returns Windows driver's hardware id

preseed_initrd()

Puts a preseed file inside a gz compressed initrd file.

Debian and Ubuntu use preseed as the OEM install mechanism. The only way to get fully automated setup without resorting to kernel params is to add a preseed.cfg file at the root of the initrd image.

setup()

Configure the environment for unattended install.

Uses an appropriate strategy according to each install model.

setup_boot_disk()

setup_cdrom(*args, **kwargs)

Mount cdrom and copy vmlinuz and initrd.img.

setup_import()

setup_nfs()

Copy the vmlinuz and initrd.img from nfs.

setup_unattended_http_server()

Setup a builtin http server for serving the kickstart file

Does nothing if unattended file is not a kickstart file

setup_url(*args, **kwargs)

Download the vmlinuz and initrd.img from URL.

setup_url_auto(*args, **kwargs)

Configures the builtin web server for serving content

update_driver_hardware_id(*args, **kwargs)

Update driver string with the hardware id get from inf files

@driver: driver string :return: new driver string

```
virttest.tests.unattended_install.copy_file_from_nfs(src, dst, mount_point, im-  
age_name)
```

```
virttest.tests.unattended_install.start_auto_content_server_thread(port,  
path)
```

```
virttest.tests.unattended_install.start_syslog_server_thread(address, port, tcp)
```

```
virttest.tests.unattended_install.start_unattended_server_thread(port, path)
```

```
virttest.tests.unattended_install.terminate_auto_content_server_thread()
```

```
virttest.tests.unattended_install.terminate_syslog_server_thread()
```

```
virttest.tests.unattended_install.terminate_unattended_server_thread()
```

Module contents

virttest.utils_test package

Submodules

virttest.utils_test.libguestfs module**class** virttest.utils_test.libguestfs.**GuestfishTools** (*params*)Bases: *virttest.utils_test.libguestfs.GuestfishPersistent*

Useful Tools for Guestfish class.

analyse_release ()

Analyse /etc/redhat-release

copy_ifcfg_back ()**create_fs** ()

Create filesystem of disk

Choose lvm or physical partition and create fs on it

create_msdos_part (*device*, *start*='1', *end*='-1')

Create a msdos partition in given device. Default partition section is whole disk(1~1). And return its part name if part add succeed.

create_whole_disk_msdos_part (*device*)

Create only one msdos partition in given device. And return its part name if part add succeed.

get_bootable_part (*device*='/dev/sda')**get_mbr_id** (*device*='/dev/sda')**get_md5** (*path*)

Get files md5 value.

get_part_size (*part_num*)**get_part_type** (*device*='/dev/sda')**get_partitions_info** (*device*='/dev/sda')

Get disk partition's information.

get_root ()

Get root filesystem w/ guestfish

params**reset_interface** (*iface_mac*)

Check interface through guestfish. Fix mac if necessary.

write_file (*path*, *content*)

Create a new file to vm with guestfish

exception virttest.utils_test.libguestfs.**VTAttachError** (*cmd*, *output*)Bases: *virttest.utils_test.libguestfs.VTError***exception** virttest.utils_test.libguestfs.**VTError**Bases: *exceptions.Exception***exception** virttest.utils_test.libguestfs.**VTMountError** (*cmd*, *output*)Bases: *virttest.utils_test.libguestfs.VTError***exception** virttest.utils_test.libguestfs.**VTXMLParseError** (*cmd*, *output*)Bases: *virttest.utils_test.libguestfs.VTError***class** virttest.utils_test.libguestfs.**VirtTools** (*vm*, *params*)Bases: *object*

Useful functions for virt-commands.

Some virt-tools need an input disk and output disk. Main for virt-clone, virt-sparsify, virt-resize.

cat (*filename*, *vm_ref=None*)

clone_vm_filesystem (*newname=None*)

Clone a new vm with only its filesystem disk.

:param newname:if newname is None, create a new name with clone added.

copy_in (*filename*, *dest='/tmp'*, *vm_ref=None*)

copy_out (*file_path*, *localdir='/tmp'*, *vm_ref=None*)

define_vm_with_newdisk ()

Define the new vm with old vm's configuration

Changes: 1.replace name 2.delete uuid 3.replace disk

expand_vm_filesystem (*resize_part_num=2*, *resized_size='+1G'*, *new_disk=None*)

Expand vm's filesize with virt-resize.

format_disk (*disk_path=None*, *filesystem=None*, *partition=None*, *lvm=None*)

Parameters disk_path – None for additional disk by update_vm_disk() only

get_filesystems_info (*vm_ref=None*)

get_primary_disk_fs_type ()

Get primary disk filesystem type

get_vm_info_with_inspector (*vm_ref=None*)

Return a dict includes os information.

guestmount (*mountpoint*, *disk_or_domain=None*)

Mount filesystems in a disk or domain to host mountpoint.

Parameters disk_or_domain – if it is None, use default vm in params

list_df (*vm_ref=None*)

sparsify_disk ()

Sparsify a disk

tar_in (*tar_file*, *dest='/tmp'*, *vm_ref=None*)

tar_out (*directory*, *tar_file='temp.tar'*, *vm_ref=None*)

update_vm_disk ()

Update oldvm's disk, and then create a newvm.

write_file_with_guestmount (*mountpoint*, *path*, *content=None*, *vm_ref=None*, *cleanup=True*)

Write content to file with guestmount

`virttest.utils_test.libguestfs.attach_additional_disk` (*vm*, *disksize*, *targetdev*)

Create a disk with disksize, then attach it to given vm.

Parameters

- **vm** – Libvirt VM object.
- **disksize** – size of attached disk
- **targetdev** – target of disk device

`virttest.utils_test.libguestfs.cleanup_vm` (*vm_name=None*, *disk=None*)

Cleanup the vm with its disk deleted.

`virttest.utils_test.libguestfs.define_new_vm` (*vm_name*, *new_name*)

Just define a new vm from given name

`virttest.utils_test.libguestfs.get_primary_disk(vm)`
Get primary disk source.

Parameters `vm` – Libvirt VM object.

`virttest.utils_test.libguestfs.preprocess_image(params)`
Create a disk which used by guestfish

`params`: Get params from cfg file

`virttest.utils_test.libguestfs.primary_disk_virtio(vm)`
To verify if system disk is virtio.

Parameters `vm` – Libvirt VM object.

virttest.utils_test.libvirt module High-level libvirt test utility functions.

This module is meant to reduce code size by performing common test procedures. Generally, code here should look like test code.

More specifically:

- Functions in this module should raise exceptions if things go wrong
- Functions in this module typically use functions and classes from lower-level modules (e.g. `utils_misc`, `qemu_vm`, `expect`).
- Functions in this module should not be used by lower-level modules.
- Functions in this module should be used in the right context. For example, a function should not be used where it may display misleading or inaccurate info or debug messages.

copyright 2014 Red Hat Inc.

class `virttest.utils_test.libvirt.LibvirtNetwork` (*net_type, address=None, iface=None, net_name=None, persistent=False*)

Bases: `object`

Class to create a temporary network for testing.

cleanup()
Clear up network.

create_bridge_xml()
Create XML for a bridged network.

create_macvtap_xml()
Create XML for a macvtap network.

create_vnet_xml()
Create XML for a virtual network.

class `virttest.utils_test.libvirt.MigrationTest`

Bases: `object`

Class for migration tests

cleanup_dest_vm (*vm, srcuri, desturi*)
Cleanup migrated vm on remote host.

do_migration (*vms, srcuri, desturi, migration_type, options=None, thread_timeout=60*)
Migrate vms.

Parameters

- **vms** – migrated vms.
- **srcuri** – local uri, used when migrate vm from remote to local
- **descuri** – remote uri, used when migrate vm from local to remote
- **migration_type** – do orderly for simultaneous migration

thread_func_migration (*vm, desturi, options=None*)

Thread for virsh migrate command.

Parameters

- **vm** – A libvirt vm instance(local or remote).
- **desturi** – remote host uri.

class `virttest.utils_test.libvirt.PoolVolumeTest` (*test, params*)

Bases: `object`

Test class for storage pool or volume

cleanup_pool (*pool_name, pool_type, pool_target, emulated_image, **kwargs*)

Delete vols, destroy the created pool and restore the env

pre_pool (*pool_name, pool_type, pool_target, emulated_image, **kwargs*)

Prepare(define or create) the specific type pool

Parameters

- **pool_name** – created pool name
- **pool_type** – dir, disk, logical, fs, netfs or else
- **pool_target** – target of storage pool
- **emulated_image** – use an image file to simulate a scsi disk it could be used for disk, logical pool, etc
- **kwargs** – key words for specific pool

pre_vol (*vol_name, vol_format, capacity, allocation, pool_name*)

Preprepare the specific type volume in pool

`virttest.utils_test.libvirt.alter_boot_order` (*vm_name, pci_id, boot_order=0*)

Alter the startup sequence of VM to PCI-device firstly

OS boot element and per-device boot elements are mutually exclusive, It's necessary that remove all OS boots before setting PCI-device order

Parameters

- **vm_name** – VM name
- **pci_id** – such as “0000:06:00.1”
- **boot_order** – order priority, such as 1, 2, ...

`virttest.utils_test.libvirt.attach_additional_device` (*vm_name, targetdev, disk_path, params, config=True*)

Create a disk with disksize, then attach it to given vm.

Parameters

- **vm_name** – Libvirt VM name.
- **disk_path** – path of attached disk
- **targetdev** – target of disk device

- **params** – dict include necessary configurations of device

`virttest.utils_test.libvirt.attach_disks(vm, path, vgname, params)`

Attach multiple disks. According parameter `disk_type` in `params`, it will create lvm or file type disks.

Parameters

- **path** – file type disk's path
- **vgname** – lvm type disk's volume group name

`virttest.utils_test.libvirt.check_activated_pool(pool_name)`

Check if `pool_name` exist in active pool list

`virttest.utils_test.libvirt.check_blockjob(vm_name, target, check_point='none', value='0')`

Run blockjob command to check block job progress, bandwidth, ect.

Parameters

- **vm_name** – Domain name
- **target** – Domain disk target dev
- **check_point** – Job progress, bandwidth or none(no job)
- **value** – Value of progress, bandwidth or 0(no job)

Returns Boolean value, true for pass, false for fail

`virttest.utils_test.libvirt.check_exit_status(result, expect_error=False)`

Check the exit status of virsh commands.

Parameters

- **result** – Virsh command result object
- **expect_error** – Boolean value, expect command success or fail

`virttest.utils_test.libvirt.check_iface(iface_name, checkpoint, extra='', **dargs)`

Check interface with specified checkpoint.

Parameters

- **iface_name** – Interface name
- **checkpoint** – Check if interface exists, and It's MAC address, IP address and State, also connectivity by ping. valid checkpoint: [exists, mac, ip, ping, state]
- **extra** – Extra string for checking

Returns Boolean value, true for pass, false for fail

`virttest.utils_test.libvirt.check_result(result, expected_fails=[], skip_if=[], any_error=False)`

Check the result of a command and check command error message against expectation.

Parameters

- **result** – Command result instance.
- **expected_fails** – list of regex of expected stderr patterns. The check will pass if any of these patterns matches.
- **skip_if** – list of regex of expected patterns. The check will raise a `TestNAError` if any of these patterns matches.

- **any_error** – Whether expect on any error message. Setting to True will will override expected_fails

`virttest.utils_test.libvirt.clean_up_snapshots (vm_name, snapshot_list=[])`

Do recovery after snapshot

Parameters

- **vm_name** – Name of domain
- **snapshot_list** – The list of snapshot name you want to remove

`virttest.utils_test.libvirt.connect_libvirtd (uri, read_only='', virsh_cmd='list',
auth_user=None, auth_pwd=None,
vm_name='', status_error='no', extra='', log_level='LIBVIRT_DEBUG=3',
su_user='', pattern=*,
terms_virsh_cmd='.*Id\\s*Name\\s*State\\s*.*')`

Connect libvirt daemon

`virttest.utils_test.libvirt.cpu_allowed_list_by_task (pid, tid)`

Get the Cpus_allowed_list in status of task.

`virttest.utils_test.libvirt.cpus_parser (cpulist)`

Parse a list of cpu list, its syntax is a comma separated list, with '-' for ranges and '^' denotes exclusive. :param cpulist: a list of physical CPU numbers

`virttest.utils_test.libvirt.cpus_string_to_affinity_list (cpus_string, num_cpus)`

Parse the cpus_string string to a affinity list.

e.g host_cpu_count = 4 0 -> [y,-,-] 0,1 -> [y,y,-] 0-2 -> [y,y,y,-] 0-2,^2 -> [y,y,-] r -> [y,y,y,y]

`virttest.utils_test.libvirt.create_channel_xml (params, alias=False, address=False)`

Create a XML contains channel information.

Parameters

- **params** – the params for Channel slot
- **alias** – allow to add 'alias' slot
- **address** – allow to add 'address' slot

`virttest.utils_test.libvirt.create_disk_xml (params)`

Create a disk configuration file.

`virttest.utils_test.libvirt.create_hostdev_xml (pci_id, boot_order=0)`

Create a hostdev configuration file.

Parameters **pci_id** – such as "0000:03:04.0"

`virttest.utils_test.libvirt.create_local_disk (disk_type, path=None, size='10',
disk_format='raw', vname=None,
lvname=None)`

`virttest.utils_test.libvirt.create_net_xml (net_name, params)`

Create a new network or update an existed network xml

`virttest.utils_test.libvirt.create_nwfilter_xml (params)`

Create a new network filter or update an existed network filter xml

`virttest.utils_test.libvirt.create_scsi_disk (scsi_option, scsi_size='2048')`

Get the scsi device created by scsi_debug kernel module

:param scsi_option. The scsi_debug kernel module options. :return: scsi device if it is created successfully.

```
virttest.utils_test.libvirt.define_new_vm(vm_name, new_name)
```

Just define a new vm from given name

```
virttest.utils_test.libvirt.define_pool(pool_name, pool_type, pool_target, cleanup_flag,  
                                          **kwargs)
```

To define a given type pool(Support types: 'dir', 'netfs', 'logical', 'iscsi', 'gluster', 'disk' and 'fs').

Parameters

- **pool_name** – Name of the pool
- **pool_type** – Type of the pool
- **pool_target** – Target for underlying storage
- **cleanup_flag** – A list contains 3 booleans and 1 string stands for need_cleanup_nfs, need_cleanup_iscsi, need_cleanup_logical, selinux_bak and need_cleanup_gluster
- **kwargs** – key words for special pool define. eg, glusterfs pool source path and source name, etc

```
virttest.utils_test.libvirt.delete_local_disk(disk_type, path=None, vname=None, lv-  
                                              name=None)
```

```
virttest.utils_test.libvirt.delete_scsi_disk()
```

Delete scsi device by removing scsi_debug kernel module.

```
virttest.utils_test.libvirt.device_exists(vm, target_dev)
```

Check if given target device exists on vm.

```
virttest.utils_test.libvirt.do_migration(vm_name, uri, extra, auth_pwd,  
                                          auth_user='root', options='-verbose',  
                                          virsh_patterns='.*100\\s%.*', su_user='',  
                                          timeout=30)
```

Migrate VM to target host.

```
virttest.utils_test.libvirt.exec_virsh_edit(source, edit_cmd, connect-  
                                             uri='qemu:///system')
```

Execute edit command.

:param source : virsh edit's option. :param edit_cmd: Edit command list to execute. :return: True if edit is successful, False if edit is failure.

```
virttest.utils_test.libvirt.get_all_cells()
```

Use virsh freecell --all to get all cells on host

```
# virsh freecell --all
0:      124200 KiB
1:      1059868 KiB
-----
Total:    1184068 KiB
```

That would return a dict like:

```
cell_dict = {"0": "124200 KiB", "1": "1059868 KiB", "Total": "1184068 KiB"}
```

Returns cell_dict

```
virttest.utils_test.libvirt.get_all_vol_paths()
```

Get all volumes' path in host

```
virttest.utils_test.libvirt.get_host_ipv4_addr()
```

Get host ipv4 addr

`virttest.utils_test.libvirt.get_ifname_host (vm_name, mac)`

Get the vm interface name on host

Returns interface name, None if not exist

`virttest.utils_test.libvirt.get_interface_details (vm_name)`

Get the interface details from virsh domiflist command output

Returns list of all interfaces details

`virttest.utils_test.libvirt.get_parts_list (session=None)`

Get all partition lists.

`virttest.utils_test.libvirt.hotplug_domain_vcpu (domain, count, by_virsh=True, hotplug=True)`

Hot-plug/Hot-unplug vcpu for domian

Parameters

- **domain** – Domain name, id, uuid
- **count** – to setvcpus it's the current vcpus number, but to qemu-monitor-command, we need to designate a specific CPU ID. The default will be got by (count - 1)
- **by_virsh** – True means hotplug/unplug by command setvcpus, otherwise, using qemu_monitor
- **hotplug** – True means hot-plug, False means hot-unplug

`virttest.utils_test.libvirt.mk_label (disk, label='msdos', session=None)`

Set label for disk.

`virttest.utils_test.libvirt.mk_part (disk, size='100M', session=None)`

Create a partition for disk

`virttest.utils_test.libvirt.mkfs (partition, fs_type, options='', session=None)`

Make a file system on the partition

`virttest.utils_test.libvirt.new_disk_vol_name (pool_name)`

According to BZ#1138523, the new volume name must be the next created partition(sdb1, etc.), so we need to inspect the original partitions of the disk then count the new partition number.

Parameters **pool_name** – Disk pool name

Returns New volume name or none

`virttest.utils_test.libvirt.pci_label_from_address (address_dict, radix=10)`

Generate a pci label from a dict of address.

Parameters

- **address_dict** – A dict contains domain, bus, slot and function.
- **radix** – The radix of your data in address_dict.

Example:

```
address_dict = {'domain': '0x0000', 'bus': '0x08', 'slot': '0x10', 'function': '0x0'}
radix = 16
return = pci_0000_08_10_0
```

`virttest.utils_test.libvirt.remotely_control_libvirtd (server_ip, server_user, server_pwd, action='restart', status_error='no')`

Remotely restart libvirt service

```
virttest.utils_test.libvirt.set_controller_multifunction (vm_name,          con-
                                                         troller_type='scsi')
```

Set multifunction on for controller device and expand to all function.

```
virttest.utils_test.libvirt.set_domain_state (vm, vm_state)
```

Set domain state.

Parameters

- **vm** – the vm object
- **vm_state** – the given vm state string “shut off”, “running” “paused”, “halt” or “pm_suspend”

```
virttest.utils_test.libvirt.set_guest_agent (vm)
```

Set domain xml with guest agent channel and install guest agent rpm in domain.

Parameters

vm – the vm object

```
virttest.utils_test.libvirt.set_vm_disk (vm, params, tmp_dir=None, test=None)
```

Replace vm first disk with given type in domain xml, including file type (local, nfs), network type (gluster, iscsi), block type (use connected iscsi block disk).

For all types, all following params are common and need be specified:

disk_device: default to ‘disk’ disk_type: ‘block’ or ‘network’ disk_target: default to ‘vda’
disk_target_bus: default to ‘virtio’ disk_format: default to ‘qcow2’ disk_src_protocol: ‘iscsi’, ‘gluster’ or ‘netfs’

For ‘gluster’ network type, following params are gluster only and need be specified:

vol_name: string pool_name: default to ‘gluster-pool’ transport: ‘tcp’, ‘rdma’ or ‘’, default to ‘’

For ‘iscsi’ network type, following params need be specified:

image_size: default to “10G”, 10G is raw size of jeos disk disk_src_host: default to “127.0.0.1”
disk_src_port: default to “3260”

For ‘netfs’ network type, following params need be specified:

mnt_path_name: the mount dir name, default to “nfs-mount” export_options: nfs mount options,
default to “rw,no_root_squash,fsid=0”

For ‘block’ type, using connected iscsi block disk, following params need be specified:

image_size: default to “10G”, 10G is raw size of jeos disk

Parameters

- **vm** – the vm object
- **tmp_dir** – string, dir path
- **params** – dict, dict include setup vm disk xml configurations

```
virttest.utils_test.libvirt.setup_or_cleanup_gluster (is_setup,          vol_name,
                                                         brick_path='', pool_name='',
                                                         file_path='/etc/glusterfs/glusterd.vol')
```

Set up or clean up glusterfs environment on localhost :param is_setup: Boolean value, true for setup, false for cleanup :param vol_name: gluster created volume name :param brick_path: Dir for create glusterfs :return: ip_addr or nothing

```
virttest.utils_test.libvirt.setup_or_cleanup_iscsi(is_setup, is_login=True,
                                                    emulated_image='emulated-
iscsi', image_size='1G',
                                                    chap_user='', chap_passwd='',
                                                    restart_tgtd='no', portal_ip='127.0.0.1')
```

Set up(and login iscsi target) or clean up iscsi service on localhost.

Parameters

- **is_setup** – Boolean value, true for setup, false for cleanup
- **is_login** – Boolean value, true for login, false for not login
- **emulated_image** – name of iscsi device
- **image_size** – emulated image's size
- **chap_user** – CHAP authentication username
- **chap_passwd** – CHAP authentication password

Returns iscsi device name or iscsi target

```
virttest.utils_test.libvirt.setup_or_cleanup_nfs(is_setup, mount_dir='nfs-
mount', is_mount=False, ex-
port_options='rw, no_root_squash',
mount_options='rw',
export_dir='nfs-export', re-
store_selinux='')
```

Set SELinux to “permissive” and Set up nfs service on localhost. Or clean up nfs service on localhost and restore SELinux.

Note: SELinux status must be backed up and restored after use. Example:

```
# Setup NFS. res = setup_or_cleanup_nfs(is_setup=True) # Backup SELinux status. selinux_bak =
res['selinux_status_bak']
```

```
# Do something. ...
```

```
# Cleanup NFS and restore NFS. res = setup_or_cleanup_nfs(is_setup=False, restore_selinux=selinux_bak)
```

Parameters

- **is_setup** – Boolean value, true for setup, false for cleanup
- **mount_dir** – NFS mount dir. This can be an absolute path on the host or a relative path origin from libvirt tmp dir. Default to “nfs-mount”.
- **is_mount** – Boolean value, Whether the target NFS should be mounted.
- **export_options** – Options for nfs dir. Default to “nfs-export”.
- **mount_options** – Options for mounting nfs dir. Default to “rw”.
- **export_dir** – NFS export dir. This can be an absolute path on the host or a relative path origin from libvirt tmp dir. Default to “nfs-export”.

Returns A dict contains export and mount result parameters: export_dir: Absolute directory of exported local NFS file system. mount_dir: Absolute directory NFS file system mounted on. selinux_status_bak: SELinux status before set

```
virttest.utils_test.libvirt.update_polkit_rule(params, pattern, new_value)
```

This function help to update the rule during testing.

Parameters

- **params** – Test run params
- **pattern** – Regex pattern for updating
- **new_value** – New value for updating

```
virttest.utils_test.libvirt.update_vm_disk_source (vm_name, disk_source_path,
                                                    source_type='file')
```

Update disk source path of the VM

Parameters **source_type** – it may be 'dev' or 'file' type, which is default

```
virttest.utils_test.libvirt.verify_virsh_console (session, user, passwd, timeout=10,
                                                  debug=False)
```

Run commands in console session.

```
virttest.utils_test.libvirt.yum_install (pkg_list, session=None)
```

Try to install packages on system

virttest.utils_test.qemu module High-level QEMU test utility functions.

This module is meant to reduce code size by performing common test procedures. Generally, code here should look like test code.

More specifically:

- Functions in this module should raise exceptions if things go wrong
- Functions in this module typically use functions and classes from lower-level modules (e.g. `utils_misc`, `qemu_vm`, `aexpect`).
- Functions in this module should not be used by lower-level modules.
- Functions in this module should be used in the right context. For example, a function should not be used where it may display misleading or inaccurate info or debug messages.

copyright 2008-2013 Red Hat Inc.

```
class virttest.utils_test.qemu.GuestSuspend (params, vm)
```

Bases: `object`

Suspend guest, supports both Linux and Windows.

SUSPEND_TYPE_DISK = 'disk'

SUSPEND_TYPE_MEM = 'mem'

action_after_suspend (*args, **kwargs)

action_before_suspend (*args, **kwargs)

action_during_suspend (*args, **kwargs)

check_bg_program (*args, **kwargs)

Make sure the background program is running as expected

kill_bg_program (*args, **kwargs)

resume_guest_disk (*args, **kwargs)

resume_guest_mem (*args, **kwargs)

setup_bg_program (*args, **kwargs)

Start up a program as a flag in guest.

start_suspend (*args, **kwargs)

```
verify_guest_down(*args, **kwargs)
```

```
verify_guest_support_suspend(**args)
```

```
verify_guest_up(*args, **kwargs)
```

```
class virttest.utils_test.qemu.MigrationData(params, srchost, dsthost, vms_name,
                                             params_append)
```

Bases: `object`

```
is_dst()
```

Returns True if host is destination.

```
is_src()
```

Returns True if host is source.

```
class virttest.utils_test.qemu.MultihostMigration(test, params, env, preprocess_env=True)
```

Bases: `object`

Class that provides a framework for multi-host migration.

Migration can be run both synchronously and asynchronously. To specify what is going to happen during the multi-host migration, it is necessary to reimplement the method `migration_scenario`. It is possible to start multiple migrations in separate threads, since `self.migrate` is thread safe.

Only one test using multihost migration framework should be started on one machine otherwise it is necessary to solve the problem with listen server port.

Multihost migration starts `SyncListenServer` through which all messages are transferred, since the multiple hosts can be in different states.

Class `SyncData` is used to transfer data over network or synchronize the migration process. Synchronization sessions are recognized by `session_id`.

It is important to note that, in order to have multi-host migration, one needs shared guest image storage. The simplest case is when the guest images are on an NFS server.

Example:

```
class TestMultihostMigration(utils_misc.MultihostMigration):
    def __init__(self, test, params, env):
        super(TestMultihostMigration, self).__init__(test, params, env)

    def migration_scenario(self):
        srchost = self.params.get("hosts")[0]
        dsthost = self.params.get("hosts")[1]

    def worker(mig_data):
        vm = env.get_vm("vm1")
        session = vm.wait_for_login(timeout=self.login_timeout)
        session.sendline("nohup dd if=/dev/zero of=/dev/null &")
        session.cmd("killall -0 dd")

    def check_worker(mig_data):
        vm = env.get_vm("vm1")
        session = vm.wait_for_login(timeout=self.login_timeout)
        session.cmd("killall -9 dd")

    # Almost synchronized migration, waiting to end it.
    # Work is started only on first VM.
    self.migrate_wait(["vm1", "vm2"], srchost, dsthost,
```

```

        worker, check_worker)

    # Migration started in different threads.
    # It allows to start multiple migrations simultaneously.
    mig1 = self.migrate(["vm1"], srchost, dsthost,
                        worker, check_worker)
    mig2 = self.migrate(["vm2"], srchost, dsthost)
    mig2.join()
    mig1.join()

mig = TestMultihostMigration(test, params, env)
mig.run()

```

before_migration (*mig_data*)

Do something right before migration.

Parameters *mig_data* – object with migration data.

check_vms_dst (*mig_data*)

Check vms after migrate.

Parameters *mig_data* – object with migration data.

check_vms_src (*mig_data*)

Check vms after migrate.

Parameters *mig_data* – object with migration data.

cleanup ()

Cleanup env after test.

master_id ()**migrate** (*vms_name*, *srchost*, *dsthost*, *start_work=None*, *check_work=None*, *params_append=None*)

Migrate machine from srchost to dsthost. It executes start_work on source machine before migration and executes check_work on dsthost after migration.

Migration execution progress:

```

source host          |   dest host
-----
    prepare guest on both sides of migration
    - start machine and check if machine works
    - synchronize transfer data needed for migration
-----
start work on source guests |   wait for migration
-----
            migrate guest to dest host.
            wait on finish migration synchronization
-----
                                |   check work on vms
-----
                        wait for sync on finish migration

```

Parameters

- **vms_name** – List of vms.
- **srchost** – src host id.
- **dsthost** – dst host id.

- **start_work** – Function started before migration.
- **check_work** – Function started after migration.
- **params_append** – Append params to self.params only for migration.

migrate_vms (*mig_data*)

Migrate vms.

migrate_vms_dest (*mig_data*)

Migrate vms destination. This function is started on dest host during migration.

Parameters mig_Data – Data for migration.

migrate_vms_src (*mig_data*)

Migrate vms source.

Parameters mig_Data – Data for migration.

For change way how machine migrates is necessary re implement this method.

migrate_wait (*vms_name*, *srchost*, *dsthost*, *start_work=None*, *check_work=None*,
params_append=None)

Migrate machine from srchost to dsthost and wait for finish. It executes start_work on source machine before migration and executes check_work on dsthost after migration.

Parameters

- **vms_name** – List of vms.
- **srchost** – src host id.
- **dsthost** – dst host id.
- **start_work** – Function which is started before migration.
- **check_work** – Function which is started after done of migration.

migration_scenario ()

Multi Host migration_scenario is started from method run where the exceptions are checked. It is not necessary to take care of cleaning up after test crash or finish.

post_migration (*vm*, *cancel_delay*, *mig_offline*, *dsthost*, *vm_ports*, *not_wait_for_migration*, *fd*,
mig_data)

postprocess_env ()

Kill vms and delete cloned images.

prepare_for_migration (*mig_data*, *migration_mode*)

Prepare destination of migration for migration.

Parameters

- **mig_data** – Class with data necessary for migration.
- **migration_mode** – Migration mode for prepare machine.

preprocess_env ()

Prepare env to start vms.

run ()

Start multihost migration scenario. After scenario is finished or if scenario crashed it calls postprocess machines and cleanup env.

class virttest.utils_test.qemu.**MultihostMigrationExec** (*test*, *params*, *env*, *preprocess_env=True*)

Bases: *virttest.utils_test.qemu.MultihostMigration*

migrate_vms_src (*mig_data*)

Migrate vms source.

Parameters **mig_Data** – Data for migration.

For change way how machine migrates is necessary re implement this method.

migrate_wait (*vms_name*, *srchost*, *dsthost*, *start_work=None*, *check_work=None*,
params_append=None)

post_migration (*vm*, *cancel_delay*, *mig_offline*, *dsthost*, *mig_exec_cmd*, *not_wait_for_migration*, *fd*,
mig_data)

class `virttest.utils_test.qemu.MultihostMigrationFd` (*test*, *params*, *env*, *preprocess_env=True*)

Bases: `virttest.utils_test.qemu.MultihostMigration`

migrate_vms_src (*mig_data*)

Migrate vms source.

Parameters **mig_Data** – Data for migration.

For change way how machine migrates is necessary re implement this method.

migrate_wait (*vms_name*, *srchost*, *dsthost*, *start_work=None*, *check_work=None*,
params_append=None)

class `virttest.utils_test.qemu.MultihostMigrationRdma` (*test*, *params*, *env*, *preprocess_env=True*)

Bases: `virttest.utils_test.qemu.MultihostMigration`

migrate_vms_src (*mig_data*)

Migrate vms source.

Parameters **mig_Data** – Data for migration.

For change way how machine migrates is necessary re implement this method.

`virttest.utils_test.qemu.clear_win_driver_verifier` (*session*, *vm*, *timeout=300*)

Clear the driver verifier in windows guest.

Parameters

- **session** – VM session.
- **timeout** – Timeout in seconds.

Returns Session after reboot.

`virttest.utils_test.qemu.get_numa_status` (*numa_node_info*, *qemu_pid*, *debug=True*)

Get the qemu process memory use status and the cpu list in each node.

Parameters

- **numa_node_info** (*string*) – Host numa node information
- **qemu_pid** – process id of qemu
- **debug** (*bool*) – Print the debug info or not

Returns memory and cpu list in each node

Return type `tuple`

`virttest.utils_test.qemu.guest_active` (*vm*)

```
virttest.utils_test.qemu.migrate(vm, env=None, mig_timeout=3600, mig_protocol='tcp',
                                mig_cancel=False, offline=False, stable_check=False,
                                clean=False, save_path=None, dest_host='localhost',
                                mig_port=None)
```

Migrate a VM locally and re-register it in the environment.

Parameters

- **vm** – The VM to migrate.
- **env** – The environment dictionary. If omitted, the migrated VM will not be registered.
- **mig_timeout** – timeout value for migration.
- **mig_protocol** – migration protocol
- **mig_cancel** – Test migrate_cancel or not when protocol is tcp.
- **dest_host** – Destination host (defaults to 'localhost').
- **mig_port** – Port that will be used for migration.

Returns The post-migration VM, in case of same host migration, True in case of multi-host migration.

```
virttest.utils_test.qemu.pin_vm_threads(vm, node)
```

Pin VM threads to single cpu of a numa node

Parameters

- **vm** – VM object
- **node** – NumaNode object

```
virttest.utils_test.qemu.setup_win_driver_verifier(session, driver, vm, timeout=300)
```

Enable driver verifier for windows guest.

Parameters

- **session** – VM session.
- **driver** – The driver which needs enable the verifier.
- **vm** – VM object.
- **timeout** – Timeout in seconds.

Returns Session after reboot.

Module contents High-level virt test utility functions.

This module is meant to reduce code size by performing common test procedures. Generally, code here should look like test code.

More specifically:

- Functions in this module should raise exceptions if things go wrong
- Functions in this module typically use functions and classes from lower-level modules (e.g. `utils_misc`, `aexpect`).
- Functions in this module should not be used by lower-level modules.
- Functions in this module should be used in the right context. For example, a function should not be used where it may display misleading or inaccurate info or debug messages.

copyright 2008-2013 Red Hat Inc.

class `virttest.utils_test.BackgroundTest` (*func, params, kwargs={}*)

Bases: `object`

This class would run a test in background through a dedicated thread.

is_alive ()

Check whether the test is still alive.

join (*timeout=600, ignore_status=False*)

Wait for the join of thread and raise its exception if any.

launch (*func, params, kwargs*)

Catch and record the exception.

start ()

Run func(params) in a dedicated thread

class `virttest.utils_test.HostStress` (*params, stress_type*)

Bases: `object`

Run Stress tool on host, such as stress, unixbench, iotest and etc.

app_running ()

check whether app really run in background

load_stress_tool (**args, **kwargs*)

load stress tool on host.

unload_stress (**args, **kwargs*)

stop stress tool manually

class `virttest.utils_test.RemoteDiskManager` (*params*)

Bases: `object`

Control images on remote host

create_image (*disk_type, path=None, size=10, vname=None, lvname=None, sparse=True, timeout=60, img_fmt=None*)

Create an image for target path.

create_vg (*vname, device*)

Create volume group with provided device.

get_free_space (*disk_type, path='/', vname=None*)

Get free space of remote host for path.

:return : the unit is 'G'.

iscsi_login_setup (*host, target_name, is_login=True*)

Login or logout to a target on remote host.

occupy_space (*disk_type, need_size, path=None, vname=None, timeout=60*)

Create an image or volume to occupy the space of destination path

remove_path (*disk_type, path*)

Only allowed to remove path to file or volume.

remove_vg (*vname*)

Remove volume group on remote host.

exception `virttest.utils_test.StressError` (*msg*)

Bases: `exceptions.Exception`

Stress test exception.

```
class virttest.utils_test.VMStress (vm, stress_type)
```

Bases: `object`

Run Stress tool in vms, such as stress, unixbench, iotest and etc.

```
app_running()
```

check whether app really run in background

```
get_session()
```

```
load_stress_tool (*args, **kwargs)
```

load stress tool in guest

```
unload_stress (*args, **kwargs)
```

stop stress tool manually

```
virttest.utils_test.canonicalize_disk_address (disk_address)
```

Canonicalize disk address. Convert {decimal|octal|hexadecimal} to decimal
pci:0x0000.0x00.0x0b.0x0 =>
pci:0.0.11.0 ide:00.00.00 => ide:0.0.0 scsi:00.00.0x11 => scsi:0.0.17

```
virttest.utils_test.check_dest_vm_network (vm, ip, remote_host, username, password)
```

Ping migrated vms on remote host.

```
virttest.utils_test.get_date (session=None)
```

Get the date time

```
virttest.utils_test.get_driver_hardware_id (driver_path, mount_point='/tmp/mnt-  
virtio', storage_path='/tmp/prewhql.iso',  
re_hw_id='(PCI.{14, 50})', run_cmd=True)
```

Get windows driver's hardware id from inf files.

Parameters

- **dirver** – Configurable driver name.
- **mount_point** – Mount point for the driver storage
- **storage_path** – The path of the virtio driver storage
- **re_hw_id** – the pattern for getting hardware id from inf files
- **run_cmd** – Use hardware id in windows cmd command or not

Returns Windows driver's hardware id

```
virttest.utils_test.get_image_info (image_file)
```

```
virttest.utils_test.get_loss_ratio (output)
```

Get the packet loss ratio from the output of ping.

Parameters `output` – Ping output.

```
virttest.utils_test.get_memory_info (lvms)
```

Get memory information from host and guests in format: Host: memfree = XXXM; Guests memsh = {XXX,XXX,...}

Params `lvms` List of VM objects

Returns String with memory info report

```
virttest.utils_test.get_readable_cdroms (params, session)
```

Get the cdrom list which contain media in guest.

Parameters

- **params** – Dictionary with the test parameters.

- **session** – A shell session on the VM provided.

`virttest.utils_test.get_time(session, time_command, time_filter_re, time_format)`

Return the host time and guest time. If the guest time cannot be fetched a `TestError` exception is raised.

Note that the shell session should be ready to receive commands (i.e. should “display” a command prompt and should be done with all previous commands).

Parameters

- **session** – A shell session.
- **time_command** – Command to issue to get the current guest time.
- **time_filter_re** – Regex filter to apply on the output of `time_command` in order to get the current time.
- **time_format** – Format string to pass to `time.strptime()` with the result of the regex filter.

Returns A tuple containing the host time and guest time.

`virttest.utils_test.get_windows_disk_drive(session, filename, extension='exe', tmout=240)`

Get the windows disk drive number

`virttest.utils_test.get_windows_file_abs_path(session, filename, extension='exe', tmout=240)`

return file abs path “drive+path” by “wmic datafile”

`virttest.utils_test.load_stress(stress_type, vms, params)`

Load stress for tests.

Parameters

- **stress_type** – The stress type you need
- **params** – Useful parameters for stress
- **vms** – Used when it's stress in vms

`virttest.utils_test.ntpdate(service_ip, session=None)`

set the date and time via NTP

`virttest.utils_test.ping(dest=None, count=None, interval=None, interface=None, packet_size=None, ttl=None, hint=None, adaptive=False, broadcast=False, flood=False, timeout=0, output_func=<function debug>, session=None)`

Wrapper of ping.

Parameters

- **dest** – Destination address.
- **count** – Count of icmp packet.
- **interval** – Interval of two icmp echo request.
- **interface** – Specified interface of the source address.
- **packet_size** – Packet size of icmp.
- **ttl** – IP time to live.
- **hint** – Path mtu discovery hint.
- **adaptive** – Adaptive ping flag.
- **broadcast** – Broadcast ping flag.
- **flood** – Flood ping flag.

- **timeout** – Timeout for the ping command.
- **output_func** – Function used to log the result of ping.
- **session** – Local execution hint or session to execute the ping command.

`virttest.utils_test.raw_ping(command, timeout, session, output_func)`

Low-level ping command execution.

Parameters

- **command** – Ping command.
- **timeout** – Timeout of the ping command.
- **session** – Local execution hint or session to execute the ping command.

`virttest.utils_test.run_autotest(vm, session, control_path, timeout, outputdir, params, copy_only=False, control_args=None, ignore_session_terminated=False)`

Run an autotest control file inside a guest (linux only utility).

Parameters

- **vm** – VM object.
- **session** – A shell session on the VM provided.
- **control_path** – A path to an autotest control file.
- **timeout** – Timeout under which the autotest control file must complete.
- **outputdir** – Path on host where we should copy the guest autotest results to.
- **copy_only** – If `copy_only` is `True`, copy the autotest to guest and return the command which need to run test on guest, without executing it.
- **control_args** – The arguments for control file.
- **ignore_session_terminated** – If set up this parameter to `True` we will ignore the session terminated during test.

The following params is used by the migration :param params: Test params used in the migration test

`virttest.utils_test.run_virt_sub_test(test, params, env, sub_type=None, tag=None)`

Call another test script in one test script. :param test: Virt Test object. :param params: Dictionary with the test parameters. :param env: Dictionary with test environment. :param sub_type: Type of called test script. :param tag: Tag for get the sub_test params

`virttest.utils_test.service_setup(vm, session, directory)`

`virttest.utils_test.start_windows_service(session, service, timeout=120)`

Start a Windows service using sc. If the service is already running, do nothing. If the service isn't installed, fail.

Parameters

- **service** – The name of the service
- **timeout** – Time duration to wait for service to start

Raises `error.TestError` – Raised if the service can't be started

`virttest.utils_test.stop_windows_service(session, service, timeout=120)`

Stop a Windows service using sc. If the service is already stopped or is not installed, do nothing.

Parameters

- **service** – The name of the service

- **timeout** – Time duration to wait for service to stop

Raises `error.TestError` – Raised if the service can't be stopped

`virttest.utils_test.summary_up_result(result_file, ignore, row_head, column_mark)`

Use to summary the monitor or other kinds of results. Now it calculates the average value for each item in the results. It fits to the records that are in matrix form.

@result_file: files which need to calculate @ignore: pattern for the comment in results which need to through away @row_head: pattern for the items in row @column_mark: pattern for the first line in matrix which used to generate the items in column :return: A dictionary with the average value of results

`virttest.utils_test.unload_stress(stress_type, vms)`

Unload stress loaded by load_stress(...).

Submodules

virttest.RFBDes module

class `virttest.RFBDes.Des` (*key*)

Bases: `object`

Base Data Encryption Standard class. For details, please refer to: http://en.wikipedia.org/wiki/Data_Encryption_Standard

E = [31, 0, 1, 2, 3, 4, 3, 4, 5, 6, 7, 8, 7, 8, 9, 10, 11, 12, 11, 12, 13, 14, 15, 16, 15, 16, 17, 18, 19, 20, 19, 20, 21, 22, 23, 24, 23, 24,

FP = [39, 7, 47, 15, 55, 23, 63, 31, 38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29, 36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 4,

IP = [57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3, 61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7, 56, 48, 4,

P = [15, 6, 19, 20, 28, 11, 27, 16, 0, 14, 22, 25, 4, 17, 30, 9, 1, 7, 23, 13, 31, 26, 2, 8, 18, 12, 29, 5, 21, 10, 3, 24]

PC1 = [56, 48, 40, 32, 24, 16, 8, 0, 57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2, 59, 51, 43, 35, 62, 54, 46, 38, 30, 22,

PC2 = [13, 16, 10, 23, 0, 4, 2, 27, 14, 5, 20, 9, 22, 18, 11, 3, 25, 7, 15, 6, 26, 19, 12, 1, 40, 51, 30, 36, 46, 54, 29, 39, 50, 44, 32, 4,

create_Kn ()

Create the 16 subkeys, from K[0] to K[15], from the given key

crypt (*data*, *crypt_type*=0)

Crypt the data in blocks, running it through des_crypt()

Parameters

- **data** – Data to be encrypted/decrypted.
- **crypt_type** – crypt type. 0 means encrypt, and 1 means decrypt.

des_crypt (*data*, *crypt_type*=0)

Crypt the block of data through DES bit-manipulation

Parameters

- **data** – data need to crypt.
- **crypt_type** – crypt type. 0 means encrypt, and 1 means decrypt.

f (*K*)

The Feistel function (F-function) of DES, operates on half a block (32 bits) at a time and consists of four stages: 1. Expansion 2. Key mixing 3. Substitution 4. Permutation

Parameters K – One of sixteen 48-bit subkeys are derived from the main key.

getKey()

Just get the crypting key.

get_sub_list (*table*, *block*)

Return sub list of block according to index in table.

Parameters

- **table** – Index list.
- **block** – bit list used to get sub list.

left_rotations = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1]

sbox = [[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7, 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8, 4, 1, 14, 8, 13, 6, 2, 11,

setKey (*key*)

Will set the crypting key for this object. RFB protocol for authentication requires client to encrypt challenge sent by server with password using DES method. However, bits in each byte of the password are put in reverse order before using it as encryption key.

Parameters **key** – Original used in DES.

virttest.aexpect module

A class and functions used for running and controlling child processes.

copyright 2008-2009 Red Hat Inc.

```
class virttest.aexpect.Expect (command=None, a_id=None, auto_close=True, echo=False,
                              linesep='n', termination_func=None, termination_params=(),
                              output_func=None, output_params=(), output_prefix='',
                              thread_name=None)
```

Bases: `virttest.aexpect.Tail`

This class runs a child process in the background and provides expect-like services.

It also provides all of Tail's functionality.

match_patterns (*cont*, *patterns*)

Match cont against a list of patterns.

Return the index of the first pattern that matches a substring of cont. None and empty strings in patterns are ignored. If no match is found, return None.

Parameters

- **cont** – input string
- **patterns** – List of strings (regular expression patterns).

match_patterns_multiline (*cont*, *patterns*)

Match list of lines against a list of patterns.

Return the index of the first pattern that matches a substring of cont. None and empty strings in patterns are ignored. If no match is found, return None.

Parameters

- **cont** – List of strings (input strings)
- **patterns** – List of strings (regular expression patterns). The pattern priority is from the last to first.

read_nonblocking (*internal_timeout=None, timeout=None*)

Read from child until there is nothing to read for timeout seconds.

Parameters

- **internal_timeout** – Time (seconds) to wait before we give up reading from the child process, or None to use the default value.
- **timeout** – Timeout for reading child process output.

read_until_any_line_matches (*patterns, timeout=60, internal_timeout=None, print_func=None*)

Read using read_nonblocking until any line matches a pattern.

Read using read_nonblocking until any line of the output matches one of the patterns (using match_patterns_multiline), or until timeout expires. Return a tuple containing the match index (or None if no match was found) and the data read so far.

Parameters

- **patterns** – A list of strings (regular expression patterns) Consider using '^' in the beginning.
- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple containing the match index and the data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires
- **ExpectProcessTerminatedError** – Raised if the child process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

read_until_last_line_matches (*patterns, timeout=60, internal_timeout=None, print_func=None*)

Read using read_nonblocking until the last non-empty line matches a pattern.

Read using read_nonblocking until the last non-empty line of the output matches one of the patterns (using match_patterns), or until timeout expires. Return a tuple containing the match index (or None if no match was found) and the data read so far.

Parameters

- **patterns** – A list of strings (regular expression patterns)
- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple containing the match index and the data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires

- **ExpectProcessTerminatedError** – Raised if the child process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

read_until_last_word_matches (*patterns*, *timeout=60*, *internal_timeout=None*, *print_func=None*)

Read using `read_nonblocking` until the last word of the output matches one of the patterns (using `match_patterns`), or until timeout expires.

Parameters

- **patterns** – A list of strings (regular expression patterns)
- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to `read_nonblocking`
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple containing the match index and the data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires
- **ExpectProcessTerminatedError** – Raised if the child process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

read_until_output_matches (*patterns*, *filter_func=<function <lambda>>*, *timeout=60*, *internal_timeout=None*, *print_func=None*, *match_func=None*)

Read from child using `read_nonblocking` until a pattern matches.

Read using `read_nonblocking` until a match is found using `match_patterns`, or until timeout expires. Before attempting to search for a match, the data is filtered using the `filter_func` function provided.

Parameters

- **patterns** – List of strings (regular expression patterns)
- **filter_func** – Function to apply to the data read from the child before attempting to match it against the patterns (should take and return a string)
- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to `read_nonblocking`
- **print_func** – A function to be used to print the data being read (should take a string parameter)
- **match_func** – Function to compare the output and patterns.

Returns Tuple containing the match index and the data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires
- **ExpectProcessTerminatedError** – Raised if the child process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

exception `virttest.aexpect.ExpectError` (*patterns*, *output*)

Bases: `exceptions.Exception`

exception `virttest.aexpect.ExpectProcessTerminatedError` (*patterns, status, output*)

Bases: `virttest.aexpect.ExpectError`

exception `virttest.aexpect.ExpectTimeoutError` (*patterns, output*)

Bases: `virttest.aexpect.ExpectError`

exception `virttest.aexpect.ShellCmdError` (*cmd, status, output*)

Bases: `virttest.aexpect.ShellError`

exception `virttest.aexpect.ShellError` (*cmd, output*)

Bases: `exceptions.Exception`

exception `virttest.aexpect.ShellProcessTerminatedError` (*cmd, status, output*)

Bases: `virttest.aexpect.ShellError`

class `virttest.aexpect.ShellSession` (*command=None, a_id=None, auto_close=True, echo=False, linesep='n', termination_func=None, termination_params=(), output_func=None, output_params=(), output_prefix='', thread_name=None, prompt='[\#\\$\]\s*\$', status_test_command='echo \$?')*

Bases: `virttest.aexpect.Expect`

This class runs a child process in the background. It is suited for processes that provide an interactive shell, such as SSH and Telnet.

It provides all services of Expect and Tail. In addition, it provides command running services, and a utility function to test the process for responsiveness.

cmd (*cmd, timeout=60, internal_timeout=None, print_func=None, ok_status=[0], ignore_all_errors=False*)

Send a command and return its output. If the command's exit status is nonzero, raise an exception.

Parameters

- **cmd** – Command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to `read_nonblocking`
- **print_func** – A function to be used to print the data being read (should take a string parameter)
- **ok_status** – do not raise `ShellCmdError` in case that exit status is one of `ok_status`. (default is `[0,]`)
- **ignore_all_errors** – toggles whether or not an exception should be raised on any error.

Returns The output of `cmd`

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellError** – Raised if the exit status cannot be obtained or if an unknown error occurs
- **ShellStatusError** – Raised if the exit status cannot be obtained
- **ShellError** – Raised if an unknown error occurs
- **ShellCmdError** – Raised if the exit status is nonzero

cmd_output (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Send a command and return its output.

Parameters

- **cmd** – Command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns The output of cmd

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellError** – Raised if an unknown error occurs

cmd_output_safe (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Send a command and return its output (serial sessions).

In serial sessions, frequently the kernel might print debug or error messages that make read_up_to_prompt to timeout. Let's try to be a little more robust and send a carriage return, to see if we can get to the prompt.

Parameters

- **cmd** – Command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns The output of cmd

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellError** – Raised if an unknown error occurs

cmd_status (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Send a command and return its exit status.

Parameters

- **cmd** – Command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns The exit status of cmd

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellStatusError** – Raised if the exit status cannot be obtained
- **ShellError** – Raised if an unknown error occurs

cmd_status_output (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Send a command and return its exit status and output.

Parameters

- **cmd** – Command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to `read_nonblocking`
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple (status, output) where status is the exit status and output is the output of cmd

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellStatusError** – Raised if the exit status cannot be obtained
- **ShellError** – Raised if an unknown error occurs

get_command_output (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Alias for `cmd_output()` for backward compatibility.

get_command_status (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Alias for `cmd_status()` for backward compatibility.

get_command_status_output (*cmd, timeout=60, internal_timeout=None, print_func=None*)

Alias for `cmd_status_output()` for backward compatibility.

is_responsive (*timeout=5.0*)

Return True if the process responds to STDIN/terminal input.

Send a newline to the child process (e.g. SSH or Telnet) and read some output using `read_nonblocking()`. If all is OK, some output should be available (e.g. the shell prompt). In that case return True. Otherwise return False.

Parameters **timeout** – Time duration to wait before the process is considered unresponsive.

read_up_to_prompt (*timeout=60, internal_timeout=None, print_func=None*)

Read using `read_nonblocking` until the last non-empty line matches the prompt.

Read using `read_nonblocking` until the last non-empty line of the output matches the prompt regular expression set by `set_prompt`, or until timeout expires.

Parameters

- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to `read_nonblocking`

- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns The data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires
- **ExpectProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

classmethod `remove_command_echo (cont, cmd)`

classmethod `remove_last_nonempty_line (cont)`

set_prompt (*prompt*)

Set the prompt attribute for later use by `read_up_to_prompt`.

:param String that describes the prompt contents.

set_status_test_command (*status_test_command*)

Set the command to be sent in order to get the last exit status.

Parameters **status_test_command** – Command that will be sent to get the last exit status.

exception `virttest.aexpect.ShellStatusError (cmd, output)`

Bases: `virttest.aexpect.ShellError`

exception `virttest.aexpect.ShellTimeoutError (cmd, output)`

Bases: `virttest.aexpect.ShellError`

class `virttest.aexpect.Spawn (command=None, a_id=None, auto_close=False, echo=False, line-sep='n')`

Bases: `object`

This class is used for spawning and controlling a child process.

A new instance of this class can either run a new server (a small Python program that reads output from the child process and reports it to the client and to a text file) or attach to an already running server. When a server is started it runs the child process. The server writes output from the child's STDOUT and STDERR to a text file. The text file can be accessed at any time using `get_output()`. In addition, the server opens as many pipes as requested by the client and writes the output to them. The pipes are requested and accessed by classes derived from `Spawn`. These pipes are referred to as “readers”. The server also receives input from the client and sends it to the child process. An instance of this class can be pickled. Every derived class is responsible for restoring its own state by properly defining `__getinitargs__()`.

The first named pipe is used by `_tail()`, a function that runs in the background and reports new output from the child as it is produced. The second named pipe is used by a set of functions that read and parse output as requested by the user in an interactive manner, similar to `pexpect`. When unpickled it automatically resumes `_tail()` if needed.

close (*sig=9*)

Kill the child process if it's alive and remove temporary files.

Parameters **sig** – The signal to send the process when attempting to kill it.

get_id ()

Return the instance's `a_id` attribute, which may be used to access the process in the future.

get_output ()

Return the STDOUT and STDERR output of the process so far.

get_pid()

Return the PID of the process.

Note: this may be the PID of the shell process running the user given command.

get_status()

Wait for the process to exit and return its exit status, or None if the exit status is not available.

get_stripped_output()

Return the STDOUT and STDERR output without the console codes escape and sequences of the process so far.

is_alive()

Return True if the process is running.

is_defunct()

Return True if the process is defunct (zombie).

kill(sig=9)

Kill the child process if alive

send(cont='')

Send a string to the child process.

Parameters cont – String to send to the child process.

send_ctrl(control_str='')

Send a control string to the aexpect process.

Parameters control_str – Control string to send to the child process container.

sendline(cont='')

Send a string followed by a line separator to the child process.

Parameters cont – String to send to the child process.

set_linesep(linesep)

Sets the line separator string (usually “\n”).

Parameters linesep – Line separator string.

```
class virttest.aexpect.Tail(command=None, a_id=None, auto_close=False, echo=False,
                             linesep='n', termination_func=None, termination_params=(),
                             output_func=None, output_params=(), output_prefix='',
                             thread_name=None)
```

Bases: *virttest.aexpect.Spawn*

This class runs a child process in the background and sends its output in real time, line-by-line, to a callback function.

See Spawn’s docstring.

This class uses a single pipe reader to read data in real time from the child process and report it to a given callback function. When the child process exits, its exit status is reported to an additional callback function.

When this class is unpickled, it automatically resumes reporting output.

set_log_file(filename)

Set a log file name for this tail instance.

Parameters filename – Base name of the log.

set_output_func(output_func)

Set the output_func attribute. See `__init__()` for details.

Parameters **output_func** – Function to call for each line of STDOUT/STDERR output from the process. Must take a single string parameter.

set_output_params (*output_params*)

Set the output_params attribute. See `__init__()` for details.

Parameters **output_params** – Parameters to send to output_func before the output line.

set_output_prefix (*output_prefix*)

Set the output_prefix attribute. See `__init__()` for details.

Parameters **output_prefix** – String to pre-pend to each line sent to output_func (see `set_output_callback()`).

set_termination_func (*termination_func*)

Set the termination_func attribute. See `__init__()` for details.

Parameters **termination_func** – Function to call when the process terminates. Must take a single parameter – the exit status.

set_termination_params (*termination_params*)

Set the termination_params attribute. See `__init__()` for details.

Parameters **termination_params** – Parameters to send to termination_func before the exit status.

`virttest.aexpect.clean_tmp_files()`

Remove all aexpect temporary files.

`virttest.aexpect.kill_tail_threads()`

Kill all Tail threads.

After calling this function no new threads should be started.

`virttest.aexpect.run_bg(command, termination_func=None, output_func=None, output_prefix='', timeout=1.0, auto_close=True)`

Run a subprocess in the background and collect its output and exit status.

Run command as a subprocess. Call output_func with each line of output from the subprocess (prefixed by output_prefix). Call termination_func when the subprocess terminates. Return when timeout expires or when the subprocess exits – whichever occurs first.

Parameters

- **command** – The shell command to execute
- **termination_func** – A function to call when the process terminates (should take an integer exit status parameter)
- **output_func** – A function to call with each line of output from the subprocess (should take a string parameter)
- **output_prefix** – A string to pre-pend to each line of the output, before passing it to stdout_func
- **timeout** – Time duration (in seconds) to wait for the subprocess to terminate before returning
- **auto_close** – If True, close() the instance automatically when its reference count drops to zero (default False).

Returns A Expect object.

`virttest.aexpect.run_fg(command, output_func=None, output_prefix='', timeout=1.0)`

Run a subprocess in the foreground and collect its output and exit status.

Run command as a subprocess. Call `output_func` with each line of output from the subprocess (prefixed by `prefix`). Return when timeout expires or when the subprocess exits – whichever occurs first. If timeout expires and the subprocess is still running, kill it before returning.

Parameters

- **command** – The shell command to execute
- **output_func** – A function to call with each line of output from the subprocess (should take a string parameter)
- **output_prefix** – A string to pre-pend to each line of the output, before passing it to `stdout_func`
- **timeout** – Time duration (in seconds) to wait for the subprocess to terminate before killing it and returning

Returns A 2-tuple containing the exit status of the process and its STDOUT/STDERR output. If timeout expires before the process terminates, the returned status is `None`.

```
virttest.aexpect.run_tail(command, termination_func=None, output_func=None, out-  
put_prefix='', timeout=1.0, auto_close=True)
```

Run a subprocess in the background and collect its output and exit status.

Run command as a subprocess. Call `output_func` with each line of output from the subprocess (prefixed by `output_prefix`). Call `termination_func` when the subprocess terminates. Return when timeout expires or when the subprocess exits – whichever occurs first.

Parameters

- **command** – The shell command to execute
- **termination_func** – A function to call when the process terminates (should take an integer exit status parameter)
- **output_func** – A function to call with each line of output from the subprocess (should take a string parameter)
- **output_prefix** – A string to pre-pend to each line of the output, before passing it to `stdout_func`
- **timeout** – Time duration (in seconds) to wait for the subprocess to terminate before returning
- **auto_close** – If `True`, `close()` the instance automatically when its reference count drops to zero (default `False`).

Returns A `Expect` object.

virttest.arch module

```
virttest.arch.get_kvm_module_list()
```

virttest.asset module

```
virttest.asset.download_all_test_providers(update=False)
```

Download all available test providers.

```
virttest.asset.download_asset(asset, interactive=True, restore_image=False)
```

Download an asset defined on an asset file.

Asset files are located under `/shared/downloads`, are `.ini` files with the following keys defined:

title Title string to display in the download progress bar.

url URL of the resource

sha1_url URL with SHA1 information for the resource, in the form sha1sum file_basename

destination Location of your file relative to the data directory (TEST_SUITE_ROOT/shared/data)

destination Location of the uncompressed file relative to the data directory (TEST_SUITE_ROOT/shared/data)

uncompress_cmd Command that needs to be executed with the compressed file as a parameter

Parameters

- **asset** – String describing an asset file.
- **interactive** – Whether to ask the user before downloading the file.
- **restore_image** – If the asset is a compressed image, we can uncompress in order to restore the image.

```
virttest.asset.download_file(asset_info, interactive=False, force=False)
```

Verifies if file that can be find on url is on destination with right hash.

This function will verify the SHA1 hash of the file. If the file appears to be missing or corrupted, let the user know.

Parameters **asset_info** – Dictionary returned by get_asset_info

```
virttest.asset.download_test_provider(provider, update=False)
```

Download a test provider defined on a .ini file inside test-providers.d.

This function will only download test providers that are in git repos. Local filesystems don't need this functionality.

Parameters **provider** – Test provider name, such as 'io-github-autotest-qemu'.

```
virttest.asset.get_all_assets()
```

```
virttest.asset.get_asset_info(asset)
```

```
virttest.asset.get_file_asset(title, src_path, destination)
```

```
virttest.asset.get_known_backends()
```

Return virtualization backends supported by virt-test.

```
virttest.asset.get_test_provider_info(provider)
```

Get a dictionary with relevant test provider info, such as:

- provider uri (git repo or filesystem location)
- provider git repo data, such as branch, ref, pubkey
- **backends that this provider has tests for. For each backend type the** provider has tests for, the 'path' will be also available.

Parameters **provider** – Test provider name, such as 'io-github-autotest-qemu'.

```
virttest.asset.get_test_provider_names(backend=None)
```

Get the names of all test providers available in test-providers.d.

Returns List with the names of all test providers.

`virttest.asset.get_test_provider_subdirs(backend=None)`

Get information of all test provider subdirs for a given backend.

If no backend is provided, return all subdirs with tests.

Parameters `backend` – Backend type, such as ‘qemu’.

Returns List of directories that contain tests for the given backend.

`virttest.asset.uncompress_asset(asset_info, force=False)`

virttest.base_installer module

This module implements classes that perform the installation of the virtualization software on a host system.

These classes can be, and usually are, inherited by subclasses that implement custom logic for each virtualization hypervisor/software.

class `virttest.base_installer.BaseInstaller(mode, name, test=None, params=None)`

Bases: `object`

Base virtualization software installer

This class holds all the skeleton features for installers and should be inherited from when creating a new installer.

install (`cleanup=True, download=True, prepare=True, build=True, install=True, init=True`)

Performs the installation of the virtualization software

This is the main entry point of this class, and should either be reimplemented completely, or simply implement one or many of the install phases.

load_modules (`module_list=None`)

Load Linux Kernel modules the virtualization software may depend on

If `module_directory` is not set, the list of modules will simply be loaded by the system stock `modprobe` tool, meaning that modules will be looked for in the system default module paths.

Parameters `module_list` (`list`) – list of kernel modules names to load

reload_modules ()

Reload the kernel modules (unload, then load)

reload_modules_if_needed ()

set_install_params (`test=None, params=None`)

Called by test to setup parameters from the configuration file

uninstall ()

Performs the uninstallations of the virtualization software

Note: This replaces old `qemu_installer.clean_previous_install()`

unload_modules (`module_list=None`)

Unloads kernel modules

By default, if no module list is explicitly provided, the list on `params` (coming from the configuration file) will be used.

write_version_keyval (`test`)

class `virttest.base_installer.BaseLocalSourceInstaller(mode, name, test=None, params=None)`

Bases: `virttest.base_installer.BaseInstaller`

set_install_params (*test, params*)

class `virttest.base_installer.FailedInstaller` (*msg='Virtualization software install failed'*)
Class used to be returned instead of the installer if a installation fails

Useful to make sure no installer object is used if virt installation fails

load_modules ()

Will refuse to load the kerkel modules as install failed

class `virttest.base_installer.GitRepoInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseLocalSourceInstaller`

get_version ()

set_install_params (*test, params*)

class `virttest.base_installer.KojiInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseInstaller`

Handles virtualization software installation via koji/brew

It uses YUM to install and remove packages.

Change notice: this is not a subclass of YumInstaller anymore. The parameters this class uses are different (koji_tag, koji_pkgs) and the install process runs YUM.

get_version ()

set_install_params (*test, params*)

class `virttest.base_installer.LocalSourceDirInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseLocalSourceInstaller`

Handles software installation by building/installing from a source dir

set_install_params (*test, params*)

class `virttest.base_installer.LocalSourceTarInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseLocalSourceInstaller`

Handles software installation by building/installing from a tarball

set_install_params (*test, params*)

exception `virttest.base_installer.NoModuleError`
Bases: `exceptions.Exception`

Error raised when no suitable modules were found to load

class `virttest.base_installer.NoopInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseInstaller`

Dummy installer that does nothing, useful when software is pre-installed

install ()

class `virttest.base_installer.RemoteSourceTarInstaller` (*mode, name, test=None, params=None*)
Bases: `virttest.base_installer.BaseLocalSourceInstaller`

Handles software installation by building/installing from a remote tarball

set_install_params (*test, params*)

exception `virttest.base_installer.VirtInstallException`

Bases: `exceptions.Exception`

Base virtualization software components installation exception

exception `virttest.base_installer.VirtInstallFailed`

Bases: `virttest.base_installer.VirtInstallException`

Installation of virtualization software components failed

exception `virttest.base_installer.VirtInstallNotInstalled`

Bases: `virttest.base_installer.VirtInstallException`

Virtualization software components are not installed

class `virttest.base_installer.YumInstaller` (*mode, name, test=None, params=None*)

Bases: `virttest.base_installer.BaseInstaller`

Installs virtualization software using YUM

Notice: this class implements a change of behaviour if compared to `qemu_installer.YumInstaller.set_install_params()`. There's no longer a default package list, as each virtualization technology will have a completely different default. This should now be kept at the configuration file only.

For now this class implements support for installing from the configured yum repos only. If the use case of installing from local RPM packages arises, we'll implement that.

get_version ()

set_install_params (*test, params*)

virttest.bootstrap module

`virttest.bootstrap.bootstrap` (*options, interactive=False*)

Common virt test assistant module.

Parameters

- **options** – Command line options.
- **interactive** – Whether to ask for confirmation.

Raises

- **error.CmdError** – If JeOS image failed to uncompress
- **ValueError** – If 7za was not found

`virttest.bootstrap.create_config_files` (*test_dir, shared_dir, interactive, step=None, force_update=False*)

`virttest.bootstrap.create_guest_os_cfg` (*t_type*)

`virttest.bootstrap.create_subtests_cfg` (*t_type*)

`virttest.bootstrap.get_directory_structure` (*rootdir, guest_file*)

`virttest.bootstrap.get_guest_os_info_list` (*test_name, guest_os*)

Returns a list of matching assets compatible with the specified test name and guest OS

`virttest.bootstrap.haz_defcon` (*datadir, imagesdir, isosdir, tmpdir*)

Compare current types from Defaults, or if default, compare on-disk type

`virttest.bootstrap.set_defcon (datadir, imagesdir, isosdir, tmpdir)`

Tries to set datadir default contexts returns True if changed

`virttest.bootstrap.setup (options)`

Run pre tests setup (Uncompress test image(s), such as the JeOS image).

Parameters

- **test_name** – Test name, such as “qemu”.
- **guest_os** – Specify the guest image used for bootstrapping. By default the JeOS image is used.

`virttest.bootstrap.verify_mandatory_programs (t_type, guest_os)`

`virttest.bootstrap.verify_recommended_programs (t_type)`

`virttest.bootstrap.verify_selinux (datadir, imagesdir, isosdir, tmpdir, interactive, selinux=False)`

Verify/Set/Warn about SELinux and default file contexts for testing.

Parameters

- **datadir** – Abs. path to data-directory symlink
- **imagesdir** – Abs. path to data/images directory
- **isosdir** – Abs. path to data/isos directory
- **tmpdir** – Abs. path to virt-test tmp dir
- **interactive** – True if running from console
- **selinux** – Whether setup SELinux contexts for shared/data

`virttest.bootstrap.write_subtests_files (config_file_list, test_type=None, output_file_object,`

Writes a collection of individual subtests config file to one output file

Optionally, for tests that we know their type, write the ‘virt_test_type’ configuration automatically.

virttest.build_helper module

class `virttest.build_helper.GitRepoParamHelper (params, name, destination_dir)`

Bases: `autotest.client.shared.git.GitRepoHelper`

Helps to deal with git repos specified in cartesian config files

This class attempts to make it simple to manage a git repo, by using a naming standard that follows this basic syntax:

`<prefix>_name_<suffix>`

`<prefix>` is always ‘git_repo’ and `<suffix>` sets options for this git repo. Example for repo named foo:

`git_repo_foo_uri = git://git.foo.org/foo.git`
`git_repo_foo_base_uri = /home/user/code/foo`
`git_repo_foo_branch = master`
`git_repo_foo_lbranch = master`
`git_repo_foo_commit = bb5fb8e678aabe286e74c4f2993dc2a9e550b627`

execute ()

class `virttest.build_helper.GnuSourceBuildHelper (source, build_dir, prefix, configure_options=[])`

Bases: `object`

Handles software installation of GNU-like source code

This basically means that the build will go through the classic GNU autotools steps: ./configure, make, make install

configure()

Runs the “configure” script passing appropriate command line options

enable_debug_symbols()

Enables option that leaves debug symbols on compiled software

This makes debugging a lot easier.

execute()

Runs appropriate steps for *building* this source code tree

get_available_configure_options()

Return the list of available options of a GNU like configure script

This will run the “configure” script at the source directory

Returns list of options accepted by configure script

get_configure_command()

Formats configure script with all options set

Returns string with all configure options, including prefix

get_configure_path()

Checks if ‘configure’ exists, if not, return ‘autogen.sh’ as a fallback

include_pkg_config_path()

Adds the current prefix to the list of paths that pkg-config searches

This is currently not optional as there is no observed adverse side effects of enabling this. As the “prefix” is usually only valid during a test run, we believe that having other pkg-config files (*.pc) in either <prefix>/share/pkgconfig or <prefix>/lib/pkgconfig is exactly for the purpose of using them.

Returns None

install()

Runs “make install”

make (*failure_feedback=True*)

Runs a parallel make, falling back to a single job in failure

Parameters **failure_feedback** – return information on build failure by raising the appropriate exceptions

Raise SourceBuildParallelFailed if parallel build fails, or SourceBuildFailed if single job build fails

make_clean()

Runs “make clean”

make_install()

Runs “make install”

make_non_parallel()

Runs “make”, using a single job

make_parallel()

Runs “make” using the correct number of parallel jobs

exception `virttest.build_helper.GnuSourceBuildInvalidSource`

Bases: `exceptions.Exception`

Exception raised when build source dir/file is not valid

class `virttest.build_helper.GnuSourceBuildParamHelper` (*params, name, destination_dir, install_prefix*)

Bases: `virttest.build_helper.GnuSourceBuildHelper`

Helps to deal with gnu_autotools build helper in cartersian config files

This class attempts to make it simple to build source coude, by using a naming standard that follows this basic syntax:

[<git_repo>|<local_src>]_<name>_<option> = value

To pass extra options to the configure script, while building foo from a git repo, set the following variable:

git_repo_foo_configure_options = --enable-feature

class `virttest.build_helper.LinuxKernelBuildHelper` (*params, prefix, source*)

Bases: `object`

Handles Building Linux Kernel.

cp_linux_kernel ()

Copying Linux kernel to target path

execute ()

Runs appropriate steps for *building* this source code tree

install ()

Copying Linux kernel to target path

make (*failure_feedback=True*)

Runs a parallel make

Parameters failure_feedback – return information on build failure by raising the appropriate exceptions

Raise SourceBuildParallelFailed if parallel build fails, or

make_clean ()

Runs “make clean”

make_guest_kernel ()

Runs “make”, using a single job

class `virttest.build_helper.LocalSourceDirHelper` (*source_dir, destination_dir*)

Bases: `object`

Helper class to deal with source code sitting somewhere in the filesystem

execute ()

Copies the source directory to the destination directory

class `virttest.build_helper.LocalSourceDirParamHelper` (*params, name, destination_dir*)

Bases: `virttest.build_helper.LocalSourceDirHelper`

Helps to deal with source dirs specified in cartersian config files

This class attempts to make it simple to manage a source dir, by using a naming standard that follows this basic syntax:

<prefix>_name_<suffix>

<prefix> is always ‘local_src’ and <suffix> sets options for this source dir. Example for source dir named foo:

```
local_src_foo_path = /home/user/foo
```

```
class virttest.build_helper.LocalTarHelper (source, destination_dir)
```

Bases: `object`

Helper class to deal with source code in a local tarball

execute ()

Executes all action this helper is supposed to perform

This is the main entry point method for this class, and all other helper classes.

extract ()

Extracts the tarball into the destination directory

```
class virttest.build_helper.LocalTarParamHelper (params, name, destination_dir)
```

Bases: `virttest.build_helper.LocalTarHelper`

Helps to deal with source tarballs specified in cartersian config files

This class attempts to make it simple to manage a tarball with source code, by using a naming standard that follows this basic syntax:

<prefix>_name_<suffix>

<prefix> is always 'local_tar' and <suffix> sets options for this source tarball. Example for source tarball named foo:

```
local_tar_foo_path = /tmp/foo-1.0.tar.gz
```

```
class virttest.build_helper.PatchHelper (source_dir, patches)
```

Bases: `object`

Helper that encapsulates the patching of source code with patch files

download ()

Copies patch files from remote locations to the source directory

execute ()

Performs all steps necessary to download patches and apply them

patch ()

Patches the source dir with all patch files

```
class virttest.build_helper.PatchParamHelper (params, prefix, source_dir)
```

Bases: `virttest.build_helper.PatchHelper`

Helps to deal with patches specified in cartersian config files

This class attempts to make it simple to patch source code, by using a naming standard that follows this basic syntax:

[<git_repo>|<local_src>|<local_tar>|<remote_tar>]_<name>_patches

<prefix> is either a 'local_src' or 'git_repo', that, together with <name> specify a directory containing source code to receive the patches. That is, for source code coming from git repo foo, patches would be specified as:

```
git_repo_foo_patches = ['http://foo/bar.patch', 'http://foo/baz.patch']
```

And for for patches to be applied on local source code named also foo:

```
local_src_foo_patches = ['http://foo/bar.patch', 'http://foo/baz.patch']
```

```
class virttest.build_helper.RemoteTarHelper (source_uri, destination_dir)
```

Bases: `virttest.build_helper.LocalTarHelper`

Helper that fetches a tarball and extracts it locally

execute()

Executes all action this helper class is supposed to perform

This is the main entry point method for this class, and all other helper classes.

This implementation fetches the remote tar file and then extracts it using the functionality present in the parent class.

class `virttest.build_helper.RemoteTarParamHelper` (*params, name, destination_dir*)

Bases: `virttest.build_helper.RemoteTarHelper`

Helps to deal with remote source tarballs specified in cartesian config

This class attempts to make it simple to manage a tarball with source code, by using a naming standard that follows this basic syntax:

<prefix>_name_<suffix>

<prefix> is always 'local_tar' and <suffix> sets options for this source tarball. Example for source tarball named foo:

`remote_tar_foo_uri = http://foo.org/foo-1.0.tar.gz`

exception `virttest.build_helper.SourceBuildFailed`

Bases: `exceptions.Exception`

Exception raised when building with parallel jobs fails

This serves as feedback for code using `virttest.build_helper.BuildHelper`.

exception `virttest.build_helper.SourceBuildParallelFailed`

Bases: `exceptions.Exception`

Exception raised when building with parallel jobs fails

This serves as feedback for code using `virttest.build_helper.BuildHelper`.

virttest.cartesian_config module

Cartesian configuration format file parser.

Filter syntax:

- , means OR
- .. means AND
- . means IMMEDIATELY-FOLLOWED-BY
- (xx=yy) where xx=VARIANT_NAME and yy=VARIANT_VALUE

Example:

```
qcow2..(guest_os=Fedora).14, RHEL.6..raw..boot, smp2..qcow2..migrate..ide
```

means match all dicts whose names have:

```
(qcow2 AND ((guest_os=Fedora) IMMEDIATELY-FOLLOWED-BY 14)) OR
((RHEL IMMEDIATELY-FOLLOWED-BY 6) AND raw AND boot) OR
(smp2 AND qcow2 AND migrate AND ide)
```

Note:

- `qcow2..Fedora.14` is equivalent to `Fedora.14..qcow2`.
- `qcow2..Fedora.14` is not equivalent to `qcow2..14.Fedora`.

- `ide, scsi` is equivalent to `scsi, ide`.

Filters can be used in 3 ways:

```
only <filter>
no <filter>
<filter>:
```

The last one starts a conditional block.

Formal definition: Regexp come from [python](#). They're not deterministic, but more readable for people. Spaces between terminals and nonterminals are only for better reading of definitions.

The base of the definitions come verbatim as follows:

```
E = {\n, #, :, "-", =, +=, <=, ?=, ?+=, ?<=, !, <, del, @, variants, include, only, no, name, value,
N = {S, DEL, FILTER, FILTER_NAME, FILTER_GROUP, PN_FILTER_GROUP, STAT, VARIANT, VAR-TYPE, VAR-NAME, V

I = I^n | n in N           // indentation from start of line
                           // where n is indentation length.
I = I^n+x | n,x in N       // indentation with shift

start symbol = S
end symbol = eps

S -> I^0+x STATV | eps

I^n    STATV
I^n    STATV

I^n STATV -> I^n STATV \n I^n STATV | I^n STAT | I^n variants VARIANT
I^n STAT -> I^n STAT \n I^n STAT | I^n COMMENT | I^n include INC
I^n STAT -> I^n del DEL | I^n FILTER

DEL -> name \n

I^n STAT -> I^n name = VALUE | I^n name += VALUE | I^n name <= VALUE
I^n STAT -> I^n name ?= VALUE | I^n name ?+= VALUE | I^n name ?<= VALUE

VALUE -> TEXT \n | 'TEXT' \n | "TEXT" \n

COMMENT_BLOCK -> #TEXT | //TEXT
COMMENT -> COMMENT_BLOCK\n
COMMENT -> COMMENT_BLOCK\n

TEXT = [^\n] TEXT           //python format regexp

I^n    variants VAR #comments:           add possibility for comment
I^n+x    VAR-NAME: DEPS
I^n+x+x2    STATV
I^n    VAR-NAME:

IDENTIFIER -> [A-Za-z0-9][A-Za-z0-9_-]*

VARIANT -> VAR COMMENT_BLOCK\n I^n+x VAR-NAME
VAR -> VAR-TYPE: | VAR-TYPE META-DATA: | :           // Named | unnamed variant

VAR-TYPE -> IDENTIFIER
```

```
variants _name_ [xxx] [zzz=yyy] [uuu]:

META-DATA -> [IDENTIFIER] | [IDENTIFIER=TEXT] | META-DATA META-DATA

I^n VAR-NAME -> I^n VAR-NAME \n I^n VAR-NAME | I^n VAR-NAME-N \n I^n+x STATV
VAR-NAME-N -> - @VAR-NAME-F: DEPS | - VAR-NAME-F: DEPS
VAR-NAME-F -> [a-zA-Z0-9\._- ]+ // Python regexp

DEPS -> DEPS-NAME-F | DEPS-NAME-F,DEPS
DEPS-NAME-F -> [a-zA-Z0-9\._- ]+ // Python regexp

INC -> name \n

FILTER_GROUP: STAT
    STAT

I^n STAT -> I^n PN_FILTER_GROUP | I^n ! PN_FILTER_GROUP

PN_FILTER_GROUP -> FILTER_GROUP: \n I^n+x STAT
PN_FILTER_GROUP -> FILTER_GROUP: STAT \n I^n+x STAT

only FILTER_GROUP
no FILTER_GROUP

FILTER -> only FILTER_GROUP \n | no FILTER_GROUP \n

FILTER_GROUP -> FILTER_NAME
FILTER_GROUP -> FILTER_GROUP..FILTER_GROUP
FILTER_GROUP -> FILTER_GROUP,FILTER_GROUP

FILTER_NAME -> FILTER_NAME.FILTER_NAME
FILTER_NAME -> VAR-NAME-F | (VAR-NAME-F=VAR-NAME-F)
```

copyright Red Hat 2008-2013

```
class virttest.cartesian_config.BlockFilter (blocked)
    Bases: object
```

```
    apply_to_dict (d)
```

```
    blocked
```

```
class virttest.cartesian_config.Condition (lfilter, line)
    Bases: virttest.cartesian_config.NoFilter
```

```
    content
```

```
class virttest.cartesian_config.FileReader (filename)
    Bases: virttest.cartesian_config.StrReader
```

```
    Preprocess an input file for easy reading.
```

```
class virttest.cartesian_config.Filter (lfilter)
    Bases: object
```

```
    filter
```

```
    match (ctx, ctx_set)
```

```
    might_match (ctx, ctx_set, descendant_labels)
```



```
class virttest.cartesian_config.LAnd
    Bases: virttest.cartesian_config.Token
    identifier = '..'

class virttest.cartesian_config.LAppend
    Bases: virttest.cartesian_config.LOperators
    apply_to_dict (d)
    identifier = '+='

class virttest.cartesian_config.LApplyPreDict
    Bases: virttest.cartesian_config.LOperators
    apply_to_dict (d)
    identifier = 'apply_pre_dict'
    set_operands (name, value)

class virttest.cartesian_config.LCoc
    Bases: virttest.cartesian_config.Token
    identifier = '.'

class virttest.cartesian_config.LColon
    Bases: virttest.cartesian_config.Token
    identifier = ':'

class virttest.cartesian_config.LComa
    Bases: virttest.cartesian_config.Token
    identifier = ','

class virttest.cartesian_config.LCond
    Bases: virttest.cartesian_config.Token
    identifier = '

class virttest.cartesian_config.LDefault
    Bases: virttest.cartesian_config.Token
    identifier = '@'

class virttest.cartesian_config.LDel
    Bases: virttest.cartesian_config.LOperators
    apply_to_dict (d)
    identifier = 'del'

class virttest.cartesian_config.LDot
    Bases: virttest.cartesian_config.Token
    identifier = '.'

class virttest.cartesian_config.LEndBlock (length)
    Bases: virttest.cartesian_config.LIndent

class virttest.cartesian_config.LEndL
    Bases: virttest.cartesian_config.Token
    identifier = 'endl'
```

```
class virttest.cartesian_config.LIdentifier
    Bases: str

    checkAlpha ()
        Check if string contain only chars

    checkChar (chars)

    checkCharAlpha (chars)
        Check if string contain only chars

    checkCharAlphaNum (chars)
        Check if string contain only chars

    checkCharNumeric (chars)
        Check if string contain only chars

    checkNumbers ()
        Check if string contain only chars

    identifier = 'Identifier re([A-Za-z0-9][A-Za-z0-9_-]*)'

class virttest.cartesian_config.LInclude
    Bases: virttest.cartesian_config.Token

    identifier = 'include'

class virttest.cartesian_config.LIndent (length)
    Bases: virttest.cartesian_config.Token

    identifier = 'indent'

    length

class virttest.cartesian_config.LLBracket
    Bases: virttest.cartesian_config.Token

    identifier = '['

class virttest.cartesian_config.LLRBracket
    Bases: virttest.cartesian_config.Token

    identifier = '('

class virttest.cartesian_config.LNo
    Bases: virttest.cartesian_config.Token

    identifier = 'no'

class virttest.cartesian_config.LNotCond
    Bases: virttest.cartesian_config.Token

    identifier = '!'

class virttest.cartesian_config.LOnly
    Bases: virttest.cartesian_config.Token

    identifier = 'only'

class virttest.cartesian_config.LOperators
    Bases: virttest.cartesian_config.Token

    function = None

    identifier = ''

    name
```

```
    set_operands (name, value)  
    value  
class virttest.cartesian_config.LOr  
    Bases: virttest.cartesian_config.Token  
    identifier = ','  
class virttest.cartesian_config.LPrepend  
    Bases: virttest.cartesian_config.LOperators  
    apply_to_dict (d)  
    identifier = '<='  
class virttest.cartesian_config.LRBracket  
    Bases: virttest.cartesian_config.Token  
    identifier = ']'  
class virttest.cartesian_config.LRRBracket  
    Bases: virttest.cartesian_config.Token  
    identifier = '('  
class virttest.cartesian_config.LRegExpAppend  
    Bases: virttest.cartesian_config.LOperators  
    apply_to_dict (d)  
    identifier = '?+='  
class virttest.cartesian_config.LRegExpPrepend  
    Bases: virttest.cartesian_config.LOperators  
    apply_to_dict (d)  
    identifier = '?<='  
class virttest.cartesian_config.LRegExpSet  
    Bases: virttest.cartesian_config.LOperators  
    apply_to_dict (d)  
    identifier = '?='  
class virttest.cartesian_config.LRegExpStart  
    Bases: virttest.cartesian_config.Token  
    identifier = '${'  
class virttest.cartesian_config.LRegExpStop  
    Bases: virttest.cartesian_config.Token  
    identifier = '}'  
class virttest.cartesian_config.LSet  
    Bases: virttest.cartesian_config.LOperators  
    apply_to_dict (d)  
        Parameters d – Dictionary for apply value  
    identifier = '='  
class virttest.cartesian_config.LString  
    Bases: virttest.cartesian_config.LIdentifier
```

```
    identifier = 'String re(.+)'
class virttest.cartesian_config.LUpdateFileMap
    Bases: virttest.cartesian_config.LOperators
    apply_to_dict (d)
    dest
    identifier = 'update_file_map'
    set_operands (filename, name, dest='_name_map_file')
    shortname
class virttest.cartesian_config.LVariant
    Bases: virttest.cartesian_config.Token
    identifier = '-'
class virttest.cartesian_config.LVariants
    Bases: virttest.cartesian_config.Token
    identifier = 'variants'
class virttest.cartesian_config.LWhite
    Bases: virttest.cartesian_config.LIdentifier
    identifier = 'WhiteSpace re(\\s)'
class virttest.cartesian_config.Label (name, next_name=None)
    Bases: object
    hash_name ()
    hash_val
    hash_var
    hash_variant ()
    long_name
    name
    var_name
class virttest.cartesian_config.Lexer (reader)
    Bases: object
    check_token (token, lType)
    flush_until (end_tokens=None)
    get_lexer ()
    get_next_check (lType)
    get_next_check_nw (lType)
    get_until (end_tokens=None)
    get_until_check (lType, end_tokens=None)
        Read tokens from iterator until get end_tokens or type of token not match lType
    Parameters
        • lType – List of allowed tokens
```

- **end_tokens** – List of tokens for end reading

Returns List of readed tokens.

get_until_gen (*end_tokens=None*)

get_until_no_white (*end_tokens=None*)

Read tokens from iterator until get one of end_tokens and strip LWhite

Parameters **end_tokens** – List of tokens for end reading

Returns List of readed tokens.

match (*line, pos*)

rest_line ()

rest_line_as_LString ()

rest_line_gen ()

rest_line_no_white ()

set_fast ()

set_prev_indent (*prev_indent*)

set_strict ()

exception `virttest.cartesian_config.LexerError` (*msg, line=None, filename=None, linenum=None*)

Bases: `virttest.cartesian_config.ParserError`

exception `virttest.cartesian_config.MissingIncludeError` (*line, filename, linenum*)

Bases: `exceptions.Exception`

class `virttest.cartesian_config.NegativeCondition` (*lfilter, line*)

Bases: `virttest.cartesian_config.OnlyFilter`

content

class `virttest.cartesian_config.NoFilter` (*lfilter, line*)

Bases: `virttest.cartesian_config.NoOnlyFilter`

is_irrelevant (*ctx, ctx_set, descendant_labels*)

might_pass (*failed_ctx, failed_ctx_set, ctx, ctx_set, descendant_labels*)

requires_action (*ctx, ctx_set, descendant_labels*)

class `virttest.cartesian_config.NoOnlyFilter` (*lfilter, line*)

Bases: `virttest.cartesian_config.Filter`

line

class `virttest.cartesian_config.Node`

Bases: `object`

append_to_shortcode

children

content

default

dep

dump (*indent, recurse=False*)

failed_cases
filename
labels
name
q_dict
var_name

class `virttest.cartesian_config.OnlyFilter` (*lfilter, line*)
Bases: `virttest.cartesian_config.NoOnlyFilter`

is_irrelevant (*ctx, ctx_set, descendant_labels*)
might_pass (*failed_ctx, failed_ctx_set, ctx, ctx_set, descendant_labels*)
requires_action (*ctx, ctx_set, descendant_labels*)

class `virttest.cartesian_config.Parser` (*filename=None, defaults=False, expand_defaults=[], debug=False*)
Bases: `object`

assign (*key, value*)
Apply a only filter programatically and keep track of it.
Equivalent to parse a “key = value” line.
Parameters **variant** – String with the variant name.

get_dicts (*node=None, ctx=[], content=[], shortname=[], dep=[]*)
Generate dictionaries from the code parsed so far. This should be called after parsing something.
Returns A dict generator.

no_filter (*variant*)
Apply a only filter programatically and keep track of it.
Equivalent to parse a “no variant” line.
Parameters **variant** – String with the variant name.

only_filter (*variant*)
Apply a only filter programatically and keep track of it.
Equivalent to parse a “only variant” line.
Parameters **variant** – String with the variant name.

parse_file (*filename*)
Parse a file.
Parameters **filename** – Path of the configuration file.

parse_string (*s*)
Parse a string.
Parameters **s** – String to parse.

exception `virttest.cartesian_config.ParserError` (*msg, line=None, filename=None, linenum=None*)
Bases: `exceptions.Exception`

class `virttest.cartesian_config.StrReader` (*s*)
Bases: `object`

Preprocess an input string for easy reading.

get_next_line (*prev_indent*)

Get the next line in the current block.

Parameters *prev_indent* – The indentation level of the previous block.

Returns (line, indent, linenum), where indent is the line's indentation level. If no line is available, (None, -1, -1) is returned.

set_next_line (*line, indent, linenum*)

Make the next call to `get_next_line()` return the given line instead of the real next line.

class `virttest.cartesian_config.Token`

Bases: `object`

identifier = ''

`virttest.cartesian_config.apply_predict` (*lexer, node, pre_dict*)

`virttest.cartesian_config.cmd_tokens` (*tokens1, tokens2*)

`virttest.cartesian_config.compare_string` (*str1, str2*)

Compare two int string and return -1, 0, 1. It can compare two memory value even in suffix

Parameters

- **str1** – The first string
- **str2** – The second string

Return Return -1, when `str1 < str2` 0, when `str1 = str2` 1, when `str1 > str2`

`virttest.cartesian_config.convert_data_size` (*size, default_suffix='B'*)

Convert data size from human readable units to an int of arbitrary size.

Parameters

- **size** – Human readable data size representation (string).
- **default_suffix** – Default suffix used to represent data.

Returns Int with data size in the appropriate order of magnitude.

`virttest.cartesian_config.next_nw` (*gener*)

`virttest.cartesian_config.parse_filter` (*lexer, tokens*)

Returns Parsed filter

`virttest.cartesian_config.postfix_parse` (*dic*)

`virttest.cartesian_config.print_dicts` (*options, dicts*)

`virttest.cartesian_config.print_dicts_default` (*options, dicts*)

Print dictionaries in the default mode

`virttest.cartesian_config.print_dicts_repr` (*options, dicts*)

virttest.cartesian_config_unittest module

class `virttest.cartesian_config_unittest.CartesianConfigTest` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

testComplicatedFilter ()

testCondition ()

```
testDefaults()
testDel()
testError1()
testFilterMixing()
testHugeTest1()
testMissingInclude()
testNameVariant()
testNegativeCondition()
testSimpleVariant()
testSyntaxErrors()
testVariableAssignment()
```

virttest.ceph module

CEPH Support This file has the functions that helps * To create rbd pool * To map/unmap rbd pool * To mount/umount cephfs to localhost * To return rbd uri which can be used as disk image file path.

exception `virttest.ceph.CephError`
Bases: `exceptions.Exception`

virttest.common module

`virttest.common.load_setup_modules(client_dir)`

virttest.data_dir module

Library used to provide the appropriate data dir for virt test.

exception `virttest.data_dir.MissingDepsDirError`
Bases: `exceptions.Exception`

class `virttest.data_dir.SubdirGlobList(basedir, globstr, filterlist=None)`
Bases: `virttest.data_dir.SubdirList`

List of all files matching glob in all non-hidden basedir subdirectories

class `virttest.data_dir.SubdirList(basedir, filterlist=None)`
Bases: `list`

List of all non-hidden subdirectories beneath basedir

exception `virttest.data_dir.UnknownBackendError(backend)`
Bases: `exceptions.Exception`

`virttest.data_dir.clean_tmp_files()`

`virttest.data_dir.get_backend_cfg_path(backend_type, cfg_basename)`

`virttest.data_dir.get_backend_dir(backend_type)`

`virttest.data_dir.get_backing_data_dir()`

`virttest.data_dir.get_data_dir()`


```
virttest.data_dir.get_deps_dir(target=None)
```

For a given test provider, report the appropriate deps dir.

The little inspect trick is used to avoid callers having to do `sys.modules[]` tricks themselves.

Parameters `target` – File we want in deps folder. Will return the path to the target if set and available. Or will only return the path to dep folder.

```
virttest.data_dir.get_download_dir()
```

```
virttest.data_dir.get_root_dir()
```

```
virttest.data_dir.get_test_provider_dir(provider)
```

Return a specific test providers dir, inside the base dir.

```
virttest.data_dir.get_test_providers_dir()
```

Return the base test providers dir (at the moment, test-providers.d).

```
virttest.data_dir.get_tmp_dir()
```

```
virttest.data_dir.set_backing_data_dir(backing_data_dir)
```

virttest.defaults module

```
virttest.defaults.get_default_guest_os_info()
```

Gets the default asset and variant information depending on host OS

virttest.element_path module

```
class virttest.element_path.Path(path)
```

```
    find(element)
```

```
    findall(element)
```

```
    findtext(element, default=None)
```

```
virttest.element_path.find(element, path)
```

```
virttest.element_path.findall(element, path)
```

```
virttest.element_path.findtext(element, path, default=None)
```

```
class virttest.element_path.xpath_descendant_or_self
```

```
virttest.element_path.xpath_tokenizer()
```

`findall(string[, pos[, endpos]])` → list. Return a list of all non-overlapping matches of pattern in string.

virttest.element_tree module

```
virttest.element_tree.Comment(text=None)
```

```
virttest.element_tree.dump(elem)
```

```
virttest.element_tree.Element(tag, attrib={}, **extra)
```

```
class virttest.element_tree.ElementTree(element=None, file=None)
```

Bases: `object`

```
    find(path)
```

```
findall (path)
findtext (path, default=None)
getiterator (tag=None)
getroot ()
parse (source, parser=None)
write (file, encoding='us-ascii')
virttest.element_tree.fromstring (text)
virttest.element_tree.iselement (element)
class virttest.element_tree.iterparse (source, events=None)
    Bases: object
    next ()
virttest.element_tree.parse (source, parser=None)
virttest.element_tree.PI (target, text=None)
virttest.element_tree.ProcessingInstruction (target, text=None)
class virttest.element_tree.QName (text_or_uri, tag=None)
    Bases: object
virttest.element_tree.SubElement (parent, tag, attrib={}, text=None, **extra)
virttest.element_tree.tostring (element, encoding=None)
class virttest.element_tree.TreeBuilder (element_factory=None)
    Bases: object
    close ()
    data (data)
    end (tag)
    start (tag, attrs)
virttest.element_tree.XML (text)
virttest.element_tree.XMLParser
    alias of XMLTreeBuilder
class virttest.element_tree.XMLTreeBuilder (html=0, target=None)
    Bases: object
    close ()
    doctype (name, pubid, system)
    feed (data)
```

virttest.env_process module

```
virttest.env_process.postprocess_image (test, params, image_name,
                                           vm_process_status=None)
    Postprocess a single QEMU image according to the instructions in params.
```

Parameters

- **test** – An Autotest test object.
- **params** – A dict containing image postprocessing parameters.
- **vm_process_status** – (optional) vm process status like running, dead or None for no vm exist.

`virttest.env_process.postprocess_on_error(test, params, env)`

Perform postprocessing operations required only if the test failed.

Parameters

- **test** – An Autotest test object.
- **params** – A dict containing all VM and image parameters.
- **env** – The environment (a dict-like object).

`virttest.env_process.postprocess_vm(test, params, env, name)`

Postprocess a single VM object according to the instructions in params. Kill the VM if requested and get a screendump.

Parameters

- **test** – An Autotest test object.
- **params** – A dict containing VM postprocessing parameters.
- **env** – The environment (a dict-like object).
- **name** – The name of the VM object.

`virttest.env_process.preprocess_image(test, params, image_name, vm_process_status=None)`

Preprocess a single QEMU image according to the instructions in params.

Parameters

- **test** – Autotest test object.
- **params** – A dict containing image preprocessing parameters.
- **vm_process_status** – This is needed in `postprocess_image`. Add it here only for keep it work with `process_images()`

Note Currently this function just creates an image if requested.

`virttest.env_process.preprocess_vm(test, params, env, name)`

Preprocess a single VM object according to the instructions in params. Start the VM if requested and get a screendump.

Parameters

- **test** – An Autotest test object.
- **params** – A dict containing VM preprocessing parameters.
- **env** – The environment (a dict-like object).
- **name** – The name of the VM object.

`virttest.env_process.process(test, params, env, image_func, vm_func, vm_first=False)`

Pre- or post-process VMs and images according to the instructions in params. Call `image_func` for each image listed in params and `vm_func` for each VM.

Parameters

- **test** – An Autotest test object.

- **params** – A dict containing all VM and image parameters.
- **env** – The environment (a dict-like object).
- **image_func** – A function to call for each image.
- **vm_func** – A function to call for each VM.
- **vm_first** – Call vm_func first or not.

`virttest.env_process.process_command(test, params, env, command, command_timeout, command_noncritical)`

Pre- or post- custom commands to be executed before/after a test is run

Parameters

- **test** – An Autotest test object.
- **params** – A dict containing all VM and image parameters.
- **env** – The environment (a dict-like object).
- **command** – Command to be run.
- **command_timeout** – Timeout for command execution.
- **command_noncritical** – If True test will not fail if command fails.

`virttest.env_process.process_images(image_func, test, params, vm_process_status=None)`

Wrapper which chooses the best way to process images.

Parameters

- **image_func** – Process function
- **test** – An Autotest test object.
- **params** – A dict containing all VM and image parameters.
- **vm_process_status** – (optional) vm process status like running, dead or None for no vm exist.

`virttest.env_process.store_vm_register(vm, log_filename, append=False)`

Store the register information of vm into a log file

Parameters

- **vm** (*vm object*) – VM object
- **log_filename** (*string*) – log file name
- **append** (*bool*) – Add the log to the end of the log file or not

Returns Store the vm register information to log file or not

Return type `bool`

virttest.funcatexit module

funcatexit.py - allow programmer to define multiple exit functions to be executed upon normal cases termination. Can be used for the environment clean up functions. The basic idea is like atexit from python libs.

`virttest.funcatexit.register(env, test_type, func, *targs, **kargs)`

Register a function to be executed upon case termination. func is returned to facilitate usage as a decorator.

param env: the global objects used by tests param test_type: test type mark for exit functions param func: function to be called at exit param targs: optional arguments to pass to func param kargs: optional keyword arguments to pass to func

`virttest.funcatexit.run_exitfuncs (env, test_type)`

Run any registered exit functions. exithandlers is traversed in reverse order so functions are executed last in, first out.

param env: the global objects used by tests param test_type: test type mark for exit functions

`virttest.funcatexit.unregister (env, test_type, func, *targs, **kargs)`

Unregister a function to be executed upon case termination. func is returned to facilitate usage as a decorator.

param env: the global objects used by tests param test_type: test type mark for exit functions param func: function to be called at exit param targs: optional arguments to pass to func param kargs: optional keyword arguments to pass to func

virttest.gluster module

GlusterFS Support This file has the functions that helps * To create/check gluster volume. * To start/check gluster services. * To create gluster uri which can be used as disk image file path.

exception `virttest.gluster.GlusterBrickError (error_mgs)`

Bases: `virttest.gluster.GlusterError`

exception `virttest.gluster.GlusterError`

Bases: `exceptions.Exception`

`virttest.gluster.add_rpc_insecure (filepath)`

Allow glusterd RPC authority insecure

`virttest.gluster.file_exists (params, filename_path)`

`virttest.gluster.get_image_filename (params, image_name, image_format)`

Form the image file name using gluster uri

`virttest.gluster.gluster_brick_create (brick_path, force=False)`

Creates brick

`virttest.gluster.gluster_brick_delete (brick_path)`

Creates brick

`virttest.gluster.glusterfs_mount (g_uri, mount_point)`

Mount gluster volume to mountpoint.

Parameters `g_uri (str)` – stripped gluster uri from `create_gluster_uri(.., True)`

virttest.guest_agent module

Interfaces to the virt agent.

copyright 2008-2012 Red Hat Inc.

class `virttest.guest_agent.QemuAgent (vm, name, serial_type, serial_filename, get_supported_cmds=False, suppress_exceptions=False)`

Bases: `virttest.qemu_monitor.Monitor`

Wraps qemu guest agent commands.

CMD_TIMEOUT = 20

FSFREEZE_STATUS_FROZEN = 'frozen'

FSFREEZE_STATUS_THAWED = 'thawed'

PROMPT_TIMEOUT = 20

READ_OBJECTS_TIMEOUT = 5

RESPONSE_TIMEOUT = 20

SERIAL_TYPE_ISA = 'isa'

SERIAL_TYPE_VIRTIO = 'virtio'

SHUTDOWN_MODE_HALT = 'halt'

SHUTDOWN_MODE_POWERDOWN = 'powerdown'

SHUTDOWN_MODE_REBOOT = 'reboot'

SUPPORTED_SERIAL_TYPE = ['virtio', 'isa']

SUSPEND_MODE_DISK = 'disk'

SUSPEND_MODE_HYBRID = 'hybrid'

SUSPEND_MODE_RAM = 'ram'

cmd (*cmd*, *args*=None, *timeout*=20, *debug*=True, *success_resp*=True)
Send a guest agent command and return the response if *success_resp*.

Parameters

- **cmd** – Command to send
- **args** – A dict containing command arguments, or None
- **timeout** – Time duration to wait for response
- **debug** – Whether to print the commands being sent and responses
- **fd** – file object or file descriptor to pass

Returns The response received

Raises

- **VAgentLockError** – Raised if the lock cannot be acquired
- **VAgentSocketError** – Raised if a socket error occurs
- **VAgentProtocolError** – Raised if no response is received
- **VAgentCmdError** – Raised if the response is an error message

cmd_obj (*obj*, *timeout*=20)

Transform a Python object to JSON, send the resulting string to the guest agent, and return the response. Unlike **cmd()**, return the raw response dict without performing any checks on it.

Parameters

- **obj** – The object to send
- **timeout** – Time duration to wait for response

Returns The response received

Raises

- **VAgentLockError** – Raised if the lock cannot be acquired
- **VAgentSocketError** – Raised if a socket error occurs

- **VAgentProtocolError** – Raised if no response is received

cmd_raw (*data*, *timeout=20*, *success_resp=True*)

Send a raw string to the guest agent and return the response. Unlike `cmd()`, return the raw response dict without performing any checks on it.

Parameters

- **data** – The data to send
- **timeout** – Time duration to wait for response

Returns The response received

Raises

- **VAgentLockError** – Raised if the lock cannot be acquired
- **VAgentSocketError** – Raised if a socket error occurs
- **VAgentProtocolError** – Raised if no response is received

fsfreeze (**args*, ***kwargs*)

Freeze File system on guest.

Parameters **check_status** – Force this function to check the fsfreeze status before/after sending cmd.

Returns Frozen FS number if cmd succeed, -1 if guest agent doesn't support fsfreeze cmd.

fsthaw (**args*, ***kwargs*)

Thaw File system on guest.

Parameters **check_status** – Force this function to check the fsfreeze status before/after sending cmd.

Returns Thaw FS number if cmd succeed, -1 if guest agent doesn't support fsfreeze cmd.

get_fsfreeze_status ()

Get guest 'fsfreeze' status. The status could be 'frozen' or 'thawed'.

shutdown (**args*, ***kwargs*)

Send "guest-shutdown", this cmd would not return any response.

Parameters **mode** – Speicfy shutdown mode, now qemu guest agent supports 'powerdown', 'reboot', 'halt' 3 modes.

Returns True if shutdown cmd is sent successfully, False if 'shutdown' is unsupported.

suspend (**args*, ***kwargs*)

This function tries to execute the scripts provided by the pm-utils package via guest agent interface. If it's not available, the suspend operation will be performed by manually writing to a sysfs file.

Notes:

1. For the best results it's strongly recommended to have the pm-utils package installed in the guest.
2. The ram and 'hybrid' mode require QEMU to support the `system_wakeup` command. Thus, it's required to query QEMU for the presence of the `system_wakeup` command before issuing guest agent command.

Parameters **mode** – Specify suspend mode, could be one of disk, ram, hybrid.

Returns True if shutdown cmd is sent successfully, False if suspend is unsupported.

Raises **VAgentSuspendUnknownModeError** – Raise if mode is not supported.

sync (*args, **kwargs)

Sync guest agent with cmd 'guest-sync'.

verify_fsfreeze_status (expected)

Verify the guest agent fsfreeze status is same as expected, if not, raise a VAgentFreezeStatusError.

Parameters **expected** – The expected status.

Raises **VAgentFreezeStatusError** – Raise if the guest fsfreeze status is unexpected.

verify_responsive ()

Make sure the guest agent is responsive by sending a command.

exception virttest.guest_agent.VAgentCmdError (cmd, args, data)

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentConnectError

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentError

Bases: *virttest.qemu_monitor.MonitorError*

exception virttest.guest_agent.VAgentFreezeStatusError (vm_name, status, expected)

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentLockError

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentNotSupportedError

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentProtocolError

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentSocketError (msg, e)

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentSuspendError

Bases: *virttest.guest_agent.VAgentError*

exception virttest.guest_agent.VAgentSuspendUnknownModeError (mode)

Bases: *virttest.guest_agent.VAgentSuspendError*

exception virttest.guest_agent.VAgentSyncError (vm_name)

Bases: *virttest.guest_agent.VAgentError*

virttest.http_server module

class virttest.http_server.HTTPRequestHandler (request, client_address, server)

Bases: *SimpleHTTPServer.SimpleHTTPRequestHandler*

address_string ()

This HTTP server does not care about name resolution for the requests

The first reason is that most of the times our clients are going to be virtual machines without a proper name resolution setup. Also, by not resolving names, we should be a bit faster and be resilient about misconfigured or resilient name servers.

copyfile_range (source_file, output_file, range_begin, range_end)

Copies a range of a file to destination.

do_GET ()

Serve a GET request.

log_message (*fmt*, **args*)

parse_header_byte_range ()

send_head_range (*range_begin*, *range_end*)

translate_path (*path*)

Translate a /-separated PATH to the local filename syntax.

Components that mean special things to the local file system (e.g. drive or directory names) are ignored. (XXX They should probably be diagnosed.)

`virttest.http_server.http_server (port=8000, cwd=None, terminate_callable=None)`

virttest.installer module

Installer classes are responsible for building and installing virtualization specific software components. This is the main entry point for tests that wish to install virtualization software components.

The most common use case is to simply call `make_installer()` inside your tests.

class `virttest.installer.InstallerRegistry` (***kwargs*)

Bases: `dict`

Holds information on known installer classes

This class is used to create a single instance, named `INSTALLER_REGISTRY`, that will hold all information on known installer types.

For registering a new installer class, use the `register()` method. If the virt type is not set explicitly, it will be set to 'base'. Example:

```
>>> INSTALLER_REGISTRY.register('yum', base_installer.YumInstaller)
```

If you want to register a virt specific installer class, set the virt (third) param:

```
>>> INSTALLER_REGISTRY.register('yum', qemu_installer.YumInstaller, 'qemu')
```

For getting a installer class, use the `get_installer()` method. This method has a fallback option 'get_default_virt' that will return a generic virt installer if set to true.

DEFAULT_VIRT_NAME = 'base'

get_installer (*mode*, *virt=None*, *get_default_virt=False*)

Gets a installer class that should be able to install the virt software

Always try to use classes that are specific to the virtualization technology that is being tested. If you have confidence that the installation is rather trivial and does not require custom steps, you may be able to get away with a base class (by setting `get_default_virt` to True).

get_modes (*virt=None*)

Returns a list of all registered installer modes

register (*mode*, *klass*, *virt=None*)

Register a class as responsible for installing virt software components

If virt is not set, it will assume a default of 'base'.

`virttest.installer.make_installer (fullname, params, test=None)`

Installer factory: returns a new installer for the chosen mode and vm type

This is the main entry point for acquiring an installer. Tests, such as the build test, should use this function.

Param priority evaluation order is 'install_mode', then 'mode'. For virt type, 'vm_type' is consulted.

Parameters

- **fullname** – the full name of instance, eg: git_repo_foo
- **params** – dictionary with parameters generated from cartersian config
- **test** – the test instance

`virttest.installer.run_installers(params, test=None)`

Runs the installation routines for all installers, one at a time

This is usually the main entry point for tests

virttest.installer_unittest module

`class virttest.installer_unittest.installer_test (methodName='runTest')`

Bases: `unittest.case.TestCase`

`setUp()`

`test_make_installer()`

`test_register_get_installer()`

`test_register_get_installer_default()`

virttest.iscsi module

Basic iscsi support for Linux host with the help of commands iscsiadm and tgtadm.

This include the basic operates such as login and get device name by target name. And it can support the real iscsi access and emulated iscsi in localhost then access it.

`class virttest.iscsi.Iscsi`

Bases: `object`

Basic iSCSI support class, which will handle the emulated iscsi export and access to both real iscsi and emulated iscsi device.

The class support different kinds of iSCSI backend (TGT and LIO), and return ISCSI instance.

`static create_iSCSI (params, root_dir='/tmp')`

`class virttest.iscsi.IscsiLIO (params, root_dir)`

Bases: `virttest.iscsi._IscsiComm`

iscsi support class for LIO backend used in RHEL7.

`delete_target()`

Delete target from host.

`export_target()`

Export target in localhost for emulated iscsi

`get_target_id()`

Get target id from image name.

`set_chap_acls_target()`

set CHAP(acls) authentication on a target. it will require authentication before an initiator is allowed to log in and access devices.

notice: Individual ACL entries override common TPG Authentication, which can be set by `set_chap_auth_target()`.

set_chap_auth_target ()
set up authentication information for every single initiator, which provides the capability to define common login information for all Endpoints in a TPG

class virttest.iscsi.IscsiTGT (params, root_dir)
Bases: `virttest.iscsi._IscsiComm`
iscsi support TGT backend used in RHEL6.

add_chap_account ()
Add CHAP authentication account

delete_chap_account ()
Delete the CHAP authentication account

delete_target ()
Delete target from host.

export_target ()
Export target in localhost for emulated iscsi

get_chap_accounts ()
Get all CHAP authentication accounts

get_target_account_info ()
Get the target account information

get_target_id ()
Get target id from image name. Only works for emulated iscsi device

set_chap_auth_target ()
Set CHAP authentication on a target, it will require authentication before an initiator is allowed to log in and access devices.

virttest.iscsi.iscsi_discover (portal_ip)
Query from iscsi server for available targets
Parameters **portal_ip** – Ip for iscsi server

virttest.iscsi.iscsi_get_nodes ()
Get the iscsi nodes

virttest.iscsi.iscsi_get_sessions ()
Get the iscsi sessions activated

virttest.iscsi.iscsi_login (target_name, portal)
Login to a target with the target name
Parameters **target_name** – Name of the target
Params **portal** Hostname/Ip for iscsi server

virttest.iscsi.iscsi_logout (target_name=None)
Logout from a target. If the target name is not set then logout all targets.
Params **target_name** Name of the target.

virttest.iscsi.iscsi_node_del (target_name=None)
Delete target node record, if the target name is not set then delete all target node records.
Params **target_name** Name of the target.

virttest.iscsi_unittest module

```
class virttest.iscsi_unittest.iscsi_test (methodName='runTest')
    Bases: unittest.case.TestCase

    setUp()

    setup_stubs_add_chap_account (iscsi_obj)

    setup_stubs_cleanup (iscsi_obj, fname='')

    setup_stubs_delete_chap_account (iscsi_obj)

    setup_stubs_export_target (iscsi_obj)

    setup_stubs_get_chap_accounts (result='')

    setup_stubs_get_device_name (iscsi_obj)

    setup_stubs_get_target_account_info()

    setup_stubs_get_target_id()

    setup_stubs_init()

    setup_stubs_logged_in (result='')

    setup_stubs_login (iscsi_obj)

    setup_stubs_portal_visible (iscsi_obj, result='')

    setup_stubs_set_chap_auth_initiator (iscsi_obj)

    setup_stubs_set_chap_auth_target (iscsi_obj)

    setup_stubs_set_initiatorName (iscsi_obj)

    tearDown()

    test_iscsi_get_device_name()

    test_iscsi_login()

    test_iscsi_target_id()

    test_iscsi_visible()
```

virttest.libvirt_network_unittest module

Unit tests for Manipulator classes in libvirt_xml module.

```
class virttest.libvirt_network_unittest.NetworkTestBase (methodName='runTest')
    Bases: unittest.case.TestCase

    Base class for NetworkXML test providing fake virsh commands.

    setUp()

class virttest.libvirt_network_unittest.NetworkXMLTest (methodName='runTest')
    Bases: virttest.libvirt_network_unittest.NetworkTestBase

    Unit test class for manipulator methods in NetworkXML class.

    test_sync_and_state_dict()
        Unit test for sync and state_dict methods of NetworkXML class.

        Traverse all possible state and call sync using the state.
```

virttest.libvirt_storage module

Classes and functions to handle block/disk images for libvirt.

This exports:

- two functions for get image/blkdebug filename
- class for image operates and basic parameters
- class for storage pool operations

```
class virttest.libvirt_storage.PoolVolume(pool_name,
                                           virsh_instance=<module
                                           'virttest.virsh'
                                           from
                                           '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                           test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `object`

Volume Manager for libvirt storage pool.

clone_volume (old_name, new_name)
Clone a volume

create_volume (name, capability, allocation=None, fmt=None)
Create a volume in pool.

delete_volume (name)
Remove a volume.

list_volumes ()
Return a dict include volumes' name(key) and path(value).

volume_exists (name)

volume_info (name)
Get volume's information with command vol-info.

```
class virttest.libvirt_storage.QemuImg(params, root_dir, tag)
```

Bases: `virttest.storage.QemuImg`

libvirt class for handling operations of disk/block images.

check_image (params, root_dir)
Check an image using the appropriate tools for each virt backend.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.

Note params should contain:

Raises VMImageCheckError – In case qemu-img check fails on the image.

commit ()
Commit image to it's base file

convert (params, root_dir)
Convert image

Parameters

- **params** – A dict
- **root_dir** – dir for save the convert image

Note params should contain:

create (*params*)

Create an image.

Parameters **params** – Dictionary containing the test parameters.

Note params should contain:

rebase (*params*)

Rebase image

Parameters **params** – A dict

Note params should contain:

remove ()

Remove an image file.

Note params should contain:

snapshot_create ()

Create a snapshot image.

Note params should contain:

snapshot_del (*blkdebug_cfg*='')

Delete a snapshot image.

Parameters **blkdebug_cfg** – The configure file of blkdebug

Note params should contain: **snapshot_image_name** – the name of snapshot image file

```
class virttest.libvirt_storage.StoragePool (virsh_instance=<module 'virttest.virsh' from
                                             '/home/docs/checkouts/readthedocs.org/user_builds/virt-
                                             test/checkouts/latest/virttest/virsh.pyc'>)
```

Bases: `object`

Pool Manager for libvirt storage with virsh commands

build_pool (*name*)

Build pool.

define_dir_pool (*name*, *target_path*)

Define a directory type pool.

define_disk_pool (*name*, *block_device*, *target_path*)

Define a disk type pool.

define_fs_pool (*name*, *block_device*, *target_path*)

Define a filesystem type pool.

define_iscsi_pool (*name*, *source_host*, *source_dev*, *target_path*)

Define a iscsi type pool.

define_lvm_pool (*name*, *block_device*, *vg_name*, *target_path*)

Define a lvm type pool.

define_netfs_pool (*name*, *source_host*, *source_path*, *target_path*)

Define a netfs type pool.

define_rbd_pool (*name*, *source_host*, *source_name*, *extra*='')

Define a rbd type pool.

delete_pool (*name*)

Destroy and Delete a pool if it exists on given libvirt

It's reasonable to delete a pool by calling pool-delete. However, due to pool-delete operation is non-recoverable. Redhat suggests to achieve this objective by virsh, 1) virsh pool-destroy pool-name 2) virsh pool-undefine pool-name

Please refer to the following URI for more details. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/chap-Virtualization_Administration_Guide-Storage_Pools-Storage_Pools.html#delete-ded-disk-storage-pool

destroy_pool (*name*)

Destroy pool if it is active.

get_pool_uuid (*name*)

Get pool's uuid.

Returns Pool uuid.

is_pool_active (*name*)

Check whether pool is active on given libvirt

is_pool_persistent (*name*)

Check whether pool is persistent

list_pools ()

Return a dict include pools' information with structure: pool_name ==> pool_details(a dict: feature ==> value)

pool_exists (*name*)

Check whether pool exists on given libvirt

pool_info (*name*)

Get pool's information.

Returns A dict include pool's information: Name ==> value UUID ==> value ...

pool_state (*name*)

Get pool's state.

Returns active/inactive, and None when something wrong.

set_pool_autostart (*name*, *extra*='')

Set given pool as autostart

start_pool (*name*)

Start pool if it is inactive.

virttest.libvirt_storage_unittest module

```
class virttest.libvirt_storage_unittest.ExistPoolTest (methodName='runTest')
```

Bases: `virttest.libvirt_storage_unittest.PoolTestBase`

```
test_exist_pool()
```

```
class virttest.libvirt_storage_unittest.NewPoolTest (methodName='runTest')
```

Bases: `virttest.libvirt_storage_unittest.PoolTestBase`

```
tearDown()
```

```
test_dir_pool()
```

```
class virttest.libvirt_storage_unittest.NotExpectedPoolTest (methodName='runTest')
    Bases: virttest.libvirt_storage_unittest.PoolTestBase

    test_not_exist_pool ()

class virttest.libvirt_storage_unittest.PoolTestBase (methodName='runTest')
    Bases: unittest.case.TestCase

    setUp ()
```

virttest.libvirt_vm module

Utility classes and functions to handle Virtual Machine creation using libvirt.

copyright 2011 Red Hat Inc.

```
class virttest.libvirt_vm.VM (name, params, root_dir, address_cache, state=None)
    Bases: virttest.virt_vm.BaseVM
```

This class handles all basic VM operations for libvirt.

```
activate_nic (nic_index_or_name)
```

```
attach_disk (source, target=None, prefix='vd', extra='', ignore_status=False, debug=False)
    Attach a disk to VM and return the target device name.
```

Parameters

- **source** – source of disk device
- **target** – target of disk device, None for automatic assignment.
- **prefix** – disk device prefix.
- **extra** – additional arguments to command

Returns target device name if succeeded

```
attach_interface (option='', ignore_status=False, debug=False)
    Attach a NIC to VM.
```

```
backup_xml (active=False)
    Backup the guest's xmlfile.
```

```
cleanup_serial_console ()
    Close serial console and associated log file
```

```
cleanup_swap ()
    Cleanup environment changed by create_swap_partition() or create_swap_file().
```

```
clone (name=None, params=None, root_dir=None, address_cache=None, copy_state=False)
    Return a clone of the VM object with optionally modified parameters. The clone is initially not alive and
    needs to be started using create(). Any parameters not passed to this function are copied from the source
    VM.
```

Parameters

- **name** – Optional new VM name
- **params** – Optional new VM creation parameters
- **root_dir** – Optional new base directory for relative filenames
- **address_cache** – A dict that maps MAC addresses to IP addresses

- **copy_state** – If True, copy the original VM’s state to the clone. Mainly useful for `make_create_command()`.

create (*args, **kwargs)

Start the VM by running a qemu command. All parameters are optional. If name, params or root_dir are not supplied, the respective values stored as class attributes are used.

Parameters

- **name** – The name of the object
- **params** – A dict containing VM params
- **root_dir** – Base directory for relative filenames
- **migration_mode** – If supplied, start VM for incoming migration using this protocol (either ‘tcp’, ‘unix’ or ‘exec’)
- **migration_exec_cmd** – Command to embed in ‘-incoming “exec: ...”’ (e.g. ‘gzip -c -d filename’) if migration_mode is ‘exec’
- **mac_source** – A VM object from which to copy MAC addresses. If not specified, new addresses will be generated.

Raises

- **VMCreateError** – If qemu terminates unexpectedly
- **VMKVMInitError** – If KVM initialization fails
- **VMHugePageError** – If hugepage initialization fails
- **VMImageMissingError** – If a CD image is missing
- **VMHashMismatchError** – If a CD image hash has doesn’t match the expected hash
- **VMBadPATypeError** – If an unsupported PCI assignment type is requested
- **VMPAError** – If no PCI assignable devices could be assigned

create_serial_console ()

Establish a session with the serial console.

The libvirt version uses virsh console to manage it.

create_swap_file (swapfile=’/swapfile’)

Make a swap file and active it through a session.

A `cleanup_swap()` should be call after use to clean up the environment changed.

Parameters **swapfile** – Swap file path in VM to be created.

create_swap_partition (swap_path=None)

Make a swap partition and active it.

A `cleanup_swap()` should be call after use to clean up the environment changed.

Parameters **swap_path** – Swap image path.

deactivate_nic (nic_index_or_name)

define (xml_file)

Define the VM.

destroy (gracefully=True, free_mac_addresses=True)

Destroy the VM.

If `gracefully` is `True`, first attempt to shutdown the VM with a shell command. If that fails, send `SIGKILL` to the `qemu` process.

Parameters

- **`gracefully`** – If `True`, an attempt will be made to end the VM using a shell command before trying to end the `qemu` process with a ‘quit’ or a kill signal.
- **`free_mac_addresses`** – If `vm` is undefined with `libvirt`, also release/reset associated mac address

`detach_disk` (*target*, *extra*=‘’, *ignore_status*=*False*, *debug*=*False*)

Detach a disk from VM.

Parameters

- **`target`** – target of disk device need to be detached.
- **`extra`** – additional arguments to command

`detach_interface` (*option*=‘’, *ignore_status*=*False*, *debug*=*False*)

Detach a NIC from VM.

`dominfo` ()

Return a dict include `vm`’s information.

`domjobabort` ()

Abort job for `vm`.

`dump` (*path*, *option*=‘’)

Dump self to path.

Raise `error.TestFail` if dump fail.

`exists` ()

Return `True` if VM exists.

`get_blk_devices` ()

Get `vm`’s block devices.

Return a dict include all devices detail info. example: {`target`: {`‘type’`: value, `‘device’`: value, `‘source’`: value}}

`get_cpu_topology_in_cmdline` ()

Return the VM’s cpu topology in VM cmdline.

Returns A dirt of cpu topology

`get_cpu_topology_in_vm` ()

`get_device_details` (*device_target*)

`get_device_size` (*device_target*)

`get_disk_devices` ()

Get `vm`’s disk type block devices.

`get_disks` (*diskname*=*None*)

Get disks in `vm`.

Parameters **`diskname`** – Specify disk to be listed, used for checking given disk.

`get_first_disk_devices` ()

Get `vm`’s first disk type block devices.

get_id()

Return VM's ID.

get_ifname (*nic_index=0*)

get_interface_mac (*interface*)

Get mac address of interface by given name.

get_interfaces ()

Get available interfaces in vm.

get_job_type ()

get_max_mem ()

Get vm's maximum memory(kilobytes).

get_pci_devices (*device_str=None*)

Get PCI devices in vm according to given device character.

Parameters **device_str** – a string to identify device.

get_pid ()

Return the VM's PID.

Returns int with PID. If VM is not alive, returns None.

get_serial_console_filename (*name*)

Return the serial console filename.

Parameters **name** – The serial port name.

get_serial_console_filenames ()

Return a list of all serial console filenames (as specified in the VM's params).

get_shared_meminfo ()

Returns the VM's shared memory information.

Returns Shared memory used by VM (MB)

get_shell_pid ()

Return the PID of the parent shell process.

Note This works under the assumption that `self.process.get_pid()` returns the PID of the parent shell process.

get_used_mem ()

Get vm's current memory(kilobytes).

get_uuid ()

Return VM's UUID.

get_vcpus_pid ()

Return the vcpu's pid for a given VM.

Returns list of PID of vcpus of a VM.

get_virsh_mac_address (*nic_index=0*)

Get the MAC of this VM domain.

Parameters **nic_index** – Index of the NIC

Raises **VMACAddressMissingError** – If no MAC address is defined for the requested NIC

get_xml ()

Return VM's xml file.

getenforce()

Set SELinux mode in the VM.

Returns SELinux mode [Enforcing|Permissive|Disabled]

has_swap()

Check if there is any active swap partition/file.

:return : True if swap is on or False otherwise.

install_package(name)

Install a package on VM. ToDo: Support multiple package manager.

Parameters **name** – Name of package to be installed

is_alive()

Return True if VM is alive.

is_autostart()

Return True if VM is autostart.

is_dead()

Return True if VM is dead.

is_esx()

Return True if VM is a esx guest.

is_lxc()

Return True if VM is linux container.

is_paused()

Return True if VM is paused.

is_persistent()

Return True if VM is persistent.

is_qemu()

Return True if VM is a qemu guest.

is_xen()

Return True if VM is a xen guest.

make_create_command(name=None, params=None, root_dir=None)

Generate a libvirt command line. All parameters are optional. If a parameter is not supplied, the corresponding value stored in the class attributes is used.

Parameters

- **name** – The name of the object
- **params** – A dict containing VM params
- **root_dir** – Base directory for relative filenames

Note The params dict should contain: mem – memory size in MBs cdrom – ISO filename to use with the qemu -cdrom parameter extra_params – a string to append to the qemu command shell_port – port of the remote shell daemon on the guest (SSH, Telnet or the home-made Remote Shell Server) shell_client – client program to use for connecting to the remote shell daemon on the guest (ssh, telnet or nc) x11_display – if specified, the DISPLAY environment variable will be set to this value for the qemu process (useful for SDL rendering) images – a list of image object names, separated by spaces nics – a list of NIC object names, separated by spaces

For each image in `images`: `drive_format` – string to pass as ‘if’ parameter for this image (e.g. `ide`, `scsi`) `image_snapshot` – if yes, pass ‘`snapshot=on`’ to `qemu` for this image `image_boot` – if yes, pass ‘`boot=on`’ to `qemu` for this image In addition, all parameters required by `get_image_filename`.

For each NIC in `nics`: `nic_model` – string to pass as ‘`model`’ parameter for this NIC (e.g. `e1000`)

managesave ()

Managed save of VM’s state

migrate (*dest_uri=’, option=’-live -timeout 60’, extra=’, ignore_status=False, debug=False*)

Migrate a VM to a remote host.

Parameters

- **dest_uri** – Destination libvirt URI
- **option** – Migration options before `<domain>` `<desturi>`
- **extra** – Migration options after `<domain>` `<desturi>`

Returns True if command succeeded

pause ()

pmsuspend (*target=’mem’, duration=0*)

Suspend a domain gracefully using power management functions

pmwakeup ()

Wakeup a domain from pmsuspended state

prepare_guest_agent (*prepare_xml=True, channel=True, start=True*)

Prepare qemu guest agent on the VM.

Parameters

- **prepare_xml** – Whether change VM’s XML
- **channel** – Whether add agent channel in VM. Only valid if `prepare_xml` is True
- **start** – Whether install and start the `qemu-ga` service

reboot (**args, **kwargs*)

Reboot the VM and wait for it to come back up by trying to log in until timeout expires.

Parameters

- **session** – A shell session object or None.
- **method** – Reboot method. Can be “`shell`” (send a shell reboot command).
- **nic_index** – Index of NIC to access in the VM, when logging in after rebooting.
- **timeout** – Time to wait for login to succeed (after rebooting).
- **serial** – Just use to unify api in `virt_vm` module.

Returns A new shell session object.

remove ()

remove_package (*name*)

Remove a package from VM. ToDo: Support multiple package manager.

Parameters **name** – Name of package to be removed

remove_with_storage()

Virsh undefine provides an option named `--remove-all-storage`, but it only removes the storage which is managed by libvirt.

This method undefines vm and removes the all storages related with this vm, no matter storages are managed by libvirt or not.

restore_from_file(path)

Override BaseVM `restore_from_file` method

resume()**save_to_file(path)**

Override BaseVM `save_to_file` method

screendump(filename, debug=False)**set_console_getty(device, getty='mgetty', remove=False)**

Set getty for given console device.

Parameters

- **device** – a console device
- **getty** – getty type: agetty, mgetty and so on.
- **remove** – do remove operation

set_kernel_console(device, speed=None, remove=False)

Set kernel parameter for given console device.

Parameters

- **device** – a console device
- **speed** – speed of serial console
- **remove** – do remove operation

set_kernel_param(parameter, value=None, remove=False)

Set a specific kernel parameter.

Parameters

- **option** – A kernel parameter to set.
- **value** – The value of the parameter to be set.
- **remove** – Remove the parameter if True.

Returns True if succeed of False if failed.

set_root_serial_console(device, remove=False)

Allow or ban root to login through serial console.

Parameters

- **device** – device to set root login
- **allow_root** – do remove operation

setenforce(mode)

Set SELinux mode in the VM.

Parameters **mode** – SELinux mode [Enforcing|Permissive|1|0]

shutdown()

Shuts down this VM.

start (*autoconsole=True*)

Starts this VM.

state ()

Return domain state.

undefine ()

Undefine the VM.

vcpuinfo ()

Return a dict's list include vm's vcpu information.

vcupin (*vcpu, cpu_list, options=''*)

To pin vcpu to cpu_list

verify_alive ()

Make sure the VM is alive.

Raises VMDeadError – If the VM is dead

wait_for_login (*nic_index=0, timeout=None, internal_timeout=None, serial=False, restart_network=False, username=None, password=None*)

Override the wait_for_login method of virt_vm to support other guest in libvirt.

If connect_uri is lxc related, we call wait_for_serial_login() directly, without attempting login it via network.

Other connect_uri, call virt_vm.wait_for_login().

wait_for_shutdown (*count=60*)

Return True on successful domain shutdown.

Wait for a domain to shutdown, libvirt does not block on domain shutdown so we need to watch for successful completion.

Parameters

- **name** – VM name
- **name** – Optional timeout value

`virttest.libvirt_vm.complete_uri (ip_address)`

Return a complete URI with the combination of ip_address and local uri. It is useful when you need to connect remote hypervisor.

Parameters **ip_address** – an ip address or a hostname

Returns a complete uri

`virttest.libvirt_vm.get_uri_with_transport (uri_type='qemu', transport='', dest_ip='')`

Return a URI to connect driver on dest with a specified transport.

Parameters

- **origin_uri** – The URI on dest used to connect itself directly.
- **transport** – The transport type connect to dest.
- **dest_ip** – The ip of destination.

`virttest.libvirt_vm.normalize_connect_uri (connect_uri)`

Processes connect_uri Cartesian into something virsh can use

Parameters **connect_uri** – Cartesian Params setting

Returns Normalized connect_uri

virttest.libvirt_xml_unittest module

```
class virttest.libvirt_xml_unittest.AccessorsTest (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_AllForbidden()

    test_XMLElementBool_deep()

    test_XMLElementBool_simple()

    test_XMLElementInt()

    test_XMLElementList()

    test_XMLElementList_Text()

    test_XMLElementNest()

    test_accessor_base()

    test_create_by_xpath()

    test_not_enuf_dargs()

    test_required_slots()

    test_too_many_dargs()

    test_type_check()

class virttest.libvirt_xml_unittest.Bar (parent, virsh_instance)
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase

    baz

class virttest.libvirt_xml_unittest.Baz (parent, virsh_instance)
    Bases: virttest.libvirt_xml.base.LibvirtXMLBase

    foobar

class virttest.libvirt_xml_unittest.LibvirtXMLTestBase (methodName='runTest')
    Bases: unittest.case.TestCase

    setUp()

    tearDown()

class virttest.libvirt_xml_unittest.TestLibvirtXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_guest_capabilities()

    test_uuid()

class virttest.libvirt_xml_unittest.TestVMXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_getters()

    test_new_from_dumpxml()

    test_restore()

    test_seclabel()

    test_valid_xml()
```



```
class virttest.libvirt_xml_unittest.testAddressXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_required()

class virttest.libvirt_xml_unittest.testCAPXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_capxmlbase()

class virttest.libvirt_xml_unittest.testCharacterXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_arbitrart_attributes()

class virttest.libvirt_xml_unittest.testDiskXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_vm_get()

    test_vm_get_by_class()

class virttest.libvirt_xml_unittest.testLibrarian (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_bad_names()

    test_no_module()

    test_serial_class()

class virttest.libvirt_xml_unittest.testNetworkXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_getters()

    test_ip_getter()

    test_valid_xml()

class virttest.libvirt_xml_unittest.testNodedevXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_get_key2syspath_dict()

    test_get_key2value_dict()

    test_new_from_dumpxml()

class virttest.libvirt_xml_unittest.testNodedevXMLBase (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_getter()

    test_static()

class virttest.libvirt_xml_unittest.testPCIXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase

    test_get_key2filename_dict()

    test_get_key2value_dict()

    test_get_path()

    test_static()
```

```
class virttest.libvirt_xml_unittest.testSerialXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase
    XML = u"<serial type='pty'><source path='/dev/null'/> <target port='-1'/></serial>"
    test_from_element ()
    test_getters ()
    test_vm_get_by_class ()
    test_vm_get_modify ()

class virttest.libvirt_xml_unittest.testStubXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase
    TypedFoobar
    UntypedFoobar
    setUp ()
    test_typed_device_stub ()
    test_untyped_device_stub ()

class virttest.libvirt_xml_unittest.testVMCPUTuneXML (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase
    test_get_set_del ()

class virttest.libvirt_xml_unittest.testVMXMLDevices (methodName='runTest')
    Bases: virttest.libvirt_xml_unittest.LibvirtXMLTestBase
    test_channels ()
    test_graphics ()
```

virttest.lvm module

Base module for support lvm in qemu test;

For EmulatedLVM, no need any special configuration, lvm params will generate automatically. Of course, customizable params is accept; For real lvm partition, we need to specify some params, at lest, vg_name and it's a real volume group on your host. If not, both pv_name and vg_name are required and a new volume group will be created on device named pv_name, But it will destroy data on your device and it's not recommended;

Required params:

lv_name: lv_name like /dev/vg/lv; If not params["vg_name"] is required and if lv_name not set, use guest_name as lv_name; device mapper path (eg, /dev/mapper/vg-lv) doesn't support it now;

lv_size string (eg, 30G) if not set image_size will be used;

vg_name LogicalVolume group name, eg, "test_vg";

pv_name PhysicalVolume name eg, /dev/sdb or /dev/sdb1;

```
class virttest.lvm.EmulatedLVM (params, root_dir='/tmp')
    Bases: virttest.lvm.LVM
```

```
    cleanup ()
        Cleanup created logical volumes;
```

```
    get_emulate_image_name ()
```

```

make_emulate_image ()
    Create emulate image via dd with 8M block size;

make_volume (img_file, extra_args='')
    Map a file to loop back device;

    Parameters img_file – image file path;

    Returns loop back device name;

setup ()
    Main function to setup a lvm environments;

    Returns LogicalVolume path

setup_pv (vg)
    Setup physical volume device if exists return it directly;

class virttest.lvm.LVM(params)
    Bases: object

    cleanup ()
        Remove useless lv, vg and pv then reload lvm related service;

    generate_id (params)
        Create prefix with image_name;

    get_vol (vname, vtype)
        Get a exists volume object;

        Parameters

        • vname – volume name;

        • vtype – volume type eg, 'pvs', 'vgs', 'lvs';

        Returns Volume object or None;

    register (vol)
        Register new volume;

        Parameters vol – Volume object or VolumeGroup objects

    rescan ()
        Rescan lvm , used before create volume or after remove volumes;

    setup ()
        Main function to setup a lvm environments;

        Returns LogicalVolume path

    setup_lv ()
        Setup a logical volume, if a exist logical volume resize it else then create it on specify volume group;

        Parameters

        • params["lv_name"] – logical volume name;

        • params["lv_size"] – logical volume size;

        Returns logical volume object;

    setup_pv (vg)
        Create a physical volume devices;

        Parameters

```

- **params**["pv_name"] – Physical volume devices path or mount point;
- **vg** – VolumeGroup object;

Returns list of PhysicalVolume object;

setup_vg(lv)

Setup logical volume group which specify on volume group specify by params["vg_name"];

Parameters **params**["vg_name"] – volume group name;

Returns volume group object;

unregister(vol)

Unregister volume or VolumeGroup;

Parameters **vol** – Volume object or VolumeGroup objects

class virttest.lvm.**LogicalVolume**(name, size, vg)

Bases: *virttest.lvm.Volume*

create()

Create LogicalVolume device;

Returns path of logical volume;

display(extra_args='')

Shown logical volume details, wrapper of lvm command lvdisplay;

Extra_args extra arguments pass to lvdisplay command;

Raise CmdError when command exit code not equal 0;

get_attr(attr)

Get logical volume attributes if not found return None;

Parameters **attr** – attribute name;

Returns attribute value string or None;

Raise CmdError when command exit code not equal 0;

remove(extra_args='--ff--yes')

Remove LogicalVolume device;

Parameters **extra_args** – extra arguments pass to lvm command;

resize(size, extra_args='--ff')

Resize LogicalVolume to new size;

Parameters

- **size** – new size of logical volume;
- **extra_args** – extra arguments pass to lvm command;

Returns size of logical volume;

class virttest.lvm.**PhysicalVolume**(name, size)

Bases: *virttest.lvm.Volume*

create(extra_args='--ff--yes')

Create physical volume on specify physical volume;

Parameters **extra_args** – extra arguments for pvcreate command;

Raise CmdError or TestError;

Returns physical volume abspath

display ()

Show physical volume details

Raise CmdError

get_attr (attr)

Get attribute of physical volume, if not found return None;

Parameters attr – attribute name of the volume;

Returns string or None

remove (extra_args=' -ff -yes')

Remove a physical volume

Parameters extra_args – extra arguments for pvremove command

Raise CmdError

resize (size, extra_args=' -ff -yes')

Resize a physical volume;

Parameters

- **size** – new size of the physical volume device;
- **extra_args** – extra arguments for pvresize command;

set_vg (vg)

Set VolumeGroup of the physical volume device;

Parameters vg – VolumeGroup object

class virttest.lvm.**Volume** (name, size)

Bases: `object`

exists ()

Check if the volume really exists or not;

get_attr (cmd, attr, res='[/\\w/]+')

Get attribute of volume, if not found return None;

Parameters

- **cmd** – command used to display volume info;
- **attr** – attribute name of the volume;
- **res** – regular expression to reading the attribute;

Returns string or None

umount (extra_args='-f')

Unmount volume;

class virttest.lvm.**VolumeGroup** (name, size, pvs)

Bases: `object`

append_lv (lv)

Collect Logical Volumes on the VolumeGroup;

Parameters lv – LogicalVolume Object

create (extra_args=' -ff -yes')

Create volume group with specified physical volumes;

Parameters `extra_args` – extra arguments for lvm command;

Raise `CmdError` or `TestError`;

Returns volume group name;

exists ()

Check `VolumeGroup` exists or not;

Returns bool type, if exists `True` else `False`;

extend_pv (*pv*, *extra_args*='')

Add `PhysicalVolume` into `VolumeGroup`;

Parameters

- `pv` – `PhysicalVolume` object
- `extra_args` – extra arguments used for `vgextend` command

get_attr (*attr*)

Get `VolumeGroup` attribute;

Parameters `attr` – attribute name;

Returns string or `None`;

reduce_pv (*pv*, *extra_args*='-ff-yes')

Reduce a `PhysicalVolume` from `VolumeGroup`;

Parameters

- `pv` – `PhysicalVolume` object;
- `extra_args` – extra arguments pass to lvm command;

remove (*extra_args*='-ff-yes')

Remove the `VolumeGroup`;

Parameters `extra_args` – extra arguments for lvm command;

`virttest.lvm.cmd_output` (*cmd*, *res*='[\\w/]+')

`virttest.lvm.normalize_data_size` (*size*)

virttest.lvsb module

Higher order classes and functions for Libvirt Sandbox (lxc) container testing

copyright 2013 Red Hat Inc.

class `virttest.lvsb.TestBaseSandboxes` (*params*, *env*)

Bases: `virttest.lvsb_base.TestSandboxes`

Simplistic sandbox aggregate manager

command_suffixes ()

Append command after a –

results (*each_timeout*=5)

Run sandboxe(s), allowing *each_timeout* to complete, return output list

class `virttest.lvsb.TestComplexSandboxes` (*params*, *env*)

Bases: `virttest.lvsb.TestBaseSandboxes`

Executes a command with complex options

class `virttest.lvsb.TestSimpleSandboxes` (*params*, *env*)

Bases: `virttest.lvsb.TestBaseSandboxes`

Executes a command with simple options

`virttest.lvsb.make_sandboxes` (*params*, *env*, *extra_ns=None*)

Return list of instantiated `lvsb_testsandboxes` classes from *params*

Parameters

- **params** – an undiluted `Params` instance
- **env** – the current `env` instance
- **extra_ns** – An extra, optional namespace to search for classes

`virttest.lvsb_base` module

Base classes supporting Libvirt Sandbox (lxc) container testing

copyright 2013 Red Hat Inc.

class `virttest.lvsb_base.SandboxBase` (*params*)

Bases: `object`

Base operations for sandboxed command

auto_clean (*boolean*)

Change behavior of asynchronous background sandbox process on `__del__`

exit_code ()

Block until asynchronous background sandbox process ends, returning code

fini ()

Finalize asynchronous background sandbox process (destroys state!)

instances = `None`

make_sandbox_command_line (*extra=None*)

Return the fully formed command-line for the sandbox using `self.options`

recv ()

Return stdout and stderr from asynchronous background sandbox process

recvrr ()

return only stderr from asynchronous background sandbox process

recvout ()

Return only stdout from asynchronous background sandbox process

run (*extra=None*)

Launch new sandbox as asynchronous background sandbox process

Parameters **extra** – String of extra command-line to use but not store

running ()

Return True/False if asynchronous background sandbox process executing

send (*data*)

Send data to asynchronous background sandbox process

stop ()

Destroy but don't finalize asynchronous background sandbox process

class `virttest.lvsb_base.SandboxCommandBase` (*params*, *name=None*)
Bases: `virttest.lvsb_base.SandboxBase`
Connection to a single new or existing sandboxed command
BINARY_PATH_PARAM = 'virt_sandbox_binary'
add_flag (*option*)
Add a flag into the list of command line options
add_mm ()
Append a – to the end of the current option list
add_optarg (*option*, *argument*)
Add an option with an argument into the list of command line options
add_pos (*argument*)
Add a positional option into the list of command line options
static flatten_options (*options*)
Convert a list of tuples into space-separated options+argument string
list_flags ()
Return a list of all flags (options without arguments)
list_long_options ()
Return a list of all long options with an argument
list_pos ()
Return a list of all positional arguments
list_short_options ()
Return a list of all short options with an argument
make_sandbox_command_line (*extra=None*)
Return entire command-line string needed to start sandbox
name
Represent a unique sandbox name generated from class and identifier

exception `virttest.lvsb_base.SandboxException` (*message*)
Bases: `exceptions.Exception`
Basic exception class for problems occurring in SandboxBase or subclasses

class `virttest.lvsb_base.SandboxSession`
Bases: `object`
Connection instance to asynchronous I/O redirector process
auto_clean (*boolean*)
Make session cleanup on GC if True
close_session (*warn_if_nonexist=True*)
Finalize assigned opaque session object
connected
Represents True/False value if background process was created/opened
exit_code ()
Block, and return exit code from session
is_running ()
Return True if `exit_code()` would block

kill_session (*sig=15*)
Send a signal to the opaque session object

new_session (*command*)
Create and set new opaque session object

open_session (*a_id*)
Restore connection to existing session identified by *a_id*

recv ()
Return combined stdout/stderr output received so far

recvrr ()
Return just stderr output

recvout ()
Return just stdout output

send (*a_string*)
Send *a_string* to session

session_id
Returns unique & persistent identifier for the background process

used = False

class `virttest.lvsb_base.TestSandboxes` (*params, env*)
Bases: `object`

Aggregate manager class of `SandboxCommandBase` or subclass instances

SANDBOX_TYPE
alias of `SandboxCommandBase`

are_failed ()
Return the number of sandbox processes with non-zero exit codes

are_running ()
Return the number of sandbox processes still running

for_each (*do_something, *args, **dargs*)
Iterate over all sandboxes, calling *do_something* on each

Parameters **do_sometihng** – Called with the item and **args, **dargs*

init_sandboxes ()
Create `self.count` Sandbox instances

virttest.lvsbs module

Higher order classes for Libvirt Sandbox Service (lxc) service container testing

class `virttest.lvsbs.SandboxService` (*params, service_name, uri='lxc:///'*)
Bases: `object`

Management for a single new/existing sandboxed service

create ()

destroy ()

list
Return list of dictionaries mapping column names to values

```
service_name
uri
xmlstr
```

virttest.nfs module

Basic nfs support for Linux host. It can support the remote nfs mount and the local nfs set up and mount.

```
class virttest.nfs.Exportfs (path, client='*', options='', ori_exported=None)
```

Bases: `object`

Add or remove one entry to exported nfs file system.

export ()

Export one directory if it is not in exported list.

Returns Export nfs file system succeed or not

is_exported ()

Check if the directory is already exported.

Returns If the entry is exported

Return type Boolean

need_reexport ()

Check if the entry is already exported but the options are not the same as we required.

Returns Need re export the entry or not

Return type Boolean

reset_export ()

Reset the exportfs to the original status before we export the specific entry.

unexport ()

Unexport an entry.

```
class virttest.nfs.NFSClient (params)
```

Bases: `object`

NFSClient class for handle nfs remotely mount and umount.

cleanup ()

Cleanup NFS client.

is_mounted ()

Check the NFS is mounted or not.

Returns If the src is mounted as expect

Return type Boolean

setup ()

Setup NFS client.

setup_remote ()

Mount sharing directory to remote host.

```
class virttest.nfs.Nfs (params)
```

Bases: `object`

Nfs class for handle nfs mount and umount. If a local nfs service is required, it will configure a local nfs server according to the params.

cleanup()

Clean up the host env.

Umount NFS from the mount point. If there has some change for exported file system in host when setup, also clean up that.

is_mounted()

Check the NFS is mounted or not.

Returns If the src is mounted as expect

Return type Boolean

mount()

Mount source into given mount point.

setup()

Setup NFS in host.

Mount NFS as configured. If a local nfs is requested, setup the NFS service and exportfs too.

umount()

Umount the given mount point.

virttest.nfs.nfs_exported()

Get the list for nfs file system already exported

Returns a list of nfs that is already exported in system

Return type a list of nfs file system exported

virttest.nfs_unittest module

class virttest.nfs_unittest.**FakeService**(*service_name*)

Bases: `object`

get_stdout(*cmd*)

restart()

status()

class virttest.nfs_unittest.**nfs_test**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

setUp()

setup_stubs_cleanup(*nfs_obj*)

setup_stubs_init()

setup_stubs_is_mounted(*nfs_obj*)

setup_stubs_setup(*nfs_obj*)

tearDown()

test_nfs_setup()

virttest.openvswitch module

Wrapper around module.

Necessary for pickling of dynamic class.

```
class virttest.openvswitch.OpenVSwitch(tmpdir, db_path=None, db_socket=None,
                                         db_pidfile=None, ovs_pidfile=None, dbschema=None,
                                         install_prefix=None)
```

Bases: `virttest.openvswitch.OpenVSwitchSystem`

OpenVSwitch class.

clean()

init_db()

init_new()

Create new dbfile without any configuration.

start_ovs_vswitchd()

```
class virttest.openvswitch.OpenVSwitchControl
```

Bases: `object`

Class select the best matches control class for installed version of OpenVSwitch.

OpenVSwitch parameters are described in man ovs-vswitchd.conf.db

add_br(br_name)

add_port(br_name, port_name)

add_port_tag(port_name, tag)

add_port_trunk(port_name, trunk)

br_exist(br_name)

check_port_in_br(br_name, port_name)

static convert_version_to_int(version)

Parameters **version** – (int) Converted from version string 1.4.0 => int 140

del_br(br_name)

del_port(br_name, port_name)

classmethod get_version()

Get version of installed OpenVSwitch.

Returns Version of OpenVSwitch.

list_br()

set_vlanmode(port_name, vlan_mode)

status()

```
class virttest.openvswitch.OpenVSwitchControlCli_140
```

Bases: `virttest.openvswitch.OpenVSwitchControl`

Don't use this class directly. This class is automatically selected by OpenVSwitchControl.

add_br(br_name)

add_fake_br(br_name, parent, vlan)

add_port (*br_name*, *port_name*)

add_port_tag (*port_name*, *tag*)

add_port_trunk (*port_name*, *trunk*)

Parameters **trunk** – list of vlans id.

br_exist (*br_name*)

del_br (*br_name*)

del_port (*br_name*, *port_name*)

list_br ()

list_ports (*br_name*)

ovs_vsctl (*parms*, *ignore_status=False*)

port_to_br (*port_name*)

Return bridge which contain port.

Parameters **port_name** – Name of port.

Returns Bridge name or None if there is no bridge which contain port.

set_vlanmode (*port_name*, *vlan_mode*)

status ()

class `virttest.openvswitch.OpenVSwitchControlCli_CNT`

Bases: `virttest.versionable_class.VersionableClass`

class `virttest.openvswitch.OpenVSwitchControlDB_140`

Bases: `virttest.openvswitch.OpenVSwitchControl`

Don't use this class directly. This class is automatically selected by OpenVSwitchControl.

class `virttest.openvswitch.OpenVSwitchControlDB_CNT`

Bases: `virttest.versionable_class.VersionableClass`

class `virttest.openvswitch.OpenVSwitchSystem` (*db_path=None*, *db_socket=None*,
db_pidfile=None, *ovs_pidfile=None*, *db-
 schema=None*, *install_prefix=None*)

Bases: `virttest.openvswitch.OpenVSwitchControlCli_CNT`,
`virttest.openvswitch.OpenVSwitchControlDB_CNT`

OpenVSwitch class.

check ()

check_db_daemon ()

Check if OVS daemon is started correctly.

check_db_file ()

Check if db_file exists.

check_db_socket ()

Check if db socket exists.

check_switch_daemon ()

Check if OVS daemon is started correctly.

clean ()

Empty cleanup function

init_system()

Create new dbfile without any configuration.

is_installed()

Check if OpenVSwitch is already installed in system on default places.

Returns Version of OpenVSwitch.

class `virttest.openvswitch.ServiceManager`

Bases: `virttest.versionable_class.VersionableClass`

class `virttest.openvswitch.ServiceManagerInterface`

Bases: `object`

classmethod `get_version()`

Get version of ServiceManager. :return: Version of ServiceManager.

restart (*service_name*)

start (*service_name*)

status (*service_name*)

stop (*service_name*)

class `virttest.openvswitch.ServiceManagerSystemD`

Bases: `virttest.openvswitch.ServiceManagerSysvinit`

restart (*service_name*)

start (*service_name*)

status (*service_name*)

stop (*service_name*)

class `virttest.openvswitch.ServiceManagerSysvinit`

Bases: `virttest.openvswitch.ServiceManagerInterface`

restart (*service_name*)

start (*service_name*)

stop (*service_name*)

virttest.ovirt module

oVirt SDK wrapper module.

copyright 2008-2012 Red Hat Inc.

class `virttest.ovirt.ClusterManager` (*params*)

Bases: `object`

This class handles all basic cluster operations.

add (*dc_name*, *cpu_type*=*'Intel Nehalem Family'*)

Add a new cluster into data center.

list ()

List all of clusters.

class `virttest.ovirt.DataCenterManager` (*params*)

Bases: `object`

This class handles all basic datacenter operations.

add (*storage_type*)
Add a new data center.

list ()
List all of datacenters.

class `virttest.ovirt.HostManager` (*params*)

Bases: `object`

This class handles all basic host operations.

add (*host_address*, *host_password*, *cluster_name*, *timeout=300*)
Register a host into specified cluster.

get_address ()
Return host IP address.

list ()
List all of hosts.

state ()
Return host state.

class `virttest.ovirt.StorageDomainManager` (*params*)

Bases: `object`

This class handles all basic storage domain operations.

attach_iso_export_domain_into_datacenter (*address*, *path*, *dc_name*, *host_name*,
domain_type, *storage_type='nfs'*,
name='my_iso')

Attach ISO/export domain into data center.

Parameters

- **name** – ISO or Export name.
- **host_name** – host name.
- **dc_name** – data center name.
- **path** – ISO/export domain path.
- **address** – ISO/export domain address.
- **domain_type** – storage domain type, it may be 'iso' or 'export'.
- **storage_type** – storage type, it may be 'nfs', 'iscsi', or 'fc'.

list ()
List all of storagedomains.

class `virttest.ovirt.VMManager` (*params*, *root_dir*, *address_cache=None*, *state=None*)

Bases: `virttest.virt_vm.BaseVM`

This class handles all basic VM operations for oVirt.

add (*memory*, *disk_size*, *cluster_name*, *storage_name*, *nic_name='eth0'*, *network_interface='virtio'*,
network_name='ovirtmgmt', *disk_interface='virtio'*, *disk_format='raw'*, *template_name='Blank'*,
timeout=300)
Create VM with one NIC and one Disk.

Parameters

- **memory** – VM's memory size such as 1024*1024*1024=1GB.
- **disk_size** – VM's disk size such as 512*1024=512MB.

- **nic_name** – VM's NICs name such as 'eth0'.
- **network_interface** – VM's network interface such as 'virtio'.
- **network_name** – network such as ovirtmgmt for ovirt, rhevm for rhel.
- **disk_format** – VM's disk format such as 'raw' or 'cow'.
- **disk_interface** – VM's disk interface such as 'virtio'.
- **cluster_name** – cluster name.
- **storage_name** – storage domain name.
- **template_name** – VM's template name, default is 'Blank'.
- **timeout** – Time out

add_vm_from_template (*cluster_name*, *template_name*='Blank', *new_name*='my_new_vm', *timeout*=300)
Create a VM from template.

Parameters

- **cluster_name** – cluster name.
- **template_name** – default template is 'Blank'.
- **new_name** – 'my_new_vm' is a default new VM's name.
- **timeout** – Time out

create_template (*cluster_name*, *template_name*='my_template', *timeout*=300)
Create a template from VM.

Parameters

- **cluster_name** – cluster name.
- **template_name** – 'my_template' is default template name.
- **timeout** – Time out

delete (*timeout*=300)
Delete a VM.

delete_from_export_domain (*export_name*)
Remove a VM from specified export domain.

Parameters **export_name** – export domain name.

destroy (*gracefully*=False)
Destroy a VM.

export_from_export_domain (*export_name*, *timeout*=300)
Export a VM from storage domain to export domain.

Parameters **export_name** – Export domain name.

get_address (*index*=0)
Return the address of the guest through ovirt node tcpdump cache.

Parameters **index** – Name or index of the NIC whose address is requested.

Returns IP address of NIC.

Raises **VMIPAddressMissingError** – If no IP address is found for the the NIC's MAC address

get_mac_address (*net_name='**)

Return MAC address of a VM.

import_from_export_domain (*export_name, storage_name, cluster_name, timeout=300*)

Import a VM from export domain to data domain.

Parameters

- **export_name** – Export domain name.
- **storage_name** – Storage domain name.
- **cluster_name** – Cluster name.

is_alive ()

Judge if a VM is alive.

is_dead ()

Judge if a VM is dead.

is_paused ()

Return if VM is suspend.

list ()

List all of VMs.

lookup_by_storagedomains (*storage_name*)

Lookup VM object in storage domain according to VM name.

resume (*timeout*)

Resume a suspended VM.

shutdown (*gracefully=True, timeout=300*)

Shut down a running VM.

snapshot (*snapshot_name='my_snapshot', timeout=300*)

Create a snapshot to VM.

Parameters

- **snapshot_name** – ‘my_snapshot’ is default snapshot name.
- **timeout** – Time out

start (*wait_for_up=True, timeout=300*)

Start a VM.

state ()

Return VM state.

suspend (*timeout*)

Suspend a VM.

update_instance ()

exception `virttest.ovirt.WaitHostStateTimeoutError` (*msg, output*)

Bases: `virttest.ovirt.WaitStateTimeoutError`

exception `virttest.ovirt.WaitStateTimeoutError` (*msg, output*)

Bases: `exceptions.Exception`

exception `virttest.ovirt.WaitVMStateTimeoutError` (*msg, output*)

Bases: `virttest.ovirt.WaitStateTimeoutError`

`virttest.ovirt.connect` (*params*)

Connect ovirt manager API.

`virttest.ovirt.disconnect()`
Disconnect ovirt manager connection.

virttest.ovs_utils module

class `virttest.ovs_utils.Machine` (*vm=None, src=None*)
Bases: `object`

add_vlan_iface (*iface, vlan_id*)
Add vlan link for interface

Parameters

- **iface** – Interface on which should be added vlan.
- **vlan_id** – Id of vlan.

bring_iface_down (*iface*)
Bring interface up

bring_iface_up (*iface*)
Bring interface up

cmd (*cmd, timeout=60*)
Return output of command.

cmd_in_src (*cmd, timeout=60*)

cmd_state (*cmd, timeout=60*)
Return status of command.

compile_autotools_app_tar (*path, package_name*)
Compile app on machine in src dir.

Parameters

- **path** – Path where should be program compiled.
- **dst_dir** – Installation path.

copy_to (*src, dst*)

del_vlan_iface (*iface, vlan_id*)
Del vlan link for interface

Parameters

- **iface** – Interface from which should be deleted vlan.
- **vlan_id** – Id of vlan.

fill_addrs ()

get_if_vlan_name (*ifname, vlan_id=0*)

get_linkv6_addr (*ifname*)
Get IPv6 address with link range.

Parameters **ifname** – String or int. Int could be used only for virt Machine.

Returns IPv6 link address.

get_vlans_ifname ()
Return vlans interface name.

Returns dict of {"ifname": [(vlanid, ifname), (...)], ...}

is_virtual()

Returns True when Machine is virtual.

ping (*dst, iface=None, count=1, vlan=0, ipv=None*)

Ping destination.

prepare_directory (*path, cleanup=False*)

Prepare dest directory. Create if directory not exist.

Parameters

- **path** – Path to directory
- **cleanup** – If true clears the contents of directory.

`virttest.ovs_utils.ping4` (*iface, dst_ip, count=1, runner=None*)

Format command for ipv4.

`virttest.ovs_utils.ping6` (*iface, dst_ip, count=1, runner=None*)

Format command for ipv6.

virttest.passfd module

virttest.passfd_setup module

`virttest.passfd_setup.import_passfd()`

Imports and lazily sets up the passfd module

Returns passfd module

`virttest.passfd_setup.passfd_setup` (*output_dir='/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest'*)

Compiles the passfd python extension.

Parameters **output_dir** – where the `_passfd.so` module will be saved

Returns None

virttest.postprocess_iozone module

Postprocessing module for IOzone. It is capable to pick results from an IOzone run, calculate the geometric mean for all throughput results for a given file size or record size, and then generate a series of 2D and 3D graphs. The graph generation functionality depends on gnuplot, and if it is not present, functionality degrades gracefully.

copyright Red Hat 2010

class `virttest.postprocess_iozone.AnalyzerLoggingConfig` (*use_console=True*)

Bases: `autotest.client.shared.logging_config.LoggingConfig`

configure_logging (*results_dir=None, verbose=False*)

class `virttest.postprocess_iozone.IOzoneAnalyzer` (*list_files, output_dir*)

Bases: `object`

Analyze an unprocessed IOzone file, and generate the following types of report:

- Summary of throughput for all file and record sizes combined
- Summary of throughput for all file sizes
- Summary of throughput for all record sizes

If more than one file is provided to the analyzer object, a comparison between the two runs is made, searching for regressions in performance.

analyze()

Analyzes and eventually compares sets of IOzone data.

average_performance(*results*, *size=None*)

Flattens a list containing performance results.

Parameters

- **results** – List of n lists containing data from performance runs.
- **size** – Numerical value of a size (say, *file_size*) that was used to filter the original results list.

Returns List with 1 list containing average data from the performance run.

parse_file(*fileobj*)

Parse an IOzone results file.

Parameters **file** – File object that will be parsed.

Returns Matrix containing IOzone results extracted from the file.

process_results(*results*, *label=None*)

Process a list of IOzone results according to label.

Parameters

- **label** – IOzone column label that we'll use to filter and compute geometric mean results, in practical term either 'file_size' or 'record_size'.
- **result** – A list of n x m columns with original iozone results.

Returns A list of n-? x (m-1) columns with geometric averages for values of each label (ex, average for all file_sizes).

report(*overall_results*, *record_size_results*, *file_size_results*)

Generates analysis data for IOZone run.

Generates a report to both logs (where it goes with nice headers) and output files for further processing (graph generation).

Parameters

- **overall_results** – 1x15 Matrix containing IOzone results for all file sizes
- **record_size_results** – nx15 Matrix containing IOzone results for each record size tested.
- **file_size_results** – nx15 Matrix containing file size results for each file size tested.

report_comparison(*record*, *file_size_results*)

Generates comparison data for 2 IOZone runs.

It compares 2 sets of nxm results and outputs a table with differences. If a difference higher or smaller than 5% is found, a warning is triggered.

Parameters

- **record** – Tuple with 4 elements containing results for record size.
- **file_size_results** – Tuple with 4 elements containing results for file size.

class `virttest.postprocess_iozone.IOzonePlotter` (*results_file*, *output_dir*)

Bases: `object`

Plots graphs based on the results of an IOzone run.

Plots graphs based on the results of an IOzone run. Uses gnuplot to generate the graphs.

generate_data_source ()

Creates data file without headers for gnuplot consumption.

plot_2d_graphs ()

For each one of the throughput parameters, generate a set of gnuplot commands that will create a parametric surface with file size vs. record size vs. throughput.

plot_3d_graphs ()

For each one of the throughput parameters, generate a set of gnuplot commands that will create a parametric surface with file size vs. record size vs. throughput.

plot_all ()

Plot all graphs that are to be plotted, provided that we have gnuplot.

`virttest.postprocess_iozone.compare_matrices` (*matrix1*, *matrix2*, *threshold=0.05*)

Compare 2 matrices nxm and return a matrix nxm with comparison data

Parameters

- **matrix1** – Reference Matrix with numeric data
- **matrix2** – Matrix that will be compared
- **threshold** – Any difference bigger than this percent threshold will be reported.

`virttest.postprocess_iozone.geometric_mean` (*values*)

Evaluates the geometric mean for a list of numeric values.

Parameters **values** – List with values.

Returns Single value representing the geometric mean for the list values.

See [Geometric mean definition](#)

virttest.ppm_utils module

Utility functions to deal with ppm (qemu screendump format) files.

copyright Red Hat 2008-2009

`virttest.ppm_utils.cal_hamming_distance` (*h1*, *h2*)

Calculate the hamming distance

`virttest.ppm_utils.find_id_for_screendump` (*md5sum*, *data_dir*)

Search dir for a PPM file whose name ends with md5sum.

Parameters

- **md5sum** – md5 sum string
- **dir** – Directory that holds the PPM files.

Returns The file's basename without any preceding path, e.g.
20080101_120000_d41d8cd98f00b204e9800998ecf8427e.ppm

`virttest.ppm_utils.generate_id_for_screendump` (*md5sum*, *data_dir*)

Generate a unique filename using the given MD5 sum.

Returns Only the file basename, without any preceding path. The filename consists of the current date and time, the MD5 sum and a .ppm extension, e.g. 20080101_120000_d41d8cd98f00b204e9800998ecf8427e.ppm.

`virttest.ppm_utils.get_data_dir(steps_filename)`

Return the data dir of the given steps filename.

`virttest.ppm_utils.get_region_md5sum(width, height, data, x1, y1, dx, dy, cropped_image_filename=None)`

Return the md5sum of a cropped region.

Parameters

- **width** – Original image width
- **height** – Original image height
- **data** – Image data
- **x1** – Desired x coord of the cropped region
- **y1** – Desired y coord of the cropped region
- **dx** – Desired width of the cropped region
- **dy** – Desired height of the cropped region
- **cropped_image_filename** – if not None, write the resulting cropped image to a file with this name

`virttest.ppm_utils.have_similar_img(base_img, comp_img_path, threshold=10)`

Check whether comp_img_path have a image looks like base_img.

`virttest.ppm_utils.image_average_hash(image, img_wd=8, img_ht=8)`

Resize and convert the image, then get image data as sequence object, calculate the average hash :param image: an image path or an opened image object

`virttest.ppm_utils.image_comparison(width, height, data1, data2)`

Generate a green-red comparison image from two given images.

Parameters

- **width** – Width of both images
- **height** – Height of both images
- **data1** – Data of first image
- **data2** – Data of second image

Returns A 3-element tuple containing the width, height and data of the generated comparison image.

Note Input images must be the same size.

`virttest.ppm_utils.image_crop(width, height, data, x1, y1, dx, dy)`

Crop an image.

Parameters

- **width** – Original image width
- **height** – Original image height
- **data** – Image data
- **x1** – Desired x coordinate of the cropped region
- **y1** – Desired y coordinate of the cropped region

- **dx** – Desired width of the cropped region
- **dy** – Desired height of the cropped region

Returns A 3-tuple containing the width, height and data of the cropped image.

`virttest.ppm_utils.image_crop_save(image, new_image, box=None)`

Crop an image and save it to a new image.

Parameters

- **image** – Full path of the original image
- **new_image** – Full path of the cropped image
- **box** – A 4-tuple defining the left, upper, right, and lower pixel coordinate.

Returns True if crop and save image succeed

`virttest.ppm_utils.image_fuzzy_compare(width, height, data1, data2)`

Return the degree of equality of two given images.

Parameters

- **width** – Width of both images
- **height** – Height of both images
- **data1** – Data of first image
- **data2** – Data of second image

Returns Ratio equal_pixel_count / total_pixel_count.

Note Input images must be the same size.

`virttest.ppm_utils.image_histogram_compare(image_a, image_b, size=(0, 0))`

Compare the histogram of two images and return similar degree.

Parameters

- **image_a** – Full path of the first image
- **image_b** – Full path of the second image
- **size** – Convert image to size(width, height), and if size=(0, 0), the function will convert the big size image align with the small one.

`virttest.ppm_utils.image_md5sum(width, height, data)`

Return the md5sum of an image.

Parameters

- **width** – PPM file width
- **height** – PPM file height
- **data** – PPM file data

`virttest.ppm_utils.image_read_from_ppm_file(filename)`

Read a PPM image.

Returns A 3 element tuple containing the width, height and data of the image.

`virttest.ppm_utils.image_verify_ppm_file(filename)`

Verify the validity of a PPM file.

Parameters **filename** – Path of the file being verified.

Returns True if filename is a valid PPM image file. This function reads only the first few bytes of the file so it should be rather fast.

`virttest.ppm_utils.image_write_to_ppm_file(filename, width, height, data)`

Write a PPM image with the given width, height and data.

Parameters

- **filename** – PPM file path
- **width** – PPM file width (pixels)
- **height** – PPM file height (pixels)

`virttest.ppm_utils.img_ham_distance(base_img, comp_img)`

Calculate two images hamming distance

`virttest.ppm_utils.img_similar(base_img, comp_img, threshold=10)`

check whether two images are similar by hamming distance

`virttest.ppm_utils.md5eval(data)`

Returns a md5 hash evaluator. This function is implemented in order to encapsulate objects in a way that is compatible with python 2.4 and python 2.6 without warnings.

Parameters **data** – Optional input string that will be used to update the object.

virttest.propcan module

Class which allows property and dict-like access to a fixed set of instance attributes. Attributes are locked by `__slots__`, however accessor methods may be created/removed on instances, or defined by the subclass. An `INITIALIZED` attribute is provided to signal completion of `__init__()` for use by accessor methods (i.e. so they know when `__init__` may be setting values).

Subclasses must define a `__slots__` class attribute containing the list of attribute names to reserve. All additional subclass descendents must explicitly copy `__slots__` from the parent in their definition.

Users of subclass instances are expected to get/set/del attributes only via the standard object or dict-like interface. i.e.

`instance.attribute = whatever` or `instance['attribute'] = whatever`

Internally, methods are free to call the accessor methods. Only accessor methods should use the special `__dict_*__()` and `__super_*__()` methods. These are there to allow convenient access to the internal dictionary values and subclass-defined attributes (such as `__slots__`).

example:

```
class A(PropCan):
    # Class with *attributes*
    __slots__ = ('a', 'b')
    # 'a' has defined a set/get/del by definition of method with prefix
    #     set_a, get_a, del_a
    # 'b' doesn't have defined set/get/del then classic set/get/del will be
    #     called instead.

    def __init__(self, a=1, b='b'):
        super(A, self).__init__(a, b)

    def set_a(self, value)
        # If is_instance(obj, A) then obj.a = "val" call this method.
        self.__dict_set__("a", value)
```



```

def get_a(self, value)
    # If is_instance(obj, A) then xx = obj.a call this method.
    return self.__dict_get__("a")

def del_a(self, value)
    # If is_instance(obj, A) then del obj.a call this method.
    self.__dict_del__("a")

class B(PropCan):
    # Class without *attributes*
    # ***** Even if class doesn't have attributes there should be
    # defined __slots__ = []. Because it is preferred by new style of class.
    # *****
    __slots__ = []

    def __init__(self):
        super(B, self).__init__()

```

class `virttest.propcan.PropCan` (*args, **dargs)

Bases: `virttest.propcan.PropCanBase`

Special value handling on retrieval of None values

has_key (key)

items ()

keys ()

set_if_none (key, value)

Set the value of key, only if it's not set or None

set_if_value_not_none (key, value)

Set the value of key, only if value is not None

values ()

class `virttest.propcan.PropCanBase` (*args, **dargs)

Bases: `dict`, `virttest.propcan.PropCanInternal`

Objects with optional accessor methods and dict-like access to fixed set of keys

INITIALIZED = False

copy ()

Copy properties by value, not by reference.

update (other=None, except=<type 'exceptions.AttributeError'>, **kwargs)

Update properties in `__all_slots__` with another dict.

class `virttest.propcan.PropCanInternal`

Bases: `object`

Semi-private methods for use only by PropCanBase subclasses (NOT instances)

class `virttest.propcan.classproperty`

Bases: `property`

virttest.propcan_unittest module

```
class virttest.propcan_unittest.TestPropCan (methodName='runTest')
    Bases: unittest.case.TestCase

    setUp()

    test_compare()

    test_extraneous_init()

    test_init_None_value()

    test_odd_values()

    test_printables()

class virttest.propcan_unittest.TestPropCanBase (methodName='runTest')
    Bases: unittest.case.TestCase

    test_dict_methods_1()

    test_dict_methods_2()

    test_double_init()

    test_empty_init()

    test_empty_params_init()

    test_mixed_init()

    test_single_init()

    test_slots_restrict()

    test_subclass_no_mask_attributeerror()

    test_subclass_single_init_delter()

    test_subclass_single_init_getter()

    test_subclass_single_init_setter()

    test_update()
```

virttest.qemu_devices_unittest module

This is a unittest for qemu_devices library.

author Lukas Doktor <ldoktor@redhat.com>

copyright 2012 Red Hat, Inc.

```
class virttest.qemu_devices_unittest.Buses (methodName='runTest')
    Bases: unittest.case.TestCase

    Set of bus-representation tests

    test_q_pci_bus()
        PCI bus tests

    test_q_pci_bus_strict()
        PCI bus tests in strict_mode (enforce additional options)

    test_q_sparse_bus()
        Sparse bus tests (general bus testing)
```

```

test_usb_bus ()
    Tests the specific handlings of QUSBBus

class virttest.qemu_devices_unittest.Container (methodName='runTest')
    Bases: unittest.case.TestCase

    Tests related to the abstract representation of qemu machine

    create_qdev (vm_name='vml', strict_mode='no', allow_hotplugged_vm='yes')

        Returns Initialized qcontainer.DevContainer object

    setUp ()

    tearDown ()

    test_pci ()

    test_qdev_equal ()

    test_qdev_functional ()
        Test basic qdev workflow

    test_qdev_hotplug ()
        Test the hotplug/unplug functionality

    test_qdev_low_level ()
        Test low level functions

class virttest.qemu_devices_unittest.Devices (methodName='runTest')
    Bases: unittest.case.TestCase

    set of qemu devices tests

    test_q_base_device ()
        QBaseDevice tests

    test_q_device ()
        QDevice tests

    test_q_string_device ()
        QStringDevice tests

class virttest.qemu_devices_unittest.MockHMPMonitor
    Bases: virttest.qemu_monitor.HumanMonitor

    Dummy class inherited from qemu_monitor.HumanMonitor

class virttest.qemu_devices_unittest.ParamsDict
    Bases: dict

    params like dictionary

    object_params (obj)

    objects (item)

```

virttest.qemu_installer module

Installer code that implement KVM specific bits.

See BaseInstaller class in base_installer.py for interface details.

```
class virttest.qemu_installer.GitRepoInstaller(mode, name, test=None, params=None)
    Bases: virttest.qemu_installer.QEMUBaseInstaller, virttest.base_installer.GitRepoInstaller
    Installer that deals with source code on Git repositories

class virttest.qemu_installer.LocalSourceDirInstaller(mode, name, test=None,
    params=None)
    Bases: virttest.qemu_installer.QEMUBaseInstaller, virttest.base_installer.LocalSourceDirIn
    Installer that deals with source code on local directories

class virttest.qemu_installer.LocalSourceTarInstaller(mode, name, test=None,
    params=None)
    Bases: virttest.qemu_installer.QEMUBaseInstaller, virttest.base_installer.LocalSourceTarIn
    Installer that deals with source code on local tarballs

class virttest.qemu_installer.RemoteSourceTarInstaller(mode, name, test=None,
    params=None)
    Bases: virttest.qemu_installer.QEMUBaseInstaller, virttest.base_installer.RemoteSourceTari
    Installer that deals with source code on remote tarballs
```

virttest.qemu_io module

```
class virttest.qemu_io.QemuIO(test, params, image_name, blkdebug_cfg='', prompt='qemu-
    io>\s*$', log_filename=None, io_options='', log_func=None)
    Bases: object
    A class for execute qemu-io command

close()
    Clean up

cmd_output(command)
    Run a command in qemu-io

get_cmd_line(ignore_option=[], essential_option=[], forbid_option=[])
    Generate the command line for qemu-io from the parameters :params ignore_option: list for the options
    should not in command :params essential_option: list for the essential options :params forbid_option: list
    for the option should not in command :return: qemu-io command line

exception virttest.qemu_io.QemuIOParamError
    Bases: exceptions.Exception
    Parameter Error for qemu-io command

class virttest.qemu_io.QemuIOShellSession(test, params, image_name, blkdebug_cfg='',
    prompt='qemu+-io>\s*$', log_filename=None,
    io_options='', log_func=None)
    Bases: virttest.qemu_io.QemuIO
    Use a shell session to execute qemu-io command

close()
    Close the shell session for qemu-io

cmd_output(*args, **kwargs)
    Get output from shell session. If the create flag is True, init the shell session and set the create flag to False.
    :param command: command to execute in qemu-io :param timeout: timeout for execute the command
```

```
class virttest.qemu_io.QemuIOSystem(test, params, image_name, blkdebug_cfg='',
                                     prompt='qemu-io>\s*$', log_filename=None,
                                     io_options='', log_func=None)

Bases: virttest.qemu_io.QemuIO

Run qemu-io with a command line which will return immediately

close()
    To keep the the same interface with QemuIOShellSession

cmd_output(*args, **kwargs)
    Get output from system_output. Add the command to the qemu-io command line with -c and record the
    output in the log file. :param command: command to execute in qemu-io :param timeout: timeout for
    execute the command
```

virttest.qemu_monitor module

Interfaces to the QEMU monitor.

copyright 2008-2010 Red Hat Inc.

```
class virttest.qemu_monitor.HumanMonitor(vm, name, filename, suppress_exceptions=False)
Bases: virttest.qemu_monitor.Monitor

Wraps "human monitor" commands.

CMD_TIMEOUT = 120

PROMPT_TIMEOUT = 60

block_mirror(device, target, speed, sync, format, mode, cmd='drive_mirror', correct=True)
    Start mirror type block device copy job
```

Parameters

- **device** – device ID
- **target** – target image
- **speed** – limited speed, unit is B/s
- **sync** – full copy to target image(unsupport in human monitor)
- **mode** – target image create mode, 'absolute-paths' or 'existing'
- **format** – target image format
- **cmd** – block mirror command
- **correct** – auto correct command, correct by default

Returns The command's output

```
block_reopen(device, new_image_file, image_format, cmd='block_job_complete', correct=True)
    Reopen new target image
```

Parameters

- **device** – device ID
- **new_image_file** – new image file name
- **image_format** – new image file format
- **cmd** – image reopen command

- **correct** – auto correct command, correct by default

Returns The command's output

block_resize (*device*, *size*)

Resize the block device size

Parameters

- **device** – Block device name
- **size** – Block device size need to set to. To keep the same with qmp monitor will use bytes as unit for the block size

Returns Command output

block_stream (*device*, *speed=None*, *base=None*, *cmd='block_stream'*, *correct=True*)

Start block-stream job;

Parameters

- **device** – device ID
- **speed** – int type, limited speed(B/s)
- **base** – base file
- **correct** – auto correct command, correct by default

Returns The command's output

cancel_block_job (*device*, *cmd='block_job_cancel'*, *correct=True*)

Cancel running block stream/mirror job on the device

Parameters

- **device** – device ID
- **correct** – auto correct command, correct by default

Returns The command's output

change_media (*device*, *target*)

Change media of cdrom of drive;

cmd (*cmd*, *timeout=120*, *debug=True*, *fd=None*)

Send command to the monitor.

Parameters

- **cmd** – Command to send to the monitor
- **timeout** – Time duration to wait for the (qemu) prompt to return
- **debug** – Whether to print the commands being sent and responses

Returns Output received from the monitor

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSocketError** – Raised if a socket error occurs
- **MonitorProtocolError** – Raised if the (qemu) prompt cannot be found after sending the command

eject_cdrom (*device*, *force=False*)

Eject media of cdrom and open cdrom door;

get_backingfile (*device*)

Return “backing_file” path of the device

Parameters **device** – device ID

Returns string, backing_file path

get_status ()

getfd (*fd, name*)

Receives a file descriptor

Parameters

- **fd** – File descriptor to pass to QEMU
- **name** – File descriptor name (internal to QEMU)

Returns The command’s output

human_monitor_cmd (*cmd='', timeout=120, debug=True, fd=None*)

Send human monitor command directly

Parameters

- **cmd** – human monitor command.
- **timeout** – Time duration to wait for response
- **debug** – Whether to print the commands being sent and responses
- **fd** – file object or file descriptor to pass

Returns The response to the command

info (*what, debug=True*)

Request info about something and return the output. :param debug: Whether to print the commands being sent and responses

live_snapshot (*device, snapshot_file, snapshot_format='qcow2'*)

Take a live disk snapshot.

Parameters

- **device** – device id of base image
- **snapshot_file** – image file name of snapshot
- **snapshot_format** – image format of snapshot

Returns The response to the command

migrate (*uri, full_copy=False, incremental_copy=False, wait=False*)

Migrate.

Parameters

- **uri** – destination URI
- **full_copy** – If true, migrate with full disk copy
- **incremental_copy** – If true, migrate with incremental disk copy
- **wait** – If true, wait for completion

Returns The command’s output

migrate_set_downtime (*value*)

Set maximum tolerated downtime (in seconds) for migration.

Parameters **value** – maximum downtime (in seconds)

Returns The command's output

migrate_set_speed (*value*)

Set maximum speed (in bytes/sec) for migrations.

Parameters **value** – Speed in bytes/sec

Returns The command's output

mouse_button (*state*)

Set mouse button state.

Parameters **state** – Button state (1=L, 2=M, 4=R)

Returns The command's output

mouse_move (*dx*, *dy*)

Move mouse.

Parameters

- **dx** – X amount
- **dy** – Y amount

Returns The command's output

nmi ()

Inject a NMI on all guest's CPUs.

query (*what*)

Alias for info.

query_block_job (*device*)

Get block job status on the device

Parameters **device** – device ID

Returns dict about job info, return empty dict if no active job

quit ()

Send “quit” without waiting for output.

screendump (*filename*, *debug=True*)

Request a screendump.

Parameters **filename** – Location for the screendump

Returns The command's output

send_args_cmd (*cmdlines*, *timeout=120*, *convert=True*)

Send a command with/without parameters and return its output. Have same effect with cmd function. Implemented under the same name for both the human and QMP monitors. Command with parameters should in following format e.g.: ‘memsave val=0 size=10240 filename=memsave’ Command without parameter: ‘sendkey ctrl-alt-f1’

Parameters

- **cmdlines** – Commands send to qemu which is separated by ”;”. For command with parameters command should send in a string with this format: \$command \$arg_name=\$arg_value \$arg_name=\$arg_value
- **timeout** – Time duration to wait for (qemu) prompt after command
- **convert** – If command need to convert. For commands such as: \$command \$arg_value

Returns The output of the command

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSendError** – Raised if the command cannot be sent
- **MonitorProtocolError** – Raised if the (qemu) prompt cannot be found after sending the command

sendkey (*keyst*, *hold_time=1*)
Send key combination to VM.

Parameters

- **keyst** – Key combination string
- **hold_time** – Hold time in ms (should normally stay 1 ms)

Returns The command's output

set_block_job_speed (*device*, *speed=0*, *cmd='block_job_set_speed'*, *correct=True*)
Set limited speed for running job on the device

Parameters

- **device** – device ID
- **speed** – int type, limited speed(B/s)
- **correct** – auto correct command, correct by default

Returns The command's output

set_link (*name*, *up*)
Set link up/down.

Parameters

- **name** – Link name
- **up** – Bool value, True=set up this link, False=Set down this link

Returns The response to the command

system_wakeup ()
Wakeup suspended guest.

verify_responsive ()
Make sure the monitor is responsive by sending a command.

verify_status (*status*)
Verify VM status

Parameters **status** – Optional VM status, 'running' or 'paused'

Returns return True if VM status is same as we expected

class virttest.qemu_monitor.**Monitor** (*vm*, *name*, *filename*)
Common code for monitor classes.

ACQUIRE_LOCK_TIMEOUT = 20

CONNECT_TIMEOUT = 30

DATA_AVAILABLE_TIMEOUT = 0

close()

Close the connection to the monitor and its log file.

correct(cmd)

Automatic conversion “-” and “_” in commands if the translate command is supported commands;

human_monitor_cmd(cmd=' ', timeout=None, debug=True, fd=None)

Send HMP command

This method allows code to send HMP commands without the need to check if the monitor is QMPMonitor or HumanMonitor.

Parameters

- **cmd** – human monitor command.
- **timeout** – Time duration to wait for response
- **debug** – Whether to print the commands being sent and responses
- **fd** – file object or file descriptor to pass

Returns The response to the command

info(what, debug=True)

Request info about something and return the response.

info_block(debug=True)

Request info about blocks and return dict of parsed results :return: Dict of disk parameters

info_numa()

Run ‘info numa’ command and parse returned information

Returns An array of (ram, cpus) tuples, where ram is the RAM size in MB and cpus is a set of CPU numbers

is_responsive()

Return True iff the monitor is responsive.

classmethod parse_info_numa(r)

Parse ‘info numa’ output

See info_numa() for information about the return value.

re_numa_node_info = <_sre.SRE_Pattern object>

re_numa_nodes = <_sre.SRE_Pattern object>

verify_supported_cmd(cmd)

Verify whether cmd is supported by monitor. If not, raise a MonitorNotSupportedCmdError Exception.

Parameters cmd – The cmd string need to verify.

exception virttest.gemu_monitor.MonitorConnectError(monitor_name)

Bases: *virttest.gemu_monitor.MonitorError*

exception virttest.gemu_monitor.MonitorError

Bases: *exceptions.Exception*

exception virttest.gemu_monitor.MonitorLockError

Bases: *virttest.gemu_monitor.MonitorError*

exception virttest.gemu_monitor.MonitorNotSupportedCmdError(monitor, cmd)

Bases: *virttest.gemu_monitor.MonitorNotSupportedError*

exception `virttest.qemu_monitor.MonitorNotSupportedError`

Bases: `virttest.qemu_monitor.MonitorError`

exception `virttest.qemu_monitor.MonitorProtocolError`

Bases: `virttest.qemu_monitor.MonitorError`

exception `virttest.qemu_monitor.MonitorSocketError` (*msg, e*)

Bases: `virttest.qemu_monitor.MonitorError`

exception `virttest.qemu_monitor.QMPCmdError` (*cmd, qmp_args, data*)

Bases: `virttest.qemu_monitor.MonitorError`

class `virttest.qemu_monitor.QMPMonitor` (*vm, name, filename, suppress_exceptions=False*)

Bases: `virttest.qemu_monitor.Monitor`

Wraps QMP monitor commands.

CMD_TIMEOUT = 120

PROMPT_TIMEOUT = 60

READ_OBJECTS_TIMEOUT = 5

RESPONSE_TIMEOUT = 120

block_mirror (*device, target, speed, sync, format, mode, cmd='drive-mirror', correct=True*)

Start mirror type block device copy job

Parameters

- **device** – device ID
- **target** – target image
- **speed** – limited speed, unit is B/s
- **sync** – what parts of the disk image should be copied to the destination;
- **mode** – ‘absolute-paths’ or ‘existing’
- **format** – target image format
- **cmd** – block mirror command
- **correct** – auto correct command, correct by default

Returns The command’s output

block_reopen (*device, new_image_file, image_format, cmd='block-job-complete', correct=True*)

Reopen new target image;

Parameters

- **device** – device ID
- **new_image_file** – new image file name
- **image_format** – new image file format
- **cmd** – image reopen command
- **correct** – auto correct command, correct by default

Returns the command’s output

block_resize (*device, size*)

Resize the block device size

Parameters

- **device** – Block device name
- **size** – Block device size need to set to. Unit is bytes.

Returns Command output

block_stream (*device, speed=None, base=None, cmd='block-stream', correct=True*)

Start block-stream job;

Parameters

- **device** – device ID
- **speed** – int type, limited speed(B/s)
- **base** – base file
- **correct** – auto correct command, correct by default

Returns The command's output

cancel_block_job (*device, cmd='block-job-cancel', correct=True*)

Cancel running block stream/mirror job on the device

Parameters

- **device** – device ID
- **correct** – auto correct command, correct by default

Returns The command's output

change_media (*device, target*)

Change media of cdrom of drive;

clear_event (*name*)

Clear a kinds of events in events list only.

Raises MonitorLockError – Raised if the lock cannot be acquired

clear_events ()

Clear the list of asynchronous events.

Raises MonitorLockError – Raised if the lock cannot be acquired

cmd (*cmd, args=None, timeout=120, debug=True, fd=None*)

Send a QMP monitor command and return the response.

Note: an id is automatically assigned to the command and the response is checked for the presence of the same id.

Parameters

- **cmd** – Command to send
- **args** – A dict containing command arguments, or None
- **timeout** – Time duration to wait for response
- **debug** – Whether to print the commands being sent and responses
- **fd** – file object or file descriptor to pass

Returns The response received

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired

- **MonitorSocketError** – Raised if a socket error occurs
- **MonitorProtocolError** – Raised if no response is received
- **QMPCmdError** – Raised if the response is an error message (the exception's args are (cmd, args, data) where data is the error data)

cmd_obj (*obj*, *timeout=120*)

Transform a Python object to JSON, send the resulting string to the QMP monitor, and return the response. Unlike cmd(), return the raw response dict without performing any checks on it.

Parameters

- **obj** – The object to send
- **timeout** – Time duration to wait for response

Returns The response received

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSocketError** – Raised if a socket error occurs
- **MonitorProtocolError** – Raised if no response is received

cmd_qmp (*cmd*, *args=None*, *q_id=None*, *timeout=120*)

Build a QMP command from the passed arguments, send it to the monitor and return the response. Unlike cmd(), return the raw response dict without performing any checks on it.

Parameters

- **cmd** – Command to send
- **args** – A dict containing command arguments, or None
- **id** – An id for the command, or None
- **timeout** – Time duration to wait for response

Returns The response received

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSocketError** – Raised if a socket error occurs
- **MonitorProtocolError** – Raised if no response is received

cmd_raw (*data*, *timeout=120*)

Send a raw string to the QMP monitor and return the response. Unlike cmd(), return the raw response dict without performing any checks on it.

Parameters

- **data** – The data to send
- **timeout** – Time duration to wait for response

Returns The response received

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSocketError** – Raised if a socket error occurs

- **MonitorProtocolError** – Raised if no response is received

eject_cdrom (*device*, *force=False*)

Eject media of cdrom and open cdrom door;

get_backingfile (*device*)

Return “backing_file” path of the device

Parameters **device** – device ID

Returns string, backing_file path

get_event (*name*)

Look for an event with the given name in the list of events.

Parameters **name** – The name of the event to look for (e.g. ‘RESET’)

Returns An event object or None if none is found

get_events ()

Return a list of the asynchronous events received since the last clear_events() call.

Returns A list of events (the objects returned have an “event” key)

Raises **MonitorLockError** – Raised if the lock cannot be acquired

get_greeting ()

Return QMP greeting message.

get_status ()

Get VM status.

Returns return VM status

getfd (*fd*, *name*)

Receives a file descriptor

Parameters

- **fd** – File descriptor to pass to QEMU
- **name** – File descriptor name (internal to QEMU)

Returns The response to the command

human_monitor_cmd (*cmd=''*, *timeout=120*, *debug=True*, *fd=None*)

Run human monitor command in QMP through human-monitor-command

Parameters

- **cmd** – human monitor command.
- **timeout** – Time duration to wait for response
- **debug** – Whether to print the commands being sent and responses
- **fd** – file object or file descriptor to pass

Returns The response to the command

info (*what*, *debug=True*)

Request info about something and return the response.

live_snapshot (*device*, *snapshot_file*, *snapshot_format='qcow2'*)

Take a live disk snapshot.

Parameters

- **device** – device id of base image
- **snapshot_file** – image file name of snapshot
- **snapshot_format** – image format of snapshot

Returns The response to the command

migrate (*uri*, *full_copy=False*, *incremental_copy=False*, *wait=False*)
Migrate.

Parameters

- **uri** – destination URI
- **full_copy** – If true, migrate with full disk copy
- **incremental_copy** – If true, migrate with incremental disk copy
- **wait** – If true, wait for completion

Returns The response to the command

migrate_set_downtime (*value*)
Set maximum tolerated downtime (in seconds) for migration.

Parameters **value** – maximum downtime (in seconds)

Returns The command's output

migrate_set_speed (*value*)
Set maximum speed (in bytes/sec) for migrations.

Parameters **value** – Speed in bytes/sec

Returns The response to the command

nmi ()
Inject a NMI on all guest's CPUs.

query (*what*, *debug=True*)
Alias for info.

query_block_job (*device*)
Get block job status on the device

Parameters **device** – device ID

Returns dict about job info, return empty dict if no active job

quit ()
Send “quit” and return the response.

screendump (*filename*, *debug=True*)
Request a screendump.

Parameters

- **filename** – Location for the screendump
- **debug** – Whether to print the commands being sent and responses

Returns The response to the command

send_args_cmd (*cmdlines*, *timeout=120*, *convert=True*)
Send a command with/without parameters and return its output. Have same effect with cmd function. Implemented under the same name for both the human and QMP monitors. Command with parameters should

in following format e.g.: 'memsave val=0 size=10240 filename=memsave' Command without parameter: 'query-vnc'

Parameters

- **cmdlines** – Commands send to qemu which is separated by ";". For command with parameters command should send in a string with this format: \$command \$arg_name=\$arg_value \$arg_name=\$arg_value
- **timeout** – Time duration to wait for (qemu) prompt after command
- **convert** – If command need to convert. For commands not in standard format such as: \$command \$arg_value

Returns The response to the command

Raises

- **MonitorLockError** – Raised if the lock cannot be acquired
- **MonitorSendError** – Raised if the command cannot be sent
- **MonitorProtocolError** – Raised if no response is received

sendkey (*keyst*, *hold_time=1*)
Send key combination to VM.

Parameters

- **keyst** – Key combination string
- **hold_time** – Hold time in ms (should normally stay 1 ms)

Returns The response to the command

set_block_job_speed (*device*, *speed=0*, *cmd='block-job-set-speed'*, *correct=True*)
Set limited speed for running job on the device

Parameters

- **device** – device ID
- **speed** – int type, limited speed(B/s)
- **correct** – auto correct command, correct by default

Returns The command's output

set_link (*name*, *up*)
Set link up/down.

Parameters

- **name** – Link name
- **up** – Bool value, True=set up this link, False=Set down this link

Returns The response to the command

system_wakeup ()
Wakeup suspended guest.

verify_responsive ()
Make sure the monitor is responsive by sending a command.

verify_status (*status*)
Verify VM status

Parameters **status** – Optional VM status, ‘running’ or ‘paused’

Returns return True if VM status is same as we expected

verify_supported_hmp_cmd (*cmd*)

Verify whether *cmd* is supported by hmp monitor. If not, raise a `MonitorNotSupportedCmdError` Exception.

Parameters **cmd** – The *cmd* string need to verify.

`virttest.gemu_monitor.create_monitor` (*vm*, *monitor_name*, *monitor_params*)

Create monitor object and connect to the monitor socket.

Parameters

- **vm** – The VM object which has the monitor.
- **monitor_name** – The name of this monitor object.
- **monitor_params** – The dict for creating this monitor object.

`virttest.gemu_monitor.get_monitor_filename` (*vm*, *monitor_name*)

Return the filename corresponding to a given monitor name.

Parameters

- **vm** – The VM object which has the monitor.
- **monitor_name** – The monitor name.

Returns The string of socket file name for qemu monitor.

`virttest.gemu_monitor.get_monitor_filenames` (*vm*)

Return a list of all monitor filenames (as specified in the VM’s params).

Parameters **vm** – The VM object which has the monitors.

`virttest.gemu_monitor.wait_for_create_monitor` (*vm*, *monitor_name*, *monitor_params*,
timeout)

Wait for the progress of creating monitor object. This function will retry to create the Monitor object until timeout.

Parameters

- **vm** – The VM object which has the monitor.
- **monitor_name** – The name of this monitor object.
- **monitor_params** – The dict for creating this monitor object.
- **timeout** – Time to wait for creating this monitor object.

virttest.gemu_monitor_unittest module

class `virttest.gemu_monitor_unittest.InfoBlocks` (*methodName*=‘runTest’)

Bases: `unittest.case.TestCase`

testParseBlocks ()

class `virttest.gemu_monitor_unittest.InfoNumaTests` (*methodName*=‘runTest’)

Bases: `unittest.case.TestCase`

testTwoNodes ()

testZeroNodes ()

class `virttest.qemu_monitor_unittest.MockMonitor`
Bases: `virttest.qemu_monitor.Monitor`
Dummy class inherited from `qemu_monitor.HumanMonitor`

virttest.qemu_qtree module

Utility classes and functions to handle KVM Qtree parsing and verification.

author Lukas Doktor <ldoktor@redhat.com>

copyright 2012 Red Hat Inc.

exception `virttest.qemu_qtree.IncompatibleTypeError` (*prop, desired_type, value*)
Bases: `exceptions.TypeError`

class `virttest.qemu_qtree.QtreeBus`
Bases: `virttest.qemu_qtree.QtreeNode`

bus: qtree object

add_child (*child*)

guess_type ()

class `virttest.qemu_qtree.QtreeContainer`
Bases: `object`

Container for Qtree

get_nodes ()

Returns flat list of all qtree nodes (last one is main-system-bus)

get_qtree ()

Returns root of qtree

parse_info_qtree (*info*)

Parses 'info qtree' output. Creates list of `self.nodes`. Last node is the main-system-bus (whole qtree)

class `virttest.qemu_qtree.QtreeDev`
Bases: `virttest.qemu_qtree.QtreeNode`

dev: qtree object

add_child (*child*)

guess_type ()

class `virttest.qemu_qtree.QtreeDisk`
Bases: `virttest.qemu_qtree.QtreeDev`

qtree disk object

generate_params ()

get_block ()

get_qname ()

set_block_prop (*prop, value*)

update_block_prop (*prop, value*)

```
class virttest.qemu_qtree.QtreeDisksContainer(nodes)
    Bases: object

    Container for QtreeDisks verification. It's necessary because some information can be verified only from infor-
    mations about all disks, not only from single disk.

    check_disk_params(params)
        Check gathered info from qtree/block with params :param params: autotest params :return: number of
        errors

    check_guests_proc_scsi(info)
        Check info from guest's /proc/scsi/scsi file with qtree/block info

        Note Not tested disks are of different type (virtio_blk, ...)

        Parameters info – contents of guest's /proc/scsi/scsi file

        Returns Number of disks missing in guest os, disks missing in qtree, disks not tested from qtree,
        disks not tested from guest

    generate_params()
        Generate params from current self.qtree and self.block info. :note: disk name is not yet the one from
        autotest params :return: number of fails

    parse_info_block(info)
        Extracts all information about self.disks and fills them in.

        Parameters info – output of info block command

        Returns self.disks defined in qtree but not in info block, self.disks defined in
        block info but not in qtree

class virttest.qemu_qtree.QtreeNode
    Bases: object

    Generic Qtree node

    add_child(child)

    generate_params()

    get_children()

    get_params()

    get_parent()

    get_qtree()

    guess_type()
        Detect type of this object from qtree props

    replace_child(oldchild, newchild)

    set_parent(parent)

    set_qtree(qtree)

    set_qtree_prop(prop, value)

    str_qtree()

    str_short()

    update_params(param, value)

    update_qtree_prop(prop, value)
```

```
verify()
```

virttest.qemu_qtree_unittest module

This is a unittest for qemu_qtree library.

author Lukas Doktor <ldoktor@redhat.com>

copyright 2012 Red Hat, Inc.

```
class virttest.qemu_qtree_unittest.KvmQtreeClassTest (methodName='runTest')
    Bases: unittest.case.TestCase
        Additional tests for qemu_qtree classes

    test_qtree_bus_bus ()
        Bus' child can't be Bus()

    test_qtree_dev_dev ()
        Dev's child can't be Dev()

    test_qtree_disk_missing_filename ()
        in info_block must contain info about file or backing_file

class virttest.qemu_qtree_unittest.ParamsDict
    Bases: dict
        params like dictionary

    object_params (obj)

    objects (item)

class virttest.qemu_qtree_unittest.QtreeContainerTest (methodName='runTest')
    Bases: unittest.case.TestCase
        QtreeContainer tests

    test_bad_qtree ()
        Incorrect qtree

    test_qtree ()
        Correct workflow

class virttest.qemu_qtree_unittest.QtreeDiskContainerTest (methodName='runTest')
    Bases: unittest.case.TestCase
        QtreeDiskContainer tests

    setUp ()

    tearDown ()

    test_check_params ()
        Correct workflow

    test_check_params_bad ()
        Whole workflow with bad data

virttest.qemu_qtree_unittest.combine (first, second, offset)
    Add string line-by-line with offset*OFFSET_PER_LEVEL
```

virttest.qemu_storage module

Classes and functions to handle block/disk images for KVM.

This exports:

- two functions for get image/blkdebug filename
- class for image operates and basic parameters

class `virttest.qemu_storage.Iscsidev` (*params, root_dir, tag*)
 Bases: `virttest.storage.Iscsidev`

Class for handle iscsi devices for VM

cleanup ()

Logout the iscsi target and clean up the config and image.

setup ()

Access the iscsi target. And return the local raw device name.

class `virttest.qemu_storage.LVMdev` (*params, root_dir, tag*)
 Bases: `virttest.storage.LVMdev`

Class for handle lvm devices for VM

cleanup ()

Cleanup useless volumes;

setup ()

Get logical volume path;

class `virttest.qemu_storage.QemuImg` (*params, root_dir, tag*)
 Bases: `virttest.storage.QemuImg`

KVM class for handling operations of disk/block images.

check_image (*params, root_dir*)

Check an image using the appropriate tools for each virt backend.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.

Note params should contain: `image_name` – the name of the image file, without extension `image_format` – the format of the image (qcow2, raw etc)

Raises `VMImageCheckError` – In case qemu-img check fails on the image.

commit (*params={}, cache_mode=None*)

Commit image to it's base file

Parameters `cache_mode` – the cache mode used to write the output disk image, the valid options are: 'none', 'writeback' (default), 'writethrough', 'directsync' and 'unsafe'.

compare_images (*image1, image2, verbose=True*)

Compare 2 images using the appropriate tools for each virt backend.

Parameters

- **image1** – image path of first image
- **image2** – image path of second image
- **verbose** – Record output in debug file or not

convert (*params*, *root_dir*, *cache_mode=None*)

Convert image

Parameters

- **params** – dictionary containing the test parameters
- **root_dir** – dir for save the convert image
- **cache_mode** – The cache mode used to write the output disk image. Valid options are: `none`, `writeback` (default), `writethrough`, `directsync` and `unsafe`.

Note *params* should contain:

convert_image_tag the image name of the convert image

convert_filename the name of the image after convert

convert_fmt the format after convert

compressed indicates that target image must be compressed

encrypted there are two value “off” and “on”, default value is “off”

create (**args*, ***kwargs*)

Create an image using `qemu_img` or `dd`.

Parameters

- **params** – Dictionary containing the test parameters.
- **ignore_errors** – Whether to ignore errors on the image creation cmd.

Note *params* should contain:

image_name name of the image file, without extension

image_format format of the image (`qcow2`, `raw` etc)

image_cluster_size (optional) cluster size for the image

image_size requested size of the image (a string `qemu-img` can understand, such as ‘10G’)

create_with_dd use `dd` to create the image (raw format only)

base_image(optional) the base image name when create snapshot

base_format(optional) the format of base image

encrypted(optional) if the image is encrypted, allowed values: `on` and `off`. Default is “off”

preallocated(optional) if preallocation when create image, allowed values: `off`, `metadata`. Default is “off”

Returns tuple (path to the image created, `utils.CmdResult` object containing the result of the creation command).

get_format ()

Get the fimage file format.

info ()

Run `qemu-img info` command on image file and return its output.

rebase (*params*, *cache_mode=None*)

Rebase image.

Parameters

- **params** – dictionary containing the test parameters

- **cache_mode** – the cache mode used to write the output disk image, the valid options are: ‘none’, ‘writeback’ (default), ‘writethrough’, ‘directsync’ and ‘unsafe’.

Note params should contain:

cmd qemu-img cmd

snapshot_img the snapshot name

base_img base image name

base_fmt base image format

snapshot_fmt the snapshot format

mode there are two value, “safe” and “unsafe”, default is “safe”

remove()

Remove an image file.

snapshot_apply()

Apply a snapshot image.

Note params should contain: **snapshot_image_name** – the name of snapshot image file

snapshot_create()

Create a snapshot image.

Note params should contain: **snapshot_image_name** – the name of snapshot image file

snapshot_del (*blkdebug_cfg*='')

Delete a snapshot image.

Parameters **blkdebug_cfg** – The configure file of blkdebug

Note params should contain: **snapshot_image_name** – the name of snapshot image file

snapshot_list()

List all snapshots in the given image

support_cmd (*cmd*)

Verifies whether qemu-img supports command cmd.

Parameters **cmd** – Command string.

virttest.qemu_virtio_port module

Interfaces and helpers for the virtio_serial ports.

copyright 2012 Red Hat Inc.

class virttest.qemu_virtio_port.**GuestWorker** (*vm*)

Bases: `object`

Class for executing “virtio_console_guest” script on guest

cleanup()

Cleanup ports and quit the worker

cleanup_ports()

Clean state of all ports and set port to default state.

Default state: No data on port or in port buffer. Read mode = blocking.

cmd (*cmd, timeout=10, patterns=None*)

Wrapper around the self.cmd command which executes the command on guest. Unlike self._cmd command when the command fails it raises the test error. :param command: Command that will be executed. :param timeout: Timeout used to verify expected output. :return: Tuple (match index, data)

read_nonblocking (*internal_timeout=None, timeout=None*)

Reads-out all remaining output from GuestWorker.

Parameters

- **internal_timeout** – Time (seconds) to wait before we give up reading from the child process, or None to use the default value.
- **timeout** – Timeout for reading child process output.

reconnect (*vm, timeout=10*)

Reconnect to guest_worker (eg. after migration) :param vm: New VM object

safe_exit_loopback_threads (*send_pts, recv_pts*)

Safely executes on_guest(“virt.exit_threads()”) using workaround of the stuck thread in loopback in mode=virt.LOOP_NONE. :param send_pts: list of possible send sockets we need to work around. :param recv_pts: list of possible recv sockets we need to read-out.

class virttest.qemu_virtio_port.**ThRecv** (*port, event, blocklen=1024, quiet=False*)

Bases: `threading.Thread`

Receives data and throws it away.

run ()

class virttest.qemu_virtio_port.**ThRecvCheck** (*port, buff, exit_event, blocklen=1024, sendlen=0, migrate_event=None, debug=None*)

Bases: `threading.Thread`

Random data receiver/checker thread.

reload_loss_idx ()

This function reloads the acceptable loss to the original value (Reload the self.sendidx to self.sendlen) :note: This function is automatically called during port reconnection.

run ()

Pick the right mode and execute it

run_debug ()

viz run_normal. Additionally it stores last n verified characters and in case of failures it quickly receive enough data to verify failure or allowed loss and then analyze this data. It provides more info about the situation. Unlike normal run this one supports booth - loss and duplications. It’s not friendly to data corruption.

run_normal ()

Receives data and verifies, whether they match the self.buff (queue). It allow data loss up to self.sendidx which can be manually loaded after host socket reconnection or you can overwrite this value from other thread.

class virttest.qemu_virtio_port.**ThSend** (*port, data, exit_event, quiet=False*)

Bases: `threading.Thread`

Random data sender thread.

run ()

class virttest.qemu_virtio_port.**ThSendCheck** (*port, exit_event, queues, blocklen=1024, migrate_event=None, reduced_set=False*)

Bases: `threading.Thread`

Random data sender thread.

run ()

class `virttest.qemu_virtio_port.VirtioConsole` (*qemu_id, name, hostfile*)

Bases: `virttest.qemu_virtio_port._VirtioPort`

Class for handling virtio-console

exception `virttest.qemu_virtio_port.VirtioPortException`

Bases: `exceptions.Exception`

General virtio_port exception

exception `virttest.qemu_virtio_port.VirtioPortFatalException`

Bases: `virttest.qemu_virtio_port.VirtioPortException`

Fatal virtio_port exception

class `virttest.qemu_virtio_port.VirtioSerial` (*qemu_id, name, hostfile*)

Bases: `virttest.qemu_virtio_port._VirtioPort`

Class for handling virtio-serialport

virttest.qemu_vm module

Utility classes and functions to handle Virtual Machine creation using qemu.

copyright 2008-2009, 2014 Red Hat Inc.

exception `virttest.qemu_vm.ImageUnbootableError` (*name*)

Bases: `virttest.virt_vm.VMError`

exception `virttest.qemu_vm.KVMInternalError` (**args*)

Bases: `virttest.virt_vm.VMError`

exception `virttest.qemu_vm.QemuSegFaultError` (*crash_message*)

Bases: `virttest.virt_vm.VMError`

class `virttest.qemu_vm.VM` (*name, params, root_dir, address_cache, state=None*)

Bases: `virttest.virt_vm.BaseVM`

This class handles all basic VM operations.

CLOSE_SESSION_TIMEOUT = 30

MIGRATION_PROTOS = ['rdma', 'x-rdma', 'tcp', 'unix', 'exec', 'fd']

activate_netdev (**args, **kwargs*)

Activate an inactive host-side networking device

Raise IndexError if nic doesn't exist

Raise VMUnknownNetTypeError: if nettype is unset/unsupported

Raise IOError if TAP device node cannot be opened

Raise VMAddNetDevError: if operation failed

activate_nic (**args, **kwargs*)

Activate an VM's inactive NIC device and verify state

Parameters **nic_index_or_name** – name or index number for existing NIC

add_netdev (**args, **kwargs*)

Hotplug a netdev device.

Parameters **params** – NIC info. dict.

Returns netdev_id

add_nic (**params)

Add new or setup existing NIC, optionally creating netdev if None

Parameters

- **params** – Parameters to set
- **nic_name** – Name for existing or new device
- **nic_model** – Model name to emulate
- **netdev_id** – Existing qemu net device ID name, None to create new
- **mac** – Optional MAC address, None to randomly generate.

block_mirror (device, target, speed, sync, format, mode='absolute-paths', correct=True)

Mirror block device to target file;

Parameters

- **device** – device ID
- **target** – destination image file name;
- **speed** – max limited speed, default unit is B/s;
- **sync** – what parts of the disk image should be copied to the destination;
- **mode** – new image open mode
- **format** – target image format
- **correct** – auto correct cmd, correct by default

block_reopen (device, new_image, format='qcow2', correct=True)

Reopen a new image, no need to do this step in rhel7 host

Parameters

- **device** – device ID
- **new_image** – new image filename
- **format** – new image format
- **correct** – auto correct cmd, correct by default

block_stream (device, speed, base=None, correct=True)

start to stream block device, aka merge snapshot;

Parameters

- **device** – device ID;
- **speed** – limited speed, default unit B/s;
- **base** – base file;
- **correct** – auto correct cmd, correct by default

cancel_block_job (device, correct=True)

cancel active job on the image_file

Parameters

- **device** – device ID

- **correct** – auto correct cmd, correct by default

catch_monitor

Return the catch monitor object, selected by the parameter `catch_monitor`. If `catch_monitor` isn't defined or it refers to a nonexistent monitor, return the last monitor. If no monitors exist, return `None`.

change_media (*device*, *target*)

Change media of cdrom;

Parameters

- **device** – Device ID;
- **target** – new media file;

check_block_locked (*value*)

Check whether specified block device is locked or not. Return `True`, if device is locked, else `False`.

Parameters

- **vm** – VM object
- **value** – Parameter that can specify block device. Can be any possible identification of a device, Such as device name/image file name/...

Returns `True` if device is locked, `False` if device is unlocked.

cleanup_serial_console ()

Close serial console and associated log file

clone (*name=None*, *params=None*, *root_dir=None*, *address_cache=None*, *copy_state=False*)

Return a clone of the VM object with optionally modified parameters. The clone is initially not alive and needs to be started using `create()`. Any parameters not passed to this function are copied from the source VM.

Parameters

- **name** – Optional new VM name
- **params** – Optional new VM creation parameters
- **root_dir** – Optional new base directory for relative filenames
- **address_cache** – A dict that maps MAC addresses to IP addresses
- **copy_state** – If `True`, copy the original VM's state to the clone. Mainly useful for `make_qemu_command()`.

create (**args*, ***kwargs*)

Start the VM by running a `qemu` command. All parameters are optional. If `name`, `params` or `root_dir` are not supplied, the respective values stored as class attributes are used.

Parameters

- **name** – The name of the object
- **params** – A dict containing VM params
- **root_dir** – Base directory for relative filenames
- **migration_mode** – If supplied, start VM for incoming migration using this protocol (either `'rdma'`, `'x-rdma'`, `'rdma'`, `'tcp'`, `'unix'` or `'exec'`)
- **migration_exec_cmd** – Command to embed in `'-incoming "exec: ..."'` (e.g. `'gzip -c -d filename'`) if `migration_mode` is `'exec'` default to listening on a random TCP port
- **migration_fd** – Open descriptor from machine should migrate.

- **mac_source** – A VM object from which to copy MAC addresses. If not specified, new addresses will be generated.

Raises

- **VMCreateError** – If qemu terminates unexpectedly
- **VMKVMInitError** – If KVM initialization fails
- **VMHugePageError** – If hugepage initialization fails
- **VMImageMissingError** – If a CD image is missing
- **VMHashMismatchError** – If a CD image hash has doesn't match the expected hash
- **VMBadPATypeError** – If an unsupported PCI assignment type is requested
- **VMPAError** – If no PCI assignable devices could be assigned
- **TAPCreationError** – If fail to create tap fd
- **BRAddIfError** – If fail to add a tap to a bridge
- **TAPBringUpError** – If fail to bring up a tap
- **PrivateBridgeError** – If fail to bring the private bridge

create_serial_console()

Establish a session with the serial console.

Let's consider the first serial port as serial console. Note: requires a version of netcat that supports -U

create_virtio_console()

Establish a session with the serial console.

deactivate_netdev(*args, **kwargs)

Reverses what activate_netdev() did

Param nic_index_or_name: name or index number for existing NIC

deactivate_nic(*args, **kwargs)

Reverses what activate_nic did

Parameters

- **nic_index_or_name** – name or index number for existing NIC
- **wait** – Time test will wait for the guest to unplug the device

del_netdev(*args, **kwargs)

Remove netdev info. from nic on VM, does not deactivate.

Param nic_index_or_name: name or index number for existing NIC

del_nic(*args, **kwargs)

Undefine nic parameters, reverses what add_nic did.

Parameters

- **nic_index_or_name** – name or index number for existing NIC
- **wait** – Time test will wait for the guest to unplug the device

destroy(gracefully=True, free_mac_addresses=True)

Destroy the VM.

If gracefully is True, first attempt to shutdown the VM with a shell command. Then, attempt to destroy the VM via the monitor with a 'quit' command. If that fails, send SIGKILL to the qemu process.

Parameters

- **gracefully** – If True, an attempt will be made to end the VM using a shell command before trying to end the qemu process with a ‘quit’ or a kill signal.
- **free_mac_addresses** – If True, the MAC addresses used by the VM will be freed.

eject_cdrom (*device*, *force=False*)

Eject cdrom and open door of the CDROM;

Parameters

- **device** – device ID;
- **force** – force eject or not;

get_block (*p_dict={}*)

Get specified block device from monitor’s info block command. The block device is defined by parameter in *p_dict*.

Parameters **p_dict** – Dictionary that contains parameters and its value used to define specified block device.

Returns Matched block device name, None when not find any device.

get_block_old (*blocks_info*, *p_dict={}*)

Get specified block device from monitor’s info block command. The block device is defined by parameter in *p_dict*.

Parameters

- **p_dict** – Dictionary that contains parameters and its value used to define specified block device.
- **blocks_info** – the results of monitor command ‘info block’

Returns Matched block device name, None when not find any device.

get_ifname (*nic_index=0*)

Return the ifname of a bridge/tap device associated with a NIC.

Parameters **nic_index** – Index of the NIC

get_job_status (*device*)

get block job info;

Parameters **device** – device ID

get_monitors_by_type (*mon_type*)

Return list of monitors of *mon_type* type. :param *mon_type*: desired monitor type (qmp, human)

get_peer (*netid*)

Return the peer of netdev or network device.

Parameters **netid** – id of netdev or device

Returns id of the peer device otherwise None

get_pid ()

Return the VM’s PID. If the VM is dead return None.

Note This works under the assumption that `self.process.get_pid()`

Returns the PID of the parent shell process.

get_serial_console_filename (*name=None*)

Return the serial console filename.

Parameters **name** – The serial port name.

get_serial_console_filenames ()

Return a list of all serial console filenames (as specified in the VM's params).

get_shared_meminfo ()

Returns the VM's shared memory information.

Returns Shared memory used by VM (MB)

get_shell_pid ()

Return the PID of the parent shell process.

Note This works under the assumption that `self.process.get_pid()`

Returns the PID of the parent shell process.

get_spice_var (*spice_var*)

Returns string value of spice variable of choice or None :param spice_var - spice related variable 'spice_port', ...

get_vcpu_pids (*vcpu_thread_pattern*)

Return the list of vcpu PIDs

Returns the list of vcpu PIDs

get_vhost_threads (*vhost_thread_pattern*)

Return the list of vhost threads PIDs

Parameters **vhost_thread_pattern** (*string*) – a regex to match the vhost threads

Returns a list of vhost threads PIDs

Return type list of integer

get_virtio_port_filenames ()

Get socket file of virtio ports

get_vnc_port ()

Return `self.vnc_port`.

graceful_shutdown (*timeout=60*)

Try to gracefully shut down the VM.

Returns True if VM was successfully shut down, None otherwise.

Note that the VM is not necessarily dead when this function returns True. If QEMU is running in -no-shutdown mode, the QEMU process may be still alive.

hotplug_nic (**args, **kwargs*)

Convenience method wrapper for `add_nic()` and `add_netdev()`.

Returns dict-like object containing nic's details

hotplug_vcpu (**args, **kwargs*)

Hotplug a vcpu, if not assign the `cpu_id`, will use the minimum unused. the function will use the `plug_command` if you assigned it, else the function will use the command automatically generated based on the type of monitor

:param `cpu_id` the `cpu_id` you want hotplug.

hotunplug_nic (**args, **kwargs*)

Convenience method wrapper for `del/deactivate nic` and `netdev`.

is_alive ()

Return True if the VM is alive and its monitor is responsive.

is_dead()

Return True if the qemu process is dead.

is_paused()

Return True if the qemu process is paused ('stop'ed)

live_snapshot (*base_file, snapshot_file, snapshot_format='qcow2'*)

Take a live disk snapshot.

Parameters

- **base_file** – base file name
- **snapshot_file** – snapshot file name
- **snapshot_format** – snapshot file format

Returns File name of disk snapshot.

loadvm (*tag_name*)

Override BaseVM loadvm method

make_create_command (*name=None, params=None, root_dir=None*)

Generate a qemu command line. All parameters are optional. If a parameter is not supplied, the corresponding value stored in the class attributes is used.

Parameters

- **name** – The name of the object
- **params** – A dict containing VM params
- **root_dir** – Base directory for relative filenames

Note The params dict should contain: mem – memory size in MBs cdrom – ISO filename to use with the qemu -cdrom parameter extra_params – a string to append to the qemu command shell_port – port of the remote shell daemon on the guest (SSH, Telnet or the home-made Remote Shell Server) shell_client – client program to use for connecting to the remote shell daemon on the guest (ssh, telnet or nc) x11_display – if specified, the DISPLAY environment variable will be set to this value for the qemu process (useful for SDL rendering) images – a list of image object names, separated by spaces nics – a list of NIC object names, separated by spaces

For each image in images: drive_format – string to pass as 'if' parameter for this image (e.g. ide, scsi) image_snapshot – if yes, pass 'snapshot=on' to qemu for this image image_boot – if yes, pass 'boot=on' to qemu for this image In addition, all parameters required by get_image_filename.

For each NIC in nics: nic_model – string to pass as 'model' parameter for this NIC (e.g. e1000)

mig_cancelled()

mig_failed()

mig_finished()

mig_succeeded()

migrate (**args, **kwargs*)

Migrate the VM.

If the migration is local, the VM object's state is switched with that of the destination VM. Otherwise, the state is switched with that of a dead VM (returned by self.clone()).

Parameters

- **timeout** – Time to wait for migration to complete.
- **protocol** – Migration protocol (as defined in `MIGRATION_PROTOS`)
- **cancel_delay** – If provided, specifies a time duration after which migration will be canceled. Used for testing `migrate_cancel`.
- **offline** – If True, pause the source VM before migration.
- **stable_check** – If True, compare the VM's state after migration to its state before migration and raise an exception if they differ.
- **clean** – If True, delete the saved state files (relevant only if `stable_check` is also True).
- **save_path** – The path for state files.
- **dest_host** – Destination host (defaults to 'localhost').
- **remote_port** – Port to use for remote migration.
- **not_wait_for_migration** – If True migration start but not wait till the end of migration.
- **fd_s** – File descriptor for migration to which source VM write data. Descriptor is closed during the migration.
- **fd_d** – File descriptor for migration from which destination VM read data.
- **migration_exec_cmd_src** – Command to embed in '-incoming "exec: "' (e.g. 'exec:gzzip -c > filename') if `migration_mode` is 'exec' default to listening on a random TCP port
- **migration_exec_cmd_dst** – Command to embed in '-incoming "exec: "' (e.g. 'gzzip -c -d filename') if `migration_mode` is 'exec' default to listening on a random TCP port
- **env** – Dictionary with test environment

monitor

Return the main monitor object, selected by the parameter `main_monitor`. If `main_monitor` isn't defined or it refers to a nonexistent monitor, return the first monitor. If no monitors exist, return `None`.

pause()

Pause the VM operation.

process_info_block(blocks_info)

Process the info block, so that can deal with the new and old qemu format.

Parameters `blocks_info` – the output of qemu command 'info block'

reboot(*args, **kwargs)

Reboot the VM and wait for it to come back up by trying to log in until timeout expires.

Parameters

- **session** – A shell session object or `None`.
- **method** – Reboot method. Can be "shell" (send a shell reboot command) or "system_reset" (send a system_reset monitor command).
- **nic_index** – Index of NIC to access in the VM, when logging in after rebooting.
- **timeout** – Time to wait for login to succeed (after rebooting).
- **serial** – Serial login or not (default is `False`).

Returns A new shell session object.

restore_from_file (*path*)

Override BaseVM restore_from_file method

resume ()

Resume the VM operation in case it's stopped.

save_to_file (*path*)

Override BaseVM save_to_file method

savevm (*tag_name*)

Override BaseVM savevm method

screendump (*filename*, *debug=True*)

send_fd (**args*, ***kwargs*)

Send file descriptor over unix socket to VM.

Parameters

- **fd** – File descriptor.
- **fd_name** – File descriptor identificator in VM.

send_key (*keyst*)

Send a key event to the VM.

Parameters **keyst** – A key event string (e.g. “ctrl-alt-delete”)

set_job_speed (*device*, *speed='0'*, *correct=True*)

set max speed of block job;

Parameters

- **device** – device ID
- **speed** – max speed of block job
- **correct** – auto correct cmd, correct by default

set_link (*netdev_name*, *up*)

Set link up/down.

Parameters

- **name** – Link name
- **up** – Bool value, True=set up this link, False=Set down this link

update_system_dependent_devs ()

update_vga_global_default (*params*, *migrate=None*)

Update VGA global default settings

Parameters

- **params** – dict for create vm
- **migrate** – is vm create for migration

verify_alive ()

Make sure the VM is alive and that the main monitor is responsive.

Raises **VMDeadError** – If the VM is dead

Raise Various monitor exceptions if the monitor is unresponsive

verify_disk_image_bootable ()

verify_kvm_internal_error ()

Verify KVM internal error.

verify_status (*status*)

Check VM status

Parameters **status** – Optional VM status, ‘running’ or ‘paused’

Raises **VMStatusError** – If the VM status is not same as parameter

verify_userspace_crash ()

Verify if the userspace component (qemu) crashed.

wait_for_migration (*timeout*)

wait_for_shutdown (*timeout=60*)

Wait until guest shuts down.

Helps until the VM is shut down by the guest.

Returns True in case the VM was shut down, None otherwise.

Note that the VM is not necessarily dead when this function returns True. If QEMU is running in -no-shutdown mode, the QEMU process may be still alive.

wait_for_status (*status, timeout, first=0.0, step=1.0, text=None*)

Wait until the VM status changes to specified status

Parameters

- **timeout** – Timeout in seconds
- **first** – Time to sleep before first attempt
- **steps** – Time to sleep between attempts in seconds
- **text** – Text to print while waiting, for debug purposes

Returns True in case the status has changed before timeout, otherwise return None.

wait_until_dead (*timeout, first=0.0, step=1.0*)

Wait until VM is dead.

Returns True if VM is dead before timeout, otherwise returns None.

Parameters

- **timeout** – Timeout in seconds
- **first** – Time to sleep before first attempt
- **steps** – Time to sleep between attempts in seconds

wait_until_paused (*timeout*)

Wait until the VM is paused.

Parameters **timeout** – Timeout in seconds.

Returns True in case the VM is paused before timeout, otherwise return None.

exception `virttest.qemu_vm.VMMigrateProtoUnsupportedError` (*protocol, output*)

Bases: `virttest.virt_vm.VMMigrateProtoUnknownError`

When QEMU tells us it doesn’t know about a given migration protocol.

This usually happens when we’re testing older QEMU. It makes sense to skip the test in this situation.

```
virttest.gemu_vm.clean_tmp_files()
```

virttest.remote module

Functions and classes used for logging into guests and transferring files.

```
class virttest.remote.AexpectIOWrapperOut (obj)
    Bases: virttest.remote_commander.messenger.StdIOWrapperOutBase64
    Basic implementation of IOWrapper for stdout
    close()
    fileno()
    write(data)

exception virttest.remote.LoginAuthenticationError (msg, output)
    Bases: virttest.remote.LoginError

exception virttest.remote.LoginBadClientError (client)
    Bases: virttest.remote.LoginError

exception virttest.remote.LoginError (msg, output)
    Bases: exceptions.Exception

exception virttest.remote.LoginProcessTerminatedError (status, output)
    Bases: virttest.remote.LoginError

exception virttest.remote.LoginTimeoutError (output)
    Bases: virttest.remote.LoginError

class virttest.remote.RemoteFile (address, client, username, password, port, remote_path, limit='',
                                   log_filename=None, verbose=False, timeout=600)
    Bases: object
    Class to handle the operations of file on remote host or guest.
    add (line_list)
        Append lines in line_list into file on remote.
    remove (pattern_list)
        Remove the lines in remote file which matches a pattern in pattern_list.
    sub (pattern2repl_dict)
        Replace the string which match the pattern to the value contained in pattern2repl_dict.
    sub_else_add (pattern2repl_dict)
        Replace the string which match the pattern. If no match in the all lines, append the value to the end of file.
    truncate (length=0)
        Truncate the detail of remote file to assigned length Content before line 1 line 2 line 3 re-
        mote_file.truncate(length=1) Content after line 1

        Parameters length – how many lines you want to keep

class virttest.remote.RemoteRunner (client='ssh', host=None, port='22', username='root',
                                     password=None, prompt='[#$]\s*$', linesep='n',
                                     log_filename=None, timeout=240, internal_timeout=10,
                                     session=None)
    Bases: object
    Class to provide a utils.run-like method to execute command on remote host or guest. Provide a similar interface
    with utils.run on local.
```

run (*command*, *timeout=60*, *ignore_status=False*)

Method to provide a `utils.run`-like interface to execute command on remote host or guest.

Parameters

- **timeout** – Total time duration to wait for command return.
- **ignore_status** – If `ignore_status=True`, do not raise an exception, no matter what the exit code of the command is. Else, raise `CmdError` if exit code of command is not zero.

class `virttest.remote.Remote_Package` (*address*, *client*, *username*, *password*, *port*, *remote_path*)
Bases: `object`

pull_file (*local_path*, *timeout=600*)
Copy file from remote to local.

push_file (*local_path*, *timeout=600*)
Copy file from local to remote.

exception `virttest.remote.SCPAuthenticationError` (*msg*, *output*)
Bases: `virttest.remote.SCPError`

exception `virttest.remote.SCPAuthenticationTimeoutError` (*output*)
Bases: `virttest.remote.SCPAuthenticationError`

exception `virttest.remote.SCPError` (*msg*, *output*)
Bases: `exceptions.Exception`

exception `virttest.remote.SCPTransferFailedError` (*status*, *output*)
Bases: `virttest.remote.SCPError`

exception `virttest.remote.SCPTransferTimeoutError` (*output*)
Bases: `virttest.remote.SCPError`

`virttest.remote.copy_files_from` (*address*, *client*, *username*, *password*, *port*, *remote_path*, *local_path*, *limit=''*, *log_filename=None*, *verbose=False*, *timeout=600*, *interface=None*)
Copy files from a remote host (guest) using the selected client.

Parameters

- **client** – Type of transfer client
- **username** – Username (if required)
- **password** – Password (if required)
- **remote_path** – Path on the remote machine where we are copying from
- **local_path** – Path on the local machine where we are copying to
- **address** – Address of remote host(guest)
- **limit** – Speed limit of file transfer.
- **log_filename** – If specified, log all output to this file (SCP only)
- **verbose** – If True, log some stats using `logging.debug` (RSS only)
- **timeout** – The time duration (in seconds) to wait for the transfer to complete.

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

Raise Whatever `remote_scp()` raises

```
virttest.remote.copy_files_to(address, client, username, password, port, local_path, re-
                             mote_path, limit='', log_filename=None, verbose=False, time-
                             out=600, interface=None)
```

Copy files to a remote host (guest) using the selected client.

Parameters

- **client** – Type of transfer client
- **username** – Username (if required)
- **password** – Password (if required)
- **local_path** – Path on the local machine where we are copying from
- **remote_path** – Path on the remote machine where we are copying to
- **address** – Address of remote host(guest)
- **limit** – Speed limit of file transfer.
- **log_filename** – If specified, log all output to this file (SCP only)
- **verbose** – If True, log some stats using logging.debug (RSS only)
- **timeout** – The time duration (in seconds) to wait for the transfer to complete.

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

Raise Whatever remote_scp() raises

```
virttest.remote.handle_prompts(session, username, password, prompt, timeout=10, de-
                               bug=False)
```

Connect to a remote host (guest) using SSH or Telnet or else.

Wait for questions and provide answers. If timeout expires while waiting for output from the child (e.g. a password prompt or a shell prompt) – fail.

Parameters

- **session** – An Expect or ShellSession instance to operate on
- **username** – The username to send in reply to a login prompt
- **password** – The password to send in reply to a password prompt
- **prompt** – The shell prompt that indicates a successful login
- **timeout** – The maximal time duration (in seconds) to wait for each step of the login procedure (i.e. the “Are you sure” prompt, the password prompt, the shell prompt, etc)

Raises

- **LoginTimeoutError** – If timeout expires
- **LoginAuthenticationError** – If authentication fails
- **LoginProcessTerminatedError** – If the client terminates during login
- **LoginError** – If some other error occurs

Returns If connect succeed return the output text to script for further debug.

```
virttest.remote.nc_copy_between_remotes(src, dst, s_port, s_passwd, d_passwd, s_name,
                                         d_name, s_path, d_path, c_type='ssh',
                                         c_prompt='\n', d_port='8888', d_protocol='udp',
                                         timeout=10, check_sum=True)
```

Copy files from guest to guest using netcat.

This method only supports linux guest OS.

Parameters

- **src/dst** – Hostname or IP address of src and dst
- **s_name/d_name** – Username (if required)
- **s_passwd/d_passwd** – Password (if required)
- **s_path/d_path** – Path on the remote machine where we are copying
- **c_type** – Login method to remote host(guest).
- **c_prompt** – command line prompt of remote host(guest)
- **d_port** – the port data transfer
- **d_protocol** – nc protocol use (tcp or udp)
- **timeout** – If a connection and stdin are idle for more than timeout seconds, then the connection is silently closed.

Returns True on success and False on failure.

```
virttest.remote.remote_commander(client, host, port, username, password, prompt, linesep='\n',  
                                log_filename=None, timeout=10, path=None)
```

Log into a remote host (guest) using SSH/Telnet/Netcat.

Parameters

- **client** – The client to use ('ssh', 'telnet' or 'nc')
- **host** – Hostname or IP address
- **port** – Port to connect to
- **username** – Username (if required)
- **password** – Password (if required)
- **prompt** – Shell prompt (regular expression)
- **linesep** – The line separator to use when sending lines (e.g. 'n' or 'rn')
- **log_filename** – If specified, log all output to this file
- **timeout** – The maximal time duration (in seconds) to wait for each step of the login procedure (i.e. the “Are you sure” prompt or the password prompt)
- **path** – The path to place where remote_runner.py is placed.

Raises **LoginBadClientError** – If an unknown client is requested

Raise Whatever handle_prompts() raises

Returns A ShellSession object.

```
virttest.remote.remote_login(client, host, port, username, password, prompt, linesep='\n',  
                             log_filename=None, timeout=10, interface=None, sta-  
                             tus_test_command='echo $?')
```

Log into a remote host (guest) using SSH/Telnet/Netcat.

Parameters

- **client** – The client to use ('ssh', 'telnet' or 'nc')
- **host** – Hostname or IP address
- **port** – Port to connect to

- **username** – Username (if required)
- **password** – Password (if required)
- **prompt** – Shell prompt (regular expression)
- **linesep** – The line separator to use when sending lines (e.g. 'n' or 'rn')
- **log_filename** – If specified, log all output to this file
- **timeout** – The maximal time duration (in seconds) to wait for each step of the login procedure (i.e. the “Are you sure” prompt or the password prompt)
- **status_test_command** – Command to be used for getting the last exit status of commands run inside the shell (used by `cmd_status_output()` and friends).

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

Raises

- **LoginError** – If using ipv6 linklocal but not assign a interface that the neighbour attache
- **LoginBadClientError** – If an unknown client is requested

Raise Whatever `handle_prompts()` raises

Returns A `ShellSession` object.

```
virttest.remote.remote_scp(command, password_list, log_filename=None, transfer_timeout=600,
                           login_timeout=20)
```

Transfer files using SCP, given a command line.

Parameters

- **command** – The command to execute (e.g. “`scp -r foobar root@localhost:/tmp/`”).
- **password_list** – Password list to send in reply to a password prompt.
- **log_filename** – If specified, log all output to this file
- **transfer_timeout** – The time duration (in seconds) to wait for the transfer to complete.
- **login_timeout** – The maximal time duration (in seconds) to wait for each step of the login procedure (i.e. the “Are you sure” prompt or the password prompt)

Raise Whatever `_remote_scp()` raises

```
virttest.remote.scp_between_remotes(src, dst, port, s_passwd, d_passwd, s_name, d_name,
                                     s_path, d_path, limit='', log_filename=None, time-
                                     out=600, src_inter=None, dst_inter=None)
```

Copy files from a remote host (guest) to another remote host (guest).

Parameters

- **src/dst** – Hostname or IP address of src and dst
- **s_name/d_name** – Username (if required)
- **s_passwd/d_passwd** – Password (if required)
- **s_path/d_path** – Path on the remote machine where we are copying from/to
- **limit** – Speed limit of file transfer.
- **log_filename** – If specified, log all output to this file
- **timeout** – The time duration (in seconds) to wait for the transfer to complete.

Src_inter The interface on local that the src neighbour attache

Dst_inter The interface on the src that the dst neighbour attache

Returns True on success and False on failure.

`virttest.remote.scp_from_remote` (*host, port, username, password, remote_path, local_path, limit='', log_filename=None, timeout=600, interface=None*)

Copy files from a remote host (guest).

Parameters

- **host** – Hostname or IP address
- **username** – Username (if required)
- **password** – Password (if required)
- **local_path** – Path on the local machine where we are copying from
- **remote_path** – Path on the remote machine where we are copying to
- **limit** – Speed limit of file transfer.
- **log_filename** – If specified, log all output to this file
- **timeout** – The time duration (in seconds) to wait for the transfer to complete.

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

Raise Whatever `remote_scp()` raises

`virttest.remote.scp_to_remote` (*host, port, username, password, local_path, remote_path, limit='', log_filename=None, timeout=600, interface=None*)

Copy files to a remote host (guest) through scp.

Parameters

- **host** – Hostname or IP address
- **username** – Username (if required)
- **password** – Password (if required)
- **local_path** – Path on the local machine where we are copying from
- **remote_path** – Path on the remote machine where we are copying to
- **limit** – Speed limit of file transfer.
- **log_filename** – If specified, log all output to this file
- **timeout** – The time duration (in seconds) to wait for the transfer to complete.

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

Raise Whatever `remote_scp()` raises

`virttest.remote.udp_copy_between_remotes` (*src, dst, s_port, s_passwd, d_passwd, s_name, d_name, s_path, d_path, c_type='ssh', c_prompt='\n', d_port='9000', timeout=600*)

Copy files from guest to guest using udp.

Parameters

- **src/dst** – Hostname or IP address of src and dst
- **s_name/d_name** – Username (if required)
- **s_passwd/d_passwd** – Password (if required)
- **s_path/d_path** – Path on the remote machine where we are copying

- **c_type** – Login method to remote host(guest).
- **c_prompt** – command line prompt of remote host(guest)
- **d_port** – the port data transfer
- **timeout** – data transfer timeout

```
virttest.remote.wait_for_login(client, host, port, username, password, prompt, linesep='\n',
                               log_filename=None, timeout=240, internal_timeout=10, inter-
                               face=None)
```

Make multiple attempts to log into a guest until one succeeds or timeouts.

Parameters

- **timeout** – Total time duration to wait for a successful login
- **internal_timeout** – The maximum time duration (in seconds) to wait for each step of the login procedure (e.g. the “Are you sure” prompt or the password prompt)

Interface The interface the neighbours attach to (only use when using ipv6 linklocal address.)

See remote_login()

Raise Whatever remote_login() raises

Returns A ShellSession object.

virttest.remote_build module

exception virttest.remote_build.**BuildError** (error_info)

Bases: `exceptions.Exception`

```
class virttest.remote_build.Builder(params, address, source, shell_client=None,
                                     shell_port=None, file_transfer_client=None,
                                     file_transfer_port=None, username=None, pass-
                                     word=None, make_flags='', build_dir=None,
                                     build_dir_prefix=None, shell_linesep=None,
                                     shell_prompt=None)
```

Bases: `object`

build()

Synchronize all files and execute ‘make’ on the remote system if needed. :returns: The path to the build directory on the remote machine

make()

Execute make on the remote system

sync_directories()

Synchronize the directories between the local and remote machines :returns: True if any files needed to be copied; False otherwise. Does not support symlinks.

virttest.remote_unittest module

class virttest.remote_unittest.**RemoteFileTest** (methodName='runTest')

Bases: `unittest.case.TestCase`

default_data = ['RemoteFile Test.\n', 'Pattern Line.']

testAdd()

testRemove()

```
testSEEA()
testSub()
test_file_path = '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/tmp/remote_file'
tmp_dir = '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/tmp'
```

virttest.rss_client module

Client for file transfer services offered by RSS (Remote Shell Server).

author Michael Goldish (mgoldish@redhat.com)

copyright 2008-2010 Red Hat Inc.

class `virttest.rss_client.FileDownloadClient` (*address, port, log_func=None, timeout=20*)

Bases: `virttest.rss_client.FileTransferClient`

Connect to a RSS (remote shell server) and download files or directory trees.

download (*src_pattern, dst_path, timeout=600*)

Receive files or directory trees from the server. The semantics of `src_pattern` and `dst_path` are similar to those of `scp`.

For example, the following are OK:

```
src_pattern='C:\foo.txt', dst_path='/tmp'
    (downloads a single file)
src_pattern='C:\Windows', dst_path='/tmp'
    (downloads a directory tree recursively)
src_pattern='C:\Windows\*', dst_path='/tmp'
    (downloads all files and directory trees under C:\Windows)
```

The following is not OK:

```
src_pattern='C:\Windows', dst_path='/tmp/*'
    (wildcards are only allowed in src_pattern)
```

Parameters

- **src_pattern** – A path or wildcard pattern specifying the files or directories, in the server's filesystem, that will be sent to the client
- **dst_path** – A path in the local filesystem where the files will be saved
- **timeout** – Time duration in seconds to wait for the transfer to complete

Raises

- **FileTransferTimeoutError** – Raised if timeout expires
- **FileTransferServerError** – Raised if something goes wrong and the server sends an informative error message to the client

Note Other exceptions can be raised.

class `virttest.rss_client.FileTransferClient` (*address, port, log_func=None, timeout=20*)

Bases: `object`

Connect to a RSS (remote shell server) and transfer files.

close()

Close the connection.

exception `virttest.rss_client.FileTransferConnectError` (*msg, e=None, filename=None*)

Bases: `virttest.rss_client.FileTransferError`

exception `virttest.rss_client.FileTransferError` (*msg, e=None, filename=None*)

Bases: `exceptions.Exception`

exception `virttest.rss_client.FileTransferNotFoundError` (*msg, e=None, filename=None*)

Bases: `virttest.rss_client.FileTransferError`

exception `virttest.rss_client.FileTransferProtocolError` (*msg, e=None, filename=None*)

Bases: `virttest.rss_client.FileTransferError`

exception `virttest.rss_client.FileTransferServerError` (*errmsg*)

Bases: `virttest.rss_client.FileTransferError`

exception `virttest.rss_client.FileTransferSocketError` (*msg, e=None, filename=None*)

Bases: `virttest.rss_client.FileTransferError`

exception `virttest.rss_client.FileTransferTimeoutError` (*msg, e=None, filename=None*)

Bases: `virttest.rss_client.FileTransferError`

class `virttest.rss_client.FileUploadClient` (*address, port, log_func=None, timeout=20*)

Bases: `virttest.rss_client.FileTransferClient`

Connect to a RSS (remote shell server) and upload files or directory trees.

upload (*src_pattern, dst_path, timeout=600*)

Send files or directory trees to the server.

The semantics of *src_pattern* and *dst_path* are similar to those of `scp`. For example, the following are OK:

```
src_pattern='/tmp/foo.txt', dst_path='C:\'
    (uploads a single file)
src_pattern='/usr/', dst_path='C:\Windows\'
    (uploads a directory tree recursively)
src_pattern='/usr/*', dst_path='C:\Windows\'
    (uploads all files and directory trees under /usr/)
```

The following is not OK:

```
src_pattern='/tmp/foo.txt', dst_path='C:\Windows\*'
    (wildcards are only allowed in src_pattern)
```

Parameters

- **src_pattern** – A path or wildcard pattern specifying the files or directories to send to the server
- **dst_path** – A path in the server's filesystem where the files will be saved
- **timeout** – Time duration in seconds to wait for the transfer to complete

Raises

- **FileTransferTimeoutError** – Raised if timeout expires
- **FileTransferServerError** – Raised if something goes wrong and the server sends an informative error message to the client

Note Other exceptions can be raised.

```
virttest.rss_client.download(address, port, src_pattern, dst_path, log_func=None, timeout=60,  
                             connect_timeout=20)
```

Connect to server and upload files.

:see:: FileDownloadClient

```
virttest.rss_client.main()
```

```
virttest.rss_client.upload(address, port, src_pattern, dst_path, log_func=None, timeout=60, con-  
                           nect_timeout=20)
```

Connect to server and upload files.

:see:: FileUploadClient

virttest.scheduler module

class virttest.scheduler.**scheduler**(tests, num_workers, total_cpus, total_mem, bindir)
A scheduler that manages several parallel test execution pipelines on a single host.

scheduler()

The scheduler function.

Sends commands to workers, telling them to run tests, clean up or terminate execution.

worker(index, run_test_func)

The worker function.

Waits for commands from the scheduler and processes them.

Parameters

- **index** – The index of this worker (in the range 0..num_workers-1).
- **run_test_func** – A function to be called to run a test (e.g. job.run_test).

virttest.service_unittest module

class virttest.service_unittest.**ConstantsTest**(methodName='runTest')

Bases: unittest.case.TestCase

test_ModuleLoad()

class virttest.service_unittest.**ResultParserTest**(methodName='runTest')

Bases: unittest.case.TestCase

test_systemd_result_parser()

test_sysvinit_result_parser()

class virttest.service_unittest.**SysVInitGeneratorTest**(methodName='runTest')

Bases: unittest.case.TestCase

setUp()

test_all_command()

test_set_target()

class virttest.service_unittest.**SystemdGeneratorTest**(methodName='runTest')

Bases: unittest.case.TestCase

setUp()

test_all_command()

```
test_list()
test_set_target()
class virttest.service_unittest.TestServiceManager (init_name, run_mock)
    Bases: object
    get_service_manager()
class virttest.service_unittest.TestSysVInitServiceManager (methodName='runTest')
    Bases: unittest.case.TestCase
    setUp()
    test_enable()
    test_list()
    test_runlevels()
    test_unknown_runlevel()
class virttest.service_unittest.TestSystemdServiceManager (methodName='runTest')
    Bases: unittest.case.TestCase
    setUp()
    test_list()
    test_start()
```

virttest.standalone_test module

```
class virttest.standalone_test.Bcolors
    Bases: object
    Very simple class with color support.
    disable()
class virttest.standalone_test.Test (params, options)
    Bases: object
    Minimal test class used to run a virt test.
    env_version = 1
    run_once()
    set_debugdir (debugdir)
    start_file_logging()
    stop_file_logging()
    verify_background_errors()
        Verify if there are any errors that happened on background threads.
        Raises Exception – Any exception stored on the background_errors queue.
    write_test_keyval (d)
virttest.standalone_test.bootstrap_tests (options)
    Bootstrap process (download the appropriate JeOS file to data dir).
    This function will check whether the JeOS is in the right location of the data dir, if not, it will download it non
    interactively.
```

Parameters **options** – OptParse object with program command line options.

`virttest.standalone_test.cleanup_env(parser, options)`
Clean up virt-test temporary files.

Parameters

- **parser** – Cartesian parser with run parameters.
- **options** – Test runner options object.

`virttest.standalone_test.configure_console_logging(loglevel=10)`
Simple helper for adding a file logger to the root logger.

`virttest.standalone_test.configure_file_logging(logfile, loglevel=10)`
Simple helper for adding a file logger to the root logger.

`virttest.standalone_test.create_config_files(options)`
Check if the appropriate configuration files are present.

If the files are not present, create them.

Parameters **options** – OptParser object with options.

`virttest.standalone_test.find_default_qemu_paths(options_qemu=None, options_dst_qemu=None)` *op-*

`virttest.standalone_test.get_cartesian_parser_details(cartesian_parser)`
Print detailed information about filters applied to the cartesian cfg.

Parameters **cartesian_parser** – Cartesian parser object.

`virttest.standalone_test.get_guest_name_list(options)`

`virttest.standalone_test.get_guest_name_parser(options)`

`virttest.standalone_test.get_paginator()`

`virttest.standalone_test.print_error(t_elapsed, open_fd=False)`
Print ERROR to stdout with ERROR (red) color.

`virttest.standalone_test.print_fail(t_elapsed, open_fd=False)`
Print FAIL to stdout with FAIL (red) color.

`virttest.standalone_test.print_guest_list(options)`
Helper function to pretty print the guest list.

This function uses a paginator, if possible (inspired on git).

Parameters

- **options** – OptParse object with cmdline options.
- **cartesian_parser** – Cartesian parser object with test options.

`virttest.standalone_test.print_header(sr)`
Print a string to stdout with HEADER (blue) color.

`virttest.standalone_test.print_pass(t_elapsed, open_fd=False)`
Print PASS to stdout with PASS (green) color.

`virttest.standalone_test.print_skip(open_fd=False)`
Print SKIP to stdout with SKIP (yellow) color.

`virttest.standalone_test.print_stdout(sr, end=True)`

`virttest.standalone_test.print_test_list(options, cartesian_parser)`

Helper function to pretty print the test list.

This function uses a paginator, if possible (inspired on git).

Parameters

- **options** – OptParse object with cmdline options.
- **cartesian_parser** – Cartesian parser object with test options.

`virttest.standalone_test.print_warn(t_elapsed, open_fd=False)`

Print WARN to stdout with WARN (yellow) color.

`virttest.standalone_test.reset_logging()`

Remove all the handlers and unset the log level on the root logger.

`virttest.standalone_test.run_tests(parser, options)`

Runs the sequence of KVM tests based on the list of dictionaries generated by the configuration system, handling dependencies.

Parameters

- **parser** – Config parser object.
- **options** – Test runner options object.

Returns True, if all tests ran passed, False if any of them failed.

virttest.step_editor module

virttest.storage module

Classes and functions to handle storage devices.

This exports:

- two functions for get image/blkdebug filename
- class for image operates and basic parameters

class `virttest.storage.Iscsidev(params, root_dir, tag)`

Bases: `virttest.storage.Rawdev`

Class for handle iscsi devices for VM

class `virttest.storage.LVMdev(params, root_dir, tag)`

Bases: `virttest.storage.Rawdev`

Class for handle LVM devices for VM

exception `virttest.storage.OptionMissing(option)`

Bases: `exceptions.Exception`

Option not found in the oobject

class `virttest.storage.QemuImg(params, root_dir, tag)`

Bases: `object`

A basic class for handling operations of disk/block images.

backup_image `(params, root_dir, action, good=True, skip_existing=False)`

Backup or restore a disk image, depending on the action chosen.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.
- **action** – Whether we want to backup or restore the image.
- **good** – If we are backing up a good image(we want to restore it) or a bad image (we are saving a bad image for posterior analysis).

Note params should contain: **image_name** – the name of the image file, without extension **image_format** – the format of the image (qcow2, raw etc)

check_option (*option*)

Check if object has the option required.

Parameters **option** – option should be checked

static clone_image (*params, vm_name, image_name, root_dir*)

Clone master image to vm specific file.

Parameters

- **params** – Dictionary containing the test parameters.
- **vm_name** – Vm name.
- **image_name** – Master image name.
- **root_dir** – Base directory for relative filenames.

is_remote_image ()

Check if image is from a remote server or not

static rm_cloned_image (*params, vm_name, image_name, root_dir*)

Remove vm specific file.

Parameters

- **params** – Dictionary containing the test parameters.
- **vm_name** – Vm name.
- **image_name** – Master image name.
- **root_dir** – Base directory for relative filenames.

class virttest.storage.**Rawdev** (*params, root_dir, tag*)

Bases: `object`

Base class for raw storage devices such as iscsi and local disks

virttest.storage.**file_exists** (*params, filename_path*)

Check if image_filename exists.

Parameters

- **params** – Dictionary containing the test parameters.
- **filename_path** (*str*) – path to file
- **root_dir** (*str*) – Base directory for relative filenames.

Returns True if image file exists else False

virttest.storage.**file_remove** (*params, filename_path*)

Remove the image :param params: Dictionary containing the test parameters. :param filename_path: path to file

`virttest.storage.get_image_blkdebug_filename(params, root_dir)`

Generate an blkdebug file path from params and root_dir.

blkdebug files allow error injection in the block subsystem.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.

Note params should contain: blkdebug – the name of the debug file.

`virttest.storage.get_image_filename(params, root_dir)`

Generate an image path from params and root_dir.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.
- **image_name** – Force name of image.
- **image_format** – Format for image.

Note params should contain: image_name – the name of the image file, without extension image_format – the format of the image (qcow2, raw etc)

Raises **VMDeviceError** – When no matching disk found (in indirect method).

`virttest.storage.get_image_filename_filesystem(params, root_dir)`

Generate an image path from params and root_dir.

Parameters

- **params** – Dictionary containing the test parameters.
- **root_dir** – Base directory for relative filenames.

Note params should contain: image_name – the name of the image file, without extension image_format – the format of the image (qcow2, raw etc)

Raises **VMDeviceError** – When no matching disk found (in indirect method).

`virttest.storage.postprocess_images(bindir, params)`

`virttest.storage.preprocess_image_backend(bindir, params, env)`

`virttest.storage.preprocess_images(bindir, params, env)`

virttest.syslog_server module

`class virttest.syslog_server.RequestHandler(request, client_address, server)`

Bases: `SocketServer.BaseRequestHandler`

A request handler that relays all received messages as DEBUG

FACILITY_NAMES = {0: 'kern', 1: 'user', 2: 'mail', 3: 'daemon', 4: 'security', 5: 'syslog', 6: 'lpr', 7: 'news', 8: 'uucp', 9

LOG_ALERT = 1

LOG_AUTH = 4

LOG_AUTHPRIV = 10

LOG_CRIT = 2

```
LOG_CRON = 9
LOG_DAEMON = 3
LOG_DEBUG = 7
LOG_EMERG = 0
LOG_ERR = 3
LOG_FTP = 11
LOG_INFO = 6
LOG_KERN = 0
LOG_LOCAL0 = 16
LOG_LOCAL1 = 17
LOG_LOCAL2 = 18
LOG_LOCAL3 = 19
LOG_LOCAL4 = 20
LOG_LOCAL5 = 21
LOG_LOCAL6 = 22
LOG_LOCAL7 = 23
LOG_LPR = 6
LOG_MAIL = 2
LOG_NEWS = 7
LOG_NOTICE = 5
LOG_SYSLOG = 5
LOG_USER = 1
LOG_UUCP = 8
LOG_WARNING = 4
PRIORITY_NAMES = {0: 'emerg', 1: 'alert', 2: 'critical', 3: 'err', 4: 'warning', 5: 'notice', 6: 'info', 7: 'debug'}
RECORD_RE = <_sre.SRE_Pattern object>
decodeFacilityPriority(priority)
    Decode both the facility and priority embedded in a syslog message
    Parameters priority (integer) – an integer with facility and priority encoded
    Returns a tuple with two strings
log(data, message_format=None)
    Logs the received message as a DEBUG message
class virttest.syslog_server.RequestHandlerTcp(request, client_address, server)
    Bases: virttest.syslog_server.RequestHandler
    handle()
        Handles a single request
class virttest.syslog_server.RequestHandlerUdp(request, client_address, server)
    Bases: virttest.syslog_server.RequestHandler
```

```

    handle ()
        Handles a single request

class virttest.syslog_server.SysLogServerTcp (address)
    Bases: SocketServer.TCPServer

class virttest.syslog_server.SysLogServerUdp (address)
    Bases: SocketServer.UDPServer

virttest.syslog_server.get_default_format ()
    Returns the current default message format

virttest.syslog_server.set_default_format (message_format)
    Changes the default message format

        Parameters message_format (string) – a message format string with 3 placeholders: facility,
            priority and message.

virttest.syslog_server.syslog_server (address='', port=514, tcp=True, terminate_callable=None)

```

virttest.test_setup module

Library to perform pre/post test setup for virt test.

```

class virttest.test_setup.EGDConfig (params, env)
    Bases: object

    Setup egd.pl server on localhost, support startup with socket unix or tcp.

    cleanup ()

    get_pid (socket)
        Check egd.pl start at socket on localhost.

    install ()
        Install egd.pl from source code

    setup ()

    startup (socket)
        Start egd.pl server with tcp or unix socket.

exception virttest.test_setup.EGDConfigError
    Bases: exceptions.Exception

    Raise when setup local egd.pl server failed.

exception virttest.test_setup.HPNotSupportedError
    Bases: exceptions.Exception

    Thrown when host does not support hugepages.

class virttest.test_setup.HugePageConfig (params)
    Bases: object

    cleanup (*args, **kwargs)

    get_hugepage_size ()
        Get the current system setting for huge memory page size.

    get_multi_supported_hugepage_size ()
        As '/proc/meminfo' only show default huge page size, this function is for get huge page size of multiple
        huge page pools.

```

For each huge page size supported by the running kernel, a subdirectory will exist, of the form:

hugepages-\${size}kB

under /sys/kernel/mm/hugepages, get the support size and return a list.

Returns supported size list in kB unit

get_node_num_huge_pages (*node*, *pagesize*)

Get number of pages of certain page size under given numa node.

Parameters

- **node** – string or int, node number
- **pagesize** – string or int, page size in kB

Returns int, node huge pages number of given page size

get_target_hugepages ()

Calculate the target number of hugepages for testing purposes.

mount_hugepage_fs (**args*, ***kwargs*)

Verify if there's a hugetlbfs mount set. If there's none, will set up a hugetlbfs mount using the class attribute that defines the mount point.

set_hugepages (**args*, ***kwargs*)

Sets the hugepage limit to the target hugepage value calculated.

set_node_num_huge_pages (*num*, *node*, *pagesize*)

Set number of pages of certain page size under given numa node.

Parameters

- **num** – string or int, number of pages
- **node** – string or int, node number
- **pagesize** – string or int, page size in kB

setup ()

class virttest.test_setup.**KSMConfig** (*params*, *env*)

Bases: `object`

cleanup (*env*)

setup (*env*)

class virttest.test_setup.**LibvirtPolkitConfig** (*params*)

Bases: `object`

Enable polkit access driver for libvirtd and set polkit rules.

For setting JavaScript polkit rule, using template of rule to satisfy libvirt ACL API testing need, just replace keys in template.

Create a non-privileged user 'testacl' for test if given 'unprivileged_user' contains 'EXAMPLE', and delete the user at cleanup.

Multiple rules could be add into one config file while action_id string is offered space separated.

e.g. action_id = "org.libvirt.api.domain.start org.libvirt.api.domain.write"

then 2 actions "org.libvirt.api.domain.start" and "org.libvirt.api.domain.write" specified, which could be used to generate 2 rules in one config file.

cleanup()

Cleanup polkit config

file_replace_append(*fpath, pat, repl*)

Replace pattern in file with replacement str if pattern found in file, else append the replacement str to file.

Parameters

- **fpath** – string, the file path
- **pat** – string, the pattern string
- **repl** – string, the string to replace

setup()

Enable polkit libvirt access driver and setup polkit ACL rules.

```
class virttest.test_setup.PciAssignable (driver=None, driver_option=None,
                                         host_set_flag=None, kvm_params=None,
                                         vf_filter_re=None, pf_filter_re=None, de-
                                         vice_driver=None, nic_name_re=None)
```

Bases: `object`

Request PCI assignable devices on host. It will check whether to request PF (physical Functions) or VF (Virtual Functions).

add_device(*device_type='vf', name=None, mac=None*)

Add device type and name to class.

Parameters

- **device_type**(*string*) – vf/pf device is added.
- **name**(*string*) – Physical device interface name. eth1 or others
- **mac**(*string*) – set mac address for vf.

check_vfs_count()

Check VFs count number according to the parameter driver_options.

get_devs(*devices=None*)

Get devices' PCI IDs according to parameters set in self.devices.

Parameters **devices**(*List of dict*) – List of device dict that contain PF VF information.

Returns List of all available devices' PCI IDs

Return type List of string

get_pf_devs(*devices=None*)

Get PFs PCI IDs requested by self.devices. It will try to get PF by device name. It will still return it, if device name you set already occupied. Please set unoccupied device name. If not sure, please just do not set device name. It will return unused PF list.

Parameters **devices**(*List of dict*) – List of device dict that contain PF VF information.

Returns List with all PCI IDs for the physical hardware requested

Return type List of string

get_pf_vf_info()

Get pf and vf related information in this host that match `self.pf_filter_re`.

for every pf it will create following information:

pf_id: The id of the pf device.

occupied: Whether the pf device assigned or not

vf_ids: Id list of related vf in this pf.

ethname: eth device name in host for this pf.

Returns return a list contains pf vf information.

Return type list of dict

get_same_group_devs (*pci_id*)

Get the device that in same iommu group.

Parameters **pci_id** (*string*) – Device's pci_id

Returns Return the device's pci id that in same group with pci_id.

Return type List of string.

get_vf_devs (*devices=None*)

Get all unused VFs PCI IDs.

Parameters **devices** (*List of dict*) – List of device dict that contain PF VF information.

Returns List of all available PCI IDs for Virtual Functions.

Return type List of string

get_vf_num_by_id (*vf_id*)

Return corresponding pf eth name and vf num according to vf id.

Parameters **vf_id** (*string*) – vf id to check.

Returns PF device name and vf num.

Return type *string*

get_vf_status (*vf_id*)

Check whether one vf is assigned to VM.

Parameters **vf_id** (*string*) – vf id to check.

Returns Return True if vf has already assigned to VM. Else return false.

Return type *bool*

get_vfs_count ()

Get VFs count number according to lspci.

is_binded_to_stub (*full_id*)

Verify whether the device with full_id is already binded to driver.

Parameters **full_id** (*String*) – Full ID for the given PCI device

release_devs (**args, **kwargs*)

Release all PCI devices currently assigned to VMs back to the virtualization host.

request_devs (*devices=None*)

Implement setup process: unbind the PCI device and then bind it to the device driver.

Parameters **devices** (*List of dict*) – List of device dict

Returns List of successfully requested devices' PCI IDs.

Return type List of string

sr_iov_cleanup()

Clean up the sriov setup

Check if the PCI hardware device drive is loaded with the appropriate, parameters (none of VFs), and if it's not, perform cleanup.

Returns True, if the setup was completed successfully, False otherwise.

Return type bool

sr_iov_setup(*args, **kwargs)

Ensure the PCI device is working in sr_iov mode.

Check if the PCI hardware device drive is loaded with the appropriate, parameters (number of VFs), and if it's not, perform setup.

Returns True, if the setup was completed successfully, False otherwise.

Return type bool

exception virttest.test_setup.PolkitConfigCleanupError

Bases: *virttest.test_setup.PolkitConfigError*

Thrown when polkit config cleanup is not behaving as expected.

exception virttest.test_setup.PolkitConfigError

Bases: *exceptions.Exception*

Base exception for Polkit Config setup.

exception virttest.test_setup.PolkitRulesSetupError

Bases: *virttest.test_setup.PolkitConfigError*

Thrown when setup polkit rules is not behaving as expected.

exception virttest.test_setup.PolkitWriteLibvirtdConfigError

Bases: *virttest.test_setup.PolkitConfigError*

Thrown when setup libvirtd config file is not behaving as expected.

class virttest.test_setup.PrivateBridgeConfig(*params=None*)

Bases: *object*

cleanup()

setup()

exception virttest.test_setup.PrivateBridgeError(*brname*)

Bases: *exceptions.Exception*

class virttest.test_setup.PrivateOvsBridgeConfig(*params=None*)

Bases: *virttest.test_setup.PrivateBridgeConfig*

exception virttest.test_setup.THPError

Bases: *exceptions.Exception*

Base exception for Transparent Hugepage setup.

exception virttest.test_setup.THPKhugepagedError

Bases: *virttest.test_setup.THPError*

Thrown when khugepaged is not behaving as expected.

exception virttest.test_setup.THPNotSupportedError

Bases: *virttest.test_setup.THPError*

Thrown when host does not support transparent hugepages.

exception `virttest.test_setup.THPWriteConfigError`

Bases: `virttest.test_setup.THPError`

Thrown when host does not support transparent hugepages.

class `virttest.test_setup.TransparentHugePageConfig` (*test, params*)

Bases: `object`

cleanup ()

: Restore the host's original configuration after test

khugepaged_test ()

Start, stop and frequency change test for khugepaged.

set_env ()

Applies test configuration on the host.

setup ()

Configure host for testing. Also, check that khugepaged is working as expected.

value_listed (*value*)

Get a parameters list from a string

virttest.utils_cgroup_unittest module

class `virttest.utils_cgroup_unittest.CgroupTest` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

test_get_cgroup_mountpoint ()

virttest.utils_config module

exception `virttest.utils_config.ConfigError` (*msg*)

Bases: `exceptions.Exception`

exception `virttest.utils_config.ConfigNoOptionError` (*option, path*)

Bases: `virttest.utils_config.ConfigError`

class `virttest.utils_config.LibvirtConfigCommon` (*path=''*)

Bases: `virttest.utils_config.SectionlessConfig`

A abstract class to manipulate options of a libvirt related configure files in a property's way.

Variables “__option_types__” and “conf_path” must be setup in the inherited classes before use.

“__option_types__” is a dict contains every possible option as keys and their type (“boolean”, “int”, “string”, “float” or “list”) as values.

Basic usage: 1) Create a config file object: `>>> # LibvirtdConfig is a subclass of LibvirtConfigCommon. >>> config = LibvirtdConfig()`

2) Set or update an option: `>>> config.listen_tcp = True >>> config.listen_tcp = 1 >>> config.listen_tcp = "1"`
All three have the same effect.

```
>>> # If the setting value don't meet the specified type.
>>> config.listen_tcp = "invalid"
>>> # It'll thown an warning message and set a raw string instead.
```

```
>>> # Use set_* methods when need to customize the result.
>>> config.set_raw("'1'")
```


- 3) Get an option: `>>> is_listening = config.listen_tcp >>> print is_listening True`
- 4) Delete an option from the config file: `>>> del config.listen_tcp`
- 5) Make the changes take effect in libvirt by restart libvirt daemon. `>>> from virttest import utils_libvirtd >>> utils_libvirtd.Libvirtd().restart()`
- 6) Restore the content of the config file. `>>> config.restore()`

conf_path = ''

exception `virttest.utils_config.LibvirtConfigUnknownKeyError (key)`

Bases: `virttest.utils_config.ConfigError`

exception `virttest.utils_config.LibvirtConfigUnknownKeyTypeError (key, key_type)`

Bases: `virttest.utils_config.ConfigError`

class `virttest.utils_config.LibvirtGuestsConfig (path='')`

Bases: `virttest.utils_config.LibvirtConfigCommon`

Class for sysconfig libvirt-guests config file.

conf_path = '/etc/sysconfig/libvirt-guests'

class `virttest.utils_config.LibvirtQemuConfig (path='')`

Bases: `virttest.utils_config.LibvirtConfigCommon`

Class for libvirt qemu config file.

conf_path = '/etc/libvirt/qemu.conf'

class `virttest.utils_config.LibvirtdConfig (path='')`

Bases: `virttest.utils_config.LibvirtConfigCommon`

Class for libvirt daemon config file.

conf_path = '/etc/libvirt/libvirtd.conf'

class `virttest.utils_config.LibvirtdSysConfig (path='')`

Bases: `virttest.utils_config.LibvirtConfigCommon`

Class for sysconfig libvirtd config file.

conf_path = '/etc/sysconfig/libvirtd'

class `virttest.utils_config.SectionlessConfig (path)`

Bases: `object`

This is a wrapper class for python's internal library ConfigParser except allows manipulating sectionless configuration file with a dict-like way.

Example config file test.conf:

```
># This is a comment line. >a = 1 >b = [hi, there] >c = hello >d = "hi, there" >e = [hi, > there]
```

Example script using `try...finally...` statement:

```
>>> from virttest import utils_config
>>> config = utils_config.SectionlessConfig('test.conf')
>>> try:
...     print len(config)
...     print config
...     print config['a']
...     del config['a']
...     config['f'] = 'test'
...     print config
```

```
... finally:
...     config.restore()
```

Example script using *with* statement:

```
>>> from virttest import utils_config
>>> with utils_config.SectionlessConfig('test.conf') as config:
...     print len(config)
...     print config
...     print config['a']
...     del config['a']
...     config['f'] = 'test'
...     print config
```

```
get_boolean (option)
get_float (option)
get_int (option)
get_list (option)
get_raw (option)
get_string (option)
restore ()
set_boolean (option, value)
set_float (option, value)
set_int (option, value)
set_list (option, value)
set_raw (option, value)
set_string (option, value)
```

virttest.utils_config_unittest module

```
class virttest.utils_config_unittest.LibvirtConfigCommonTest (methodName='runTest')
    Bases: unittest.case.TestCase

    class NoTypesConfig (path='')
        Bases: virttest.utils_config.LibvirtConfigCommon
        conf_path = '/tmp/config_unittest.conf'

    class LibvirtConfigCommonTest.UndefinedTypeConfig (path='')
        Bases: virttest.utils_config.LibvirtConfigCommon
        conf_path = '/tmp/config_unittest.conf'

    class LibvirtConfigCommonTest.UnimplementedConfig (path='')
        Bases: virttest.utils_config.LibvirtConfigCommon

    LibvirtConfigCommonTest.test_no_path()
    LibvirtConfigCommonTest.test_undefined_type()
    LibvirtConfigCommonTest.test_unimplemented()
```

```
class virttest.utils_config_unittest.LibvirtConfigTest (methodName='runTest')
    Bases: unittest.case.TestCase

    test_accessers ()

class virttest.utils_config_unittest.SectionlessConfigTest (methodName='runTest')
    Bases: unittest.case.TestCase

    test_accessers ()

    test_restore ()

    test_specific_accessers ()

    test_sync_file ()
```

virttest.utils_conn module

connection tools to manage kinds of connection.

```
exception virttest.utils_conn.ConnCertError (cert, output)
    Bases: virttest.utils_conn.ConnectionError
```

Error in building certificate file with certtool command.

```
exception virttest.utils_conn.ConnCmdClientError (cmd, output)
    Bases: virttest.utils_conn.ConnectionError
```

Error in executing cmd on client.

```
exception virttest.utils_conn.ConnCopyError (src_path, dest_path)
    Bases: virttest.utils_conn.ConnectionError
```

Error in coping file.

```
exception virttest.utils_conn.ConnForbiddenError (detail)
    Bases: virttest.utils_conn.ConnectionError
```

Error in forbidden operation.

```
exception virttest.utils_conn.ConnLoginError (dest, detail)
    Bases: virttest.utils_conn.ConnectionError
```

Error in login.

```
exception virttest.utils_conn.ConnMkdirError (directory, output)
    Bases: virttest.utils_conn.ConnectionError
```

Error in making directory.

```
exception virttest.utils_conn.ConnNotImplementedError (method_type, class_type)
    Bases: virttest.utils_conn.ConnectionError
```

Error in calling unimplemented method

```
exception virttest.utils_conn.ConnPrivKeyError (key, output)
    Bases: virttest.utils_conn.ConnectionError
```

Error in building private key with certtool command.

```
exception virttest.utils_conn.ConnRmCertError (cert, output)
    Bases: virttest.utils_conn.ConnectionError
```

Error in removing certificate file with rm command.

exception `virttest.utils_conn.ConnSCPError` (*src_ip, src_path, dest_ip, dest_path, detail*)

Bases: `virttest.utils_conn.ConnectionError`

Error in SCP.

exception `virttest.utils_conn.ConnServerRestartError` (*output*)

Bases: `virttest.utils_conn.ConnectionError`

Error in restarting libvirtd on server.

exception `virttest.utils_conn.ConnToolNotFoundError` (*tool, detail*)

Bases: `virttest.utils_conn.ConnectionError`

Error in not found tools.

class `virttest.utils_conn.ConnectionBase` (**args, **dargs*)

Bases: `virttest.propcan.PropCanBase`

Base class of a connection between server and client.

Connection is build to from client to server. And there are some information for server and client in ConnectionBase.

auto_recover

client_ip

client_pwd

client_session

client_user

close_session()

If some session exists, close it down.

conn_check()

waiting for implemented by subclass.

conn_recover()

waiting for implemented by subclass.

conn_setup()

waiting for implemented by subclass.

del_client_session()

Delete client session.

del_server_session()

Delete server session.

get_client_session()

If the client session exists,return it. else create a session to client and set client_session.

get_server_session()

If the server session exists,return it. else create a session to server and set server_session.

server_ip

server_pwd

server_session

server_user

set_client_session (*value*)

Set client session to value.

set_server_session (*value=None*)

Set server session to value.

tmp_dir

exception `virttest.utils_conn.ConnectionError`

Bases: `exceptions.Exception`

The base error in connection.

exception `virttest.utils_conn.SSHCheckError` (*server_ip, output*)

Bases: `virttest.utils_conn.ConnectionError`

Base Error in check of SSH connection.

class `virttest.utils_conn.SSHConnection` (**args, **dargs*)

Bases: `virttest.utils_conn.ConnectionBase`

Connection of SSH transport.

Some specific variables in SSHConnection class.

`ssh_rsa_pub_path`: Path of `id_rsa.pub`, default is `/root/.ssh/id_rsa.pub`. `ssh_id_rsa_path`: Path of `id_rsa`, default is `/root/.ssh/id_rsa`. `SSH_KEYGEN`, `SSH_ADD`, `SSH_COPY_ID`, `SSH_AGENT`, `SHELL`, `SSH`: tools to build a non-pwd connection.

SHELL

SSH

SSH_ADD

SSH_AGENT

SSH_COPY_ID

SSH_KEYGEN

conn_check ()

Check the SSH connection.

(1).Initialize some variables. (2).execute ssh command to check conn.

conn_recover ()

Clean up authentication host.

conn_setup ()

Setup of SSH connection.

(1).Initialization of some variables. (2).Check tools. (3).Initialization of `id_rsa`. (4).set a `ssh_agent`. (5).copy pub key to server.

ssh_id_rsa_path

ssh_rsa_pub_path

exception `virttest.utils_conn.SSHRmAuthKeysError` (*auth_keys, output*)

Bases: `virttest.utils_conn.ConnectionError`

Error in removing `authorized_keys` file.

class `virttest.utils_conn.TCPConnection` (**args, **dargs*)

Bases: `virttest.utils_conn.ConnectionBase`

Connection class for TCP transport.

Some specific variables for TCPConnection class.

auth_tcp

conn_recover()

Clean up for TCP connection.

(1).initialize variables. (2).Delete the RemoteFile. (3).restart libvirtd on server.

conn_setup()

Enable tcp connect of libvirtd on server.

(1).initialization for variables. (2).edit /etc/sysconfig/libvirtd on server. (3).edit /etc/libvirt/libvirtd.conf on server. (4).restart libvirtd service on server.

listen_addr

remote_libvirtdconf

remote_syslibvirtd

sasl_allowed_users

tcp_port

class virttest.utils_conn.**TLSConnection**(*args, **dargs)

Bases: *virttest.utils_conn.ConnectionBase*

Connection of TLS transport.

Some specific variables for TLSConnection class.

server_cn, client_cn, ca_cn: Info to build pki key. CERTOOL: tool to build key for TLS connection. pki_CA_dir: Dir to store CA key. libvirt_pki_dir, libvirt_pki_private_dir: Dir to store pki in libvirt. sysconfig_libvirtd_path, libvirtd_conf_path: Path of libvirt config file. hosts_path: /etc/hosts auth_tls, tls_port, listen_addr: custom TLS Auth, port and listen address tls_allowed_dn_list: DN's list are checked tls_verify_cert: disable verification, default is to always verify tls_sanity_cert: disable checks, default is to always run sanity checks custom_pki_path: custom pki path ca_cakey_path: CA certification path, sometimes need to reuse previous cert scp_new_cacert: copy new CA certification, default is to always copy restart_libvirtd: default is to restart libvirtd

CERTOOL

auth_tls

ca_cakey_path

ca_cn

cert_recover()

Do the clean up certifications work.

(1).initialize variables. (2).Delete local and remote generated certifications file.

client_cn

client_hosts

client_setup()

setup private key and certificate file for client.

(1).initialization for variables. (2).build a key for client. (3).copy files to client. (4).edit /etc/hosts on client.

conn_recover()

Do the clean up work.

(1).initialize variables. (2).Delete remote file. (3).Restart libvirtd on server.

conn_setup (*server_setup=True, client_setup=True*)

setup a TLS connection between server and client. At first check the certtool needed to setup. Then call some setup functions to complete connection setup.

custom_pki_path

libvirt_pki_dir

libvirt_pki_private_dir

listen_addr

pki_CA_dir

restart_libvirtd

scp_new_cacert

server_cn

server_libvirtdconf

server_setup ()

setup private key and certificate file for server.

(1).initialization for variables. (2).build server key. (3).copy files to server. (4).edit /etc/sysconfig/libvirtd on server. (5).edit /etc/libvirt/libvirtd.conf on server. (6).restart libvirtd service on server.

server_syslibvirtd

tls_allowed_dn_list

tls_port

tls_sanity_cert

tls_verify_cert

class virttest.utils_conn.**UNIXConnection** (*args, **dargs)

Bases: *virttest.utils_conn.ConnectionBase*

Connection class for UNIX transport.

Some specific variables for UNIXConnection class.

access_drivers

auth_unix_ro

auth_unix_rw

client_ip

client_libvirtdconf

client_pwd

client_user

conn_recover ()

Do the clean up work.

(1).Delete remote file. (2).Restart libvirtd on server.

conn_setup ()

Setup a UNIX connection.

(1).Initialize variables. (2).Update libvirtd.conf configuration. (3).Restart libvirtd on client.

restart_libvirtd

unix_sock_dir

unix_sock_group

unix_sock_ro_perms

unix_sock_rw_perms

`virttest.utils_conn.build_CA(tmp_dir, cn='AUTOTEST.VIRT', ca_cakey_path=None, certtool='certtool')`

setup private key and certificate file which are needed to build. certificate file for client and server.

(1).initialization for variables. (2).make a private key with certtool command. (3).prepare a info file. (4).make a certificate file with certtool command.

`virttest.utils_conn.build_client_key(tmp_dir, client_cn='TLSClient', certtool='certtool')`

(1).initialization for variables. (2).make a private key with certtool command. (3).prepare a info file. (4).make a certificate file with certtool command.

`virttest.utils_conn.build_server_key(tmp_dir, server_cn='TLSServer', certtool='certtool', ca_cakey_path=None)`

(1).initialization for variables. (2).make a private key with certtool command. (3).prepare a info file. (4).make a certificate file with certtool command.

virttest.utils_disk module

Virtualization test - Virtual disk related utility functions

copyright Red Hat Inc.

class `virttest.utils_disk.CdromDisk(path, tmpdir)`

Bases: `virttest.utils_disk.Disk`

Represents a CDROM disk that we can master according to our needs.

close (*args, **kwargs)

setup_virtio_win2008 (virtio_floppy, cdrom_virtio)

Setup the install cdrom with the virtio storage drivers, win2008 style.

Win2008, Vista and 7 require people to point out the path to the drivers on the unattended file, so we just need to copy the drivers to the extra cdrom disk. Important to note that it's possible to specify drivers from a CDROM, so the floppy driver copy is optional. Process:

1.Copy the virtio drivers on the virtio floppy to the install cdrom, if there is one available

class `virttest.utils_disk.CdromInstallDisk(path, tmpdir, source_cdrom, extra_params)`

Bases: `virttest.utils_disk.Disk`

Represents a install CDROM disk that we can master according to our needs.

close (*args, **kwargs)

get_answer_file_path (filename)

class `virttest.utils_disk.Disk`

Bases: `object`

Abstract class for Disk objects, with the common methods implemented.

close ()

copy_to (src)

get_answer_file_path (*filename*)

class `virttest.utils_disk.FloppyDisk` (**args, **kwargs*)

Bases: `virttest.utils_disk.Disk`

Represents a floppy disk. We can copy files to it, and setup it in convenient ways.

close ()

Copy everything that is in the mountpoint to the floppy.

copy_to (*src*)

setup_virtio_win2003 (*virtio_floppy, virtio_oemsetup_id*)

Setup the install floppy with the virtio storage drivers, win2003 style.

Win2003 and WinXP depend on the file `txtsetup.oem` file to install the virtio drivers from the floppy, which is a .ini file. Process:

- 1.Copy the virtio drivers on the virtio floppy to the install floppy
- 2.Parse the ini file with config parser
- 3.Modify the identifier of the default session that is going to be executed on the config parser object
- 4.Re-write the config file to the disk

setup_virtio_win2008 (*virtio_floppy*)

Setup the install floppy with the virtio storage drivers, win2008 style.

Win2008, Vista and 7 require people to point out the path to the drivers on the unattended file, so we just need to copy the drivers to the driver floppy disk. Important to note that it's possible to specify drivers from a CDROM, so the floppy driver copy is optional. Process:

- 1.Copy the virtio drivers on the virtio floppy to the install floppy, if there is one available

class `virttest.utils_disk.GuestFSModiDisk` (*disk*)

Bases: `object`

class of guest disk using guestfs lib to do some operation(like read/write) on guest disk:

mount_all ()

mounts ()

os_inspects ()

read_file (*file_name*)

read file from the guest disk, return the content of the file

Parameters *file_name* – the file you want to read.

replace_image_file_content (*file_name, find_con, rep_con*)

replace file content matchs in the file with rep_con. suport using Regular expression

Parameters

- **file_name** – the file you want to replace
- **find_con** – the origin content you want to replace.
- **rep_con** – the replace content you want.

umount_all ()

write_to_image_file (*file_name, content, w_append=False*)

Write content to the file on the guest disk.

When using this method all the original content will be overriding. if you don't hope your original data be override set `w_append=True`.

Parameters

- **file_name** – the file you want to write
- **content** – the content you want to write.
- **w_append** – append the content or override

`virttest.utils_disk.copytree(src, dst, overwrite=True, ignore='')`
Copy dirs from source to target.

Parameters

- **src** – source directory
- **dst** – destination directory
- **overwrite** – overwrite file if exist or not
- **ignore** – files want to ignore

`virttest.utils_disk.is_mount(src, dst)`
Check is src or dst mounted.

Parameters

- **src** – source device or directory, if None will skip to check
- **dst** – mountpoint, if None will skip to check

Returns if mounted mountpoint or device, else return False

`virttest.utils_disk.mount(src, dst, fstype=None, options=None, verbose=False)`
Mount src under dst if it's really mounted, then remount with options.

Parameters

- **src** – source device or directory, if None will skip to check
- **dst** – mountpoint, if None will skip to check
- **fstype** – filesystem type need to mount

Returns if mounted return True else return False

`virttest.utils_disk.umount(src, dst, verbose=False)`
Umount src from dst, if src really mounted under dst.

Parameters

- **src** – source device or directory, if None will skip to check
- **dst** – mountpoint, if None will skip to check

Returns if unmounted return True else return False

virttest.utils_env module

class `virttest.utils_env.Env` (*filename=None, version=0*)
Bases: `UserDict.IterableUserDict`

A dict-like object containing global objects used by tests.

clean_objects ()

Destroy all objects registered in this Env object.

create_vm (*vm_type, target, name, params, bindir*)

Create and register a VM in this Env object

destroy ()

Destroy all objects stored in Env and remove the backing file.

get_all_vms ()

Return a list of all VM objects in this Env object.

get_lvmdev (*name*)

Get lvm device object by name from env;

Parameters **name** – lvm device object name;

Returns lvmdev object

get_syncserver (*port*)

Return a Sync Server object by its port.

Parameters **port** – Sync Server port.

get_vm (*name*)

Return a VM object by its name.

Parameters **name** – VM name.

register_lvmdev (**args, **kwargs*)

Register lvm device object into env;

Parameters

- **name** – name of register lvmdev object
- **lvmdev** – lvmdev object;

register_syncserver (**args, **kwargs*)

Register a Sync Server in this Env object.

Parameters

- **port** – Sync Server port.
- **server** – Sync Server object.

register_vm (**args, **kwargs*)

Register a VM in this Env object.

Parameters

- **name** – VM name.
- **vm** – VM object.

save (*filename=None*)

Pickle the contents of the Env object into a file.

Parameters **filename** – Filename to pickle the dict into. If not supplied, use the filename from which the dict was loaded.

start_tcpdump (*params*)

stop_tcpdump ()

unregister_lvmdev (*args, **kwargs)

Remove lvm device object from env;

Parameters **name** – name of lvm device object;

unregister_syncserver (*args, **kwargs)

Remove a given Sync Server.

Parameters **port** – Sync Server port.

unregister_vm (*args, **kwargs)

Remove a given VM.

Parameters **name** – VM name.

exception `virttest.utils_env.EnvSaveError`

Bases: `exceptions.Exception`

`virttest.utils_env.get_env_version()`

`virttest.utils_env.lock_safe(function)`

Get the environment safe lock, run the function, then release the lock.

Unfortunately, it only works if the 1st argument of the function is an Env instance. This is mostly to save up code.

Parameters **function** – Function to wrap.

`virttest.utils_env_unittest` module

class `virttest.utils_env_unittest.FakeSyncListenServer` (*address='', port=123, tmpdir=None*)

Bases: `object`

close()

class `virttest.utils_env_unittest.FakeVm` (*vm_name, params*)

Bases: `object`

get_params()

is_alive()

class `virttest.utils_env_unittest.TestEnv` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

setUp()

tearDown()

test_get_all_vms()

- 1.Create an env object.
- 2.Create 2 vms and register them in the env.
- 3.Create a SyncListenServer and register it in the env.
- 4.Verify that the 2 vms are in the output of `get_all_vms`.
- 5.Verify that the sync server is not in the output of `get_all_vms`.

test_locking()

- 1.Create an env file.

2. Create a thread that creates a dict as one of env's elements, and keeps updating it, using the env save_lock attribute.
3. Try to save the environment.

test_register_syncserver()

1. Create an env file.
2. Create a SyncListenServer object and register it in the env.
3. Get that SyncListenServer with get_syncserver.
4. Verify that both objects are the same.

test_register_vm()

1. Create an env object.
2. Create a VM and register it from env.
3. Get the vm back from the env.
4. Verify that the 2 objects are the same.

test_save()

1. Verify that calling env.save() with no filename where env doesn't specify a filename will throw an EnvSaveError.
2. Register a VM in environment, save env to a file, recover env from that file, get the vm and verify that the instance attribute of the 2 objects is the same.
3. Register a SyncListenServer and don't save env. Restore env from file and try to get the syncserver, verify it doesn't work.
4. Now save env to a file, restore env from file and verify that the syncserver can be found there, and that the sync server instance attribute is equal to the initial sync server instance.

test_unregister_syncserver()

Unregister a sync server.

1. Create an env file.
2. Create and register 2 SyncListenServers in the env.
3. Get one of the SyncListenServers in the env.
4. Unregister one of the SyncListenServers.
5. Verify that the SyncListenServer unregistered can't be retrieved anymore with get_syncserver().

test_unregister_vm()

1. Create an env object.
2. Register 2 vms to the env.
3. Verify both vms are in the env.
4. Remove one of those vms.
5. Verify that the removed vm is no longer in env.

virttest.utils_gdb module**class** `virttest.utils_gdb.GDB` (*command=None*)Bases: `virttest.aexpect.Expect`

Class to manipulate a inferior process in gdb.

back_trace ()

Get current backtrace stack as a list of lines.

cmd (*command, cont=True*)

Call a gdb of GDB/MI command.

Parameters

- **command** – Command line to be called
- **cont** – Whether continue the inferior after calling the command

cont ()

Continue a stopped inferior.

exit ()

Exit the gdb session.

insert_break (*break_func*)

Insert a function breakpoint.

Parameters **break_func** – Function at which breakpoint inserted**kill** ()

Kill inferior by sending a SIGTERM signal.

run (*arg_str=''*)

Start the inferior with an optional argument string.

Parameters **arg_str** – Argument the inferior to be called with**send_signal** (*signal_name*)

Send a signal to the inferior.

Parameters **signal_name** – Signal name as a string or integer**set_callback** (*callback_type, func, params=None*)

Set a callback function to a customized function.

Parameters

- **callback_type** – Could be one of “stop”, “start”, “termination”, “break” or “signal”
- **func** – Function to be set as callback
- **params** – Parameters to be passed to callback function

stop ()

Stop inferior by sending a SIGINT signal.

wait_for_start (*timeout=60*)

Wait the inferior to start.

Parameters **timeout** – Max time to wait**wait_for_stop** (*timeout=60*)

Wait the inferior to be stopped.

Parameters **timeout** – Max time to wait

wait_for_termination (*timeout=60*)

Wait the gdb session to be exited.

Parameters *timeout* – Max time to wait

exception `virttest.utils_gdb.GDBCmdError` (*command, msg*)

Bases: `virttest.utils_gdb.GDBError`

Exception raised when calling an gdb command.

exception `virttest.utils_gdb.GDBError`

Bases: `exceptions.Exception`

General module exception class

virttest.utils_libguestfs module

libguestfs tools test utility functions.

class `virttest.utils_libguestfs.Guestfish` (*disk_img=None, ro_mode=False, lib-
virt_domain=None, inspector=False, uri=None,
mount_options=None, run_mode='interactive'*)

Bases: `virttest.utils_libguestfs.LibguestfsBase`

Execute guestfish, using a new guestfish shell each time.

complete_cmd (*command*)

Execute built-in command in a complete guestfish command (Not a guestfish session). *command*: guestfish
[*-options*] [*commands*]

class `virttest.utils_libguestfs.GuestfishPersistent` (*disk_img=None, ro_mode=False,
libvirt_domain=None, in-
spector=False, uri=None,
mount_options=None,
run_mode='interactive'*)

Bases: `virttest.utils_libguestfs.Guestfish`

Execute operations using persistent guestfish session.

SESSION_COUNTER = 0

add_domain (*domain, libvirturi=None, readonly=False, iface=None, live=False, allowuuid=False,
readonlydisk=None*)
domain/add-domain - add the disk(s) from a named libvirt domain

This function adds the disk(s) attached to the named libvirt domain “dom”. It works by connecting to libvirt, requesting the domain and domain XML from libvirt, parsing it for disks, and calling “add_drive_opts” on each one.

add_drive (*filename*)

add-drive - add an image to examine or modify

This function is the equivalent of calling “add_drive_opts” with no optional parameters, so the disk is added writable, with the format being detected automatically.

add_drive_opts (*filename, readonly=False, format=None, iface=None, name=None, label=None,
protocol=None, server=None, username=None, secret=None, cachemode=None,
discard=None, copyonread=False*)

add-drive-opts - add an image to examine or modify.

This function adds a disk image called “filename” to the handle. “filename” may be a regular host file or a host device.

add_drive_ro (*filename*)

add-ro/add-drive-ro - add a drive in snapshot mode (read-only)

This function is the equivalent of calling “add_drive_opts” with the optional parameter “GUESTFS_ADD_DRIVE_OPTS_READONLY” set to 1, so the disk is added read-only, with the format being detected automatically.

add_drive_ro_with_if (*filename, iface*)

add-drive-ro-with-if - add a drive read-only specifying the QEMU block emulation to use

This is the same as “add_drive_ro” but it allows you to specify the QEMU interface emulation to use at run time.

add_drive_scratch (*size, name=None, label=None*)

add-drive-scratch - add a temporary scratch drive

This command adds a temporary scratch drive to the handle. The “size” parameter is the virtual size (in bytes). The scratch drive is blank initially (all reads return zeroes until you start writing to it). The drive is deleted when the handle is closed.

add_drive_with_if (*filename, iface*)

add-drive-with-if - add a drive specifying the QEMU block emulation to use

This is the same as “add_drive” but it allows you to specify the QEMU interface emulation to use at run time.

alloc (*filename, size*)

alloc - allocate and add a disk file

This creates an empty (zeroed) file of the given size, and then adds so it can be further examined.

aug_clear (*augpath*)

aug-clear - clear Augeas path

Set the value associated with “path” to “NULL”. This is the same as the augtool(1) “clear” command.

aug_close ()

aug-close - close the current Augeas handle and free up any resources used by it.

After calling this, you have to call “aug_init” again before you can use any other Augeas functions.

aug_defnode (*node, expr, value*)

aug-defnode - defines a variable “name” whose value is the result of evaluating “expr”.

If “expr” evaluates to an empty nodeset, a node is created, equivalent to calling “aug_set” “expr”, “value”. “name” will be the nodeset containing that single node.

On success this returns a pair containing the number of nodes in the nodeset, and a boolean flag if a node was created.

aug_defvar (*name, expr*)

aug-defvar - define an Augeas variable

Defines an Augeas variable “name” whose value is the result of evaluating “expr”. If “expr” is NULL, then “name” is undefined.

On success this returns the number of nodes in “expr”, or 0 if “expr” evaluates to something which is not a nodeset.

aug_get (*augpath*)

aug-get - look up the value of an Augeas path

Look up the value associated with “path”. If “path” matches exactly one node, the “value” is returned.

aug_init (*root, flags*)

aug-init - create a new Augeas handle

Create a new Augeas handle for editing configuration files. If there was any previous Augeas handle associated with this guestfs session, then it is closed.

aug_insert (*augpath, label, before*)

aug-insert - insert a sibling Augeas node

Create a new sibling “label” for “path”, inserting it into the tree before or after “path” (depending on the boolean flag “before”).

“path” must match exactly one existing node in the tree, and “label” must be a label, ie. not contain “/”, “*” or end with a bracketed index “[N]”.

aug_label (*augpath*)

aug-label - return the label from an Augeas path expression

The label (name of the last element) of the Augeas path expression “augpath” is returned. “augpath” must match exactly one node, else this function returns an error.

aug_load ()

aug-load - load files into the tree

Load files into the tree. See “aug_load” in the Augeas documentation for the full gory details.

aug_ls (*augpath*)

aug-ls - list Augeas nodes under augpath

This is just a shortcut for listing “aug_match” “path/*” and sorting the resulting nodes into alphabetical order.

aug_match (*augpath*)

aug-match - return Augeas nodes which match augpath

Returns a list of paths which match the path expression “path”. The returned paths are sufficiently qualified so that they match exactly one node in the current tree.

aug_mv (*src, dest*)

aug-mv - move Augeas node

Move the node “src” to “dest”. “src” must match exactly one node. “dest” is overwritten if it exists.

aug_rm (*augpath*)

aug-rm - remove an Augeas path

Remove “path” and all of its children. On success this returns the number of entries which were removed.

aug_save ()

aug-save - write all pending Augeas changes to disk

This writes all pending changes to disk. The flags which were passed to “aug_init” affect exactly how files are saved.

aug_set (*augpath, val*)

aug-set - set Augeas path to value

Set the value associated with “path” to “val”.

In the Augeas API, it is possible to clear a node by setting the value to NULL. Due to an oversight in the libguestfs API you cannot do that with this call. Instead you must use the “aug_clear” call.

aug_setm (*base, sub, val*)

aug-setm - set multiple Augeas nodes

available (*groups*)

available - test availability of some parts of the API

This command is used to check the availability of some groups of functionality in the appliance, which not all builds of the libguestfs appliance will be able to provide.

available_all_groups ()

available-all-groups - return a list of all optional groups

This command returns a list of all optional groups that this daemon knows about. Note this returns both supported and unsupported groups. To find out which ones the daemon can actually support you have to call “available” / “feature_available” on each member of the returned list.

blkid (*device*)

blkid - print block device attributes

This command returns block device attributes for “device”. The following fields are usually present in the returned hash. Other fields may also be present.

blockdev_flushbufs (*device*)

blockdev-flushbufs - flush device buffers

This tells the kernel to flush internal buffers associated with “device”.

blockdev_getbsz (*device*)

blockdev-getbsz - get blocksize of block device

This returns the block size of a device.

blockdev_getro (*device*)

blockdev-getro - is block device set to read-only

Returns a boolean indicating if the block device is read-only (true if read-only, false if not).

blockdev_getsize64 (*device*)

blockdev-getsize64 - get total size of device in bytes

This returns the size of the device in bytes

blockdev_getss (*device*)

blockdev-getss - get sectorsize of block device

This returns the size of sectors on a block device. Usually 512, but can be larger for modern devices.

blockdev_getsz (*device*)

blockdev-getsz - get total size of device in 512-byte sectors

This returns the size of the device in units of 512-byte sectors (even if the sectorsize isn’t 512 bytes ... weird).

blockdev_rereadpt (*device*)

blockdev-rereadpt - reread partition table

Reread the partition table on “device”.

blockdev_setbsz (*device*, *blocksize*)

blockdev-setbsz - set blocksize of block device

This sets the block size of a device.

blockdev_setro (*device*)

blockdev-setro - set block device to read-only

Sets the block device named “device” to read-only.

blockdev_setrw (*device*)

blockdev-setrw - set block device to read-write

Sets the block device named “device” to read-write.

canonical_device_name (*device*)

canonical-device-name - return canonical device name

This utility function is useful when displaying device names to the user.

case_sensitive_path (*path*)

case-sensitive-path - return true path on case-insensitive filesystem

The “drop-caches” command can be used to resolve case insensitive paths on a filesystem which is case sensitive. The use case is to resolve paths which you have read from Windows configuration files or the Windows Registry, to the true path.

cat (*path*)

cat - list the contents of a file

Return the contents of the file named “path”.

checksum (*csumtype, path*)

checksum - compute MD5, SHAx or CRC checksum of file

This call computes the MD5, SHAx or CRC checksum of the file named “path”.

checksum_device (*csumtype, device*)

checksum-device - compute MD5, SHAx or CRC checksum of the contents of a device

This call computes the MD5, SHAx or CRC checksum of the contents of the device named “device”. For the types of checksums supported see the “checksum” command.

checksums_out (*csumtype, directory, sumsfile*)

checksums-out - compute MD5, SHAx or CRC checksum of files in a directory

This command computes the checksums of all regular files in “directory” and then emits a list of those checksums to the local output file “sumsfile”.

chmod (*mode, path*)

chmod - change file mode

Change the mode (permissions) of “path” to “mode”. Only numeric modes are supported.

chown (*owner, group, path*)

chown - change file owner and group

Change the file owner to “owner” and group to “group”.

close_session ()

If a persistent session exists, close it down.

command (*cmd*)

command - run a command from the guest filesystem

This call runs a command from the guest filesystem. The filesystem must be mounted, and must contain a compatible operating system (ie. something Linux, with the same or compatible processor architecture).

command_lines (*cmd*)

command-lines - run a command, returning lines

This is the same as “command”, but splits the result into a list of lines.

compress_device_out (*ctype, device, zdevice*)

compress-device-out - output compressed device

This command compresses “device” and writes it out to the local file “zdevice”.

The “ctype” and optional “level” parameters have the same meaning as in “compress_out”.

compress_out (*ctype, file, zfile*)

compress-out - output compressed file

This command compresses “file” and writes it out to the local file “zfile”.

The compression program used is controlled by the “ctype” parameter. Currently this includes: “compress”, “gzip”, “bzip2”, “xz” or “lzop”. Some compression types may not be supported by particular builds of libguestfs, in which case you will get an error containing the substring “not supported”.

The optional “level” parameter controls compression level. The meaning and default for this parameter depends on the compression program being used.

config (*hvparam, hvvalue*)

config - add hypervisor parameters

This can be used to add arbitrary hypervisor parameters of the form *-param value*. Actually it’s not quite arbitrary - we prevent you from setting some parameters which would interfere with parameters that we use.

copy_in (*local, remotedir*)

copy-in - copy local files or directories into an image

“copy-in” copies local files or directories recursively into the disk image, placing them in the directory called “/remotedir” (which must exist).

copy_out (*remote, localdir*)

copy-out - copy remote files or directories out of an image

“copy-out” copies remote files or directories recursively out of the disk image, placing them on the host disk in a local directory called “localdir” (which must exist).

copy_size (*src, dest, size*)

copy-size - copy size bytes from source to destination using dd

This command copies exactly “size” bytes from one source device or file “src” to another destination device or file “dest”.

cp (*src, dest*)

cp - copy a file

This copies a file from “src” to “dest” where “dest” is either a destination filename or destination directory.

cp_a (*src, dest*)

cp-a - copy a file or directory recursively

This copies a file or directory from “src” to “dest” recursively using the “cp -a” command.

dd (*src, dest*)

dd - copy from source to destination using dd

This command copies from one source device or file “src” to another destination device or file “dest”. Normally you would use this to copy to or from a device or partition, for example to duplicate a filesystem

debug (*subcmd, extraargs*)

debug - debugging and internals

The “debug” command exposes some internals of “guestfsd” (the guestfs daemon) that runs inside the hypervisor.

delete_event (*name*)

delete-event - delete a previously registered event handler

Delete the event handler which was previously registered as “name”. If multiple event handlers were registered with the same name, they are all deleted.

device_index (*device*)

device-index - convert device to index

This function takes a device name (eg. “/dev/sdb”) and returns the index of the device in the list of devices

df ()

df - report file system disk space usage

This command runs the “df” command to report disk space used.

df_h ()

df-h - report file system disk space usage (human readable)

This command runs the “df -h” command to report disk space used in human-readable format.

disk_format (*filename*)

disk-format - detect the disk format of a disk image

Detect and return the format of the disk image called “filename”, “filename” can also be a host device, etc

disk_has_backing_file (*filename*)

disk-has-backing-file - return whether disk has a backing file

Detect and return whether the disk image “filename” has a backing file

disk_virtual_size (*filename*)

disk-virtual-size - return virtual size of a disk

Detect and return the virtual size in bytes of the disk image”

dmesg ()

dmesg - return kernel messages

This returns the kernel messages (“dmesg” output) from the guest kernel. This is sometimes useful for extended debugging of problems.

do_mount (*mountpoint*)

do_mount - Automatically mount

Mount a lvm or physical partition to ‘/’

download (*remotefilename*, *filename*)

download - download a file to the local machine

Download file “remotefilename” and save it as “filename” on the local machine.

download_offset (*remotefilename*, *filename*, *offset*, *size*)

download-offset - download a file to the local machine with offset and size

Download file “remotefilename” and save it as “filename” on the local machine.

drop_caches (*whattodrop*)

drop-caches - drop kernel page cache, dentries and inodes

The “drop-caches” command instructs the guest kernel to drop its page cache, and/or dentries and inode caches. The parameter “whattodrop” tells the kernel what precisely to drop.

du (*path*)

du - estimate file space usage

This command runs the “du -s” command to estimate file space usage for “path”.

e2fsck (*device*, *correct=None*, *forceall=None*)

e2fsck - check an ext2/ext3 filesystem

This runs the ext2/ext3 filesystem checker on “device”. It can take the following optional arguments:

e2fsck_f (*device*)

e2fsck-f - check an ext2/ext3 filesystem

This runs “e2fsck -p -f device”, ie. runs the ext2/ext3 filesystem checker on “device”, noninteractively (-p), even if the filesystem appears to be clean (-f).

echo (*params=None*)

echo - display a line of text

This echos the parameters to the terminal.

echo_daemon (*words*)

echo-daemon - echo arguments back to the client

This command concatenates the list of “words” passed with single spaces between them and returns the resulting string.

egrep (*regex*, *path*)

egrep - return lines matching a pattern

This calls the external “egrep” program and returns the matching lines.

egrepi (*regex*, *path*)

egrepi - return lines matching a pattern

This calls the external “egrep -i” program and returns the matching lines.

equal (*file1*, *file2*)

equal - test if two files have equal contents

This compares the two files “file1” and “file2” and returns true if their content is exactly equal, or false otherwise.

event (*name*, *eventset*, *script*)

event - register a handler for an event or events

Register a shell script fragment which is executed when an event is raised. See “guestfs_set_event_callback” in guestfs(3) for a discussion of the event API in libguestfs.

exists (*path*)

exists - test if file or directory exists

This returns “true” if and only if there is a file, directory (or anything) with the given “path” name

extlinux (*directory*)

extlinux - install the SYSLINUX bootloader on an ext2/3/4 or btrfs filesystem

Install the SYSLINUX bootloader on the device mounted at “directory”. Unlike “syslinux” which requires a FAT filesystem, this can be used on an ext2/3/4 or btrfs filesystem.

fallocate (*path*, *len*)

fallocate - preallocate a file in the guest filesystem

This command preallocates a file (containing zero bytes) named “path” of size “len” bytes. If the file exists already, it is overwritten.

fallocate64 (*path*, *len*)

fallocate - preallocate a file in the guest filesystem

This command preallocates a file (containing zero bytes) named “path” of size “len” bytes. If the file exists already, it is overwritten.

feature_available (*groups*)

feature-available - test availability of some parts of the API

This is the same as “available”, but unlike that call it returns a simple true/false boolean result, instead of throwing an exception if a feature is not found. For other documentation see “available”.

fgrep (*pattern, path*)

fgrep - return lines matching a pattern

This calls the external “fgrep” program and returns the matching lines.

fgrepi (*pattern, path*)

fgrepi - return lines matching a pattern

This calls the external “fgrep -i” program and returns the matching lines.

file (*path*)

file - determine file type

This call uses the standard file(1) command to determine the type or contents of the file.

file_architecture (*filename*)

file-architecture - detect the architecture of a binary file

This detects the architecture of the binary “filename”, and returns it if known.

filesize (*file*)

filesize - return the size of the file in bytes

This command returns the size of “file” in bytes.

filesystem_available (*filesystem*)

filesystem-available - check if filesystem is available

Check whether libguestfs supports the named filesystem. The argument “filesystem” is a filesystem name, such as “ext3”.

fill (*c, len, path*)

fill - fill a file with octets

This command creates a new file called “path”. The initial content of the file is “len” octets of “c”, where “c” must be a number in the range “[0..255]”.

fill_dir (*dir, nr*)

fill-dir - fill a directory with empty files

This function, useful for testing filesystems, creates “nr” empty files in the directory “dir” with names 00000000 through “nr-1” (ie. each file name is 8 digits long padded with zeroes).

fill_pattern (*pattern, len, path*)

fill-pattern - fill a file with a repeating pattern of bytes

This function is like “fill” except that it creates a new file of length “len” containing the repeating pattern of bytes in “pattern”. The pattern is truncated if necessary to ensure the length of the file is exactly “len” bytes.

findfs_label (*label*)

findfs-label - find a filesystem by label

This command searches the filesystems and returns the one which has the given label. An error is returned if no such filesystem can be found.

findfs_uuid (*uuid*)

findfs-uuid - find a filesystem by UUID

This command searches the filesystems and returns the one which has the given UUID. An error is returned if no such filesystem can be found.

fsck (*fstype, device*)

fsck - run the filesystem checker

This runs the filesystem checker (fsck) on “device” which should have filesystem type “fstype”.

get_append ()

get-append - get the additional kernel options

Return the additional kernel options which are added to the libguestfs appliance kernel command line.

get_attach_method ()

get-attach-method - get the backend

Return the current backend.

get_autosync ()

get-autosync - get autosync mode

Get the autosync flag.

get_backend ()

get-backend - get the backend

Return the current backend.

get_direct ()

get-direct - get direct appliance mode flag

Return the direct appliance mode flag.

get_e2attrs (*file*)

get-e2attrs - get ext2 file attributes of a file

This returns the file attributes associated with “file”.

get_e2generation (*file*)

get-e2generation - get ext2 file generation of a file

This returns the ext2 file generation of a file. The generation (which used to be called the “version”) is a number associated with an inode. This is most commonly used by NFS servers.

get_e2label (*device*)

get-e2label - get the ext2/3/4 filesystem label

This returns the ext2/3/4 filesystem label of the filesystem on “device”.

get_e2uuid (*device*)

get-e2uuid - get the ext2/3/4 filesystem UUID

This returns the ext2/3/4 filesystem UUID of the filesystem on “device”.

get_memsize ()

get-memsize - get memory allocated to the hypervisor

This gets the memory size in megabytes allocated to the hypervisor.

get_network ()

get-network - get enable network flag

This returns the enable network flag.

get_path()
get-path - get the search path
Return the current search path.

get_pgroup()
get-pgroup - get process group flag
This returns the process group flag.

get_pid()
get-pid - get PID of hypervisor
Return the process ID of the hypervisor. If there is no hypervisor running, then this will return an error.

get_program()
get-program - get the program name
Get the program name. See “set_program”.

get_qemu()
get-qemu - get the hypervisor binary (usually qemu)
Return the current hypervisor binary (usually qemu).

get_recovery_proc()
get-recovery-proc - get recovery process enabled flag
Return the recovery process enabled flag.

get_smp()
get-smp - get number of virtual CPUs in appliance
This returns the number of virtual CPUs assigned to the appliance.

get_trace()
get-trace - get command trace enabled flag
Return the command trace flag.

get_umask()
get-umask - get the current umask
Return the current umask. By default the umask is 022 unless it has been set by calling “umask”.

get_verbose()
get-verbose - get verbose mode
This returns the verbose messages flag.

glob(*command*, *args*)
glob - expand wildcards in command
Expand wildcards in any paths in the args list, and run “command” repeatedly on each matching path.

glob_expand(*path*)
glob-expand - expand a wildcard path
This command searches for all the pathnames matching “pattern” according to the wildcard expansion rules used by the shell.

grep(*regex*, *path*)
grep - return lines matching a pattern
This calls the external “grep” program and returns the matching lines.

grep *(regex, path)*

grep - return lines matching a pattern

This calls the external “grep -i” program and returns the matching lines.

grub_install *(root, device)*

grub-install root device

This command installs GRUB 1 (the Grand Unified Bootloader) on “device”, with the root directory being “root”.

head *(path)*

head - return first 10 lines of a file

This command returns up to the first 10 lines of a file as a list of strings.

head_n *(nlines, path)*

head-n - return first N lines of a file

If the parameter “nlines” is a positive number, this returns the first “nlines” lines of the file “path”.

help *(orcmd=None)*

help - display a list of commands or help on a command

hexdump *(path)*

hexdump - dump a file in hexadecimal

This runs “hexdump -C” on the given “path”. The result is the human-readable, canonical hex dump of the file.

initrd_cat *(initrdpath, filename)*

initrd-cat - list the contents of a single file in an initrd

This command unpacks the file “filename” from the initrd file called “initrdpath”. The filename must be given *without* the initial “/” character.

initrd_list *(path)*

initrd-list - list files in an initrd

This command lists out files contained in an initrd.

inner_cmd *(command)*

Execute inner command of guestfish in a persistent session.

Parameters **command** – inner command to be executed.

inspect_get_arch *(root)*

inspect-get-arch - get architecture of inspected operating system

This returns the architecture of the inspected operating system.

inspect_get_distro *(root)*

inspect-get-distro - get distro of inspected operating system

This returns the distro (distribution) of the inspected operating system.

inspect_get_filesystems *(root)*

inspect-get-filesystems - get filesystems associated with inspected operating system

This returns a list of all the filesystems that we think are associated with this operating system.

inspect_get_hostname *(root)*

inspect-get-hostname - get hostname of the operating system

This function returns the hostname of the operating system as found by inspection of the guest’s configuration files.

inspect_get_major_version (*root*)

inspect-get-major-version - get major version of inspected operating system

This returns the major version number of the inspected operating system.

inspect_get_minor_version (*root*)

inspect-get-minor-version - get minor version of inspected operating system

This returns the minor version number of the inspected operating system

inspect_get_mountpoints (*root*)

inspect-get-mountpoints - get mountpoints of inspected operating system

This returns a hash of where we think the filesystems associated with this operating system should be mounted.

inspect_get_roots ()

inspect-get-roots - return list of operating systems found by last inspection

This function is a convenient way to get the list of root devices

inspect_os ()

inspect-os - inspect disk and return list of operating systems found

This function uses other libguestfs functions and certain heuristics to inspect the disk(s) (usually disks belonging to a virtual machine), looking for operating systems.

is_blockdev (*path, followsymlinks=None*)

is-blockdev - test if block device

This returns “true” if and only if there is a block device with the given “path” name

is_blockdev_opts (*path, followsymlinks=None*)

is-blockdev_opts - test if block device

This returns “true” if and only if there is a block device with the given “path” name

An alias of command is-blockdev

is_chardev (*path, followsymlinks=None*)

is-chardev - test if character device

This returns “true” if and only if there is a character device with the given “path” name.

is_chardev_opts (*path, followsymlinks=None*)

is-chardev_opts - test if character device

This returns “true” if and only if there is a character device with the given “path” name.

An alias of command is-chardev

is_config ()

is-config - is ready to accept commands

This returns true if this handle is in the “CONFIG” state

is_dir (*path, followsymlinks=None*)

is-dir - test if a directory

This returns “true” if and only if there is a directory with the given “path” name. Note that it returns false for other objects like files.

is_dir_opts (*path, followsymlinks=None*)

is-dir_opts - test if character device

This returns “true” if and only if there is a character device with the given “path” name.

An alias of command `is-dir`

is_fifo (*path*, *followsymlinks=None*)
is-fifo - test if FIFO (named pipe)

This returns “true” if and only if there is a FIFO (named pipe) with the given “path” name.

is_fifo_opts (*path*, *followsymlinks=None*)
is-fifo-opts - test if FIFO (named pipe)

This returns “true” if and only if there is a FIFO (named pipe) with the given “path” name.

An alias of command `is-fifo`

is_file (*path*, *followsymlinks=None*)
is-file - test if a regular file

This returns “true” if and only if there is a regular file with the given “path” name.

is_file_opts (*path*, *followsymlinks=None*)
is-file_opts - test if a regular file

This returns “true” if and only if there is a regular file with the given “path” name.

An alias of command `is-file`

is_lv (*device*)
is-lv - test if device is a logical volume

This command tests whether “device” is a logical volume, and returns true iff this is the case.

is_ready ()
is-ready - is ready to accept commands

This returns true if this handle is ready to accept commands (in the “READY” state).

is_socket (*path*, *followsymlinks=None*)
is-socket - test if socket

This returns “true” if and only if there is a Unix domain socket with the given “path” name.

is_socket_opts (*path*, *followsymlinks=None*)
is-socket-opts - test if socket

This returns “true” if and only if there is a Unix domain socket with the given “path” name.

An alias of command `is-socket`

is_symlink (*path*)
is-symlink - test if symbolic link

This returns “true” if and only if there is a symbolic link with the given “path” name.

is_whole_device (*device*)
is_whole_device - test if a device is a whole device

This returns “true” if and only if “device” refers to a whole block device. That is, not a partition or a logical device.

is_zero (*path*)
is-zero - test if a file contains all zero bytes

This returns true iff the file exists and the file is empty or it contains all zero bytes.

is_zero_device (*device*)
is-zero-device - test if a device contains all zero bytes

This returns true iff the device exists and contains all zero bytes. Note that for large devices this can take a long time to run.

kill_subprocess ()

kill-subprocess - kill the hypervisor

This kills the hypervisor.

launch ()

launch - launch the backend

You should call this after configuring the handle (eg. adding drives) but before performing any actions.

lcd (directory)

lcd - change working directory

Change the local directory, ie. the current directory of guestfish itself.

lchown (owner, group, path)

lchown - change file owner and group

Change the file owner to “owner” and group to “group”. This is like “chown” but if “path” is a symlink then the link itself is changed, not the target.

list_devices ()

list-devices - list the block devices

List all the block devices.

list_disk_labels ()

list-disk-labels - mapping of disk labels to devices

If you add drives using the optional “label” parameter of “add_drive_opts”, you can use this call to map between disk labels, and raw block device and partition names (like “/dev/sda” and “/dev/sda1”).

list_events ()

list-events - list event handlers

List the event handlers registered using the guestfish “event” command.

list_filesystems ()

list-filesystems - list filesystems

This inspection command looks for filesystems on partitions, block devices and logical volumes, returning a list of devices containing filesystems and their type.

list_md_devices ()

list-md-devices - list Linux md (RAID) devices

List all Linux md devices.

list_partitions ()

list-partitions - list the partitions

List all the partitions detected on all block devices.

ll (directory)

ll - list the files in a directory (long format)

List the files in “directory” (relative to the root directory, there is no cwd) in the format of ‘ls -la’.

ls (directory)

ls - list the files in a directory

List the files in “directory” (relative to the root directory, there is no cwd). The ‘.’ and ‘..’ entries are not returned, but hidden files are shown.

lstat (*path*)

lstat - get file information for a symbolic link

Returns file information for the given “path”.

lstatlist (*path, names*)

lstatlist - lstat on multiple files

This call allows you to perform the “lstat” operation on multiple files, where all files are in the directory “path”. “names” is the list of files from this directory.

lvcreate (*logvol, volgroup, mbytes*)

lvcreate - create an LVM logical volume

This creates an LVM logical volume called “logvol” on the volume group “volgroup”, with “size” megabytes.

lvm_canonical_lv_name (*lvname*)

lvm-canonical-lv-name - get canonical name of an LV

This converts alternative naming schemes for LVs that you might find to the canonical name.

lvm_clear_filter ()

lvm-clear-filter - clear LVM device filter

This undoes the effect of “lvm_set_filter”. LVM will be able to see every block device. This command also clears the LVM cache and performs a volume group scan.

lvm_remove_all ()

lvm-remove-all - remove all LVM LVs, VGs and PVs

This command removes all LVM logical volumes, volume groups and physical volumes.

lvm_set_filter (*device*)

lvm-set-filter - set LVM device filter

This sets the LVM device filter so that LVM will only be able to “see” the block devices in the list “devices”, and will ignore all other attached block devices.

lvremove (*device*)

lvremove - remove an LVM logical volume

Remove an LVM logical volume “device”, where “device” is the path to the LV, such as “/dev/VG/LV”.

lvrename (*logvol, newlogvol*)

lvrename - rename an LVM logical volume

Rename a logical volume “logvol” with the new name “newlogvol”

lvresize (*device, mbytes*)

lvresize - resize an LVM logical volume

This resizes (expands or shrinks) an existing LVM logical volume to “mbytes”.

lvresize_free (*lv, percent*)

lvresize-free - expand an LV to fill free space

This expands an existing logical volume “lv” so that it fills “pc”% of the remaining free space in the volume group. Commonly you would call this with pc = 100 which expands the logical volume as much as possible, using all remaining free space in the volume group.

lvs ()

lvs - list the LVM logical volumes (LVs)

List all the logical volumes detected.

lvs_full ()

lvs-full - list the LVM logical volumes (LVs)

List all the logical volumes detected. This is the equivalent of the `lvs(8)` command. The “full” version includes all fields.

lvuuid (device)

lvuuid - get the UUID of a logical volume

This command returns the UUID of the LVM LV “device”.

man ()

man - open the manual

Opens the manual page for guestfish.

max_disks ()

max-disks - maximum number of disks that may be added

Return the maximum number of disks that may be added to a handle

md_create (name, device, missingbitmap=None, nrdevices=None, spare=None, chunk=None, level=None)

md-create - create a Linux md (RAID) device

Create a Linux md (RAID) device named “name” on the devices in the list “devices”.

md_detail (md)

md-detail - obtain metadata for an MD device

This command exposes the output of ‘`mdadm -DY <md>`’. The following fields are usually present in the returned hash. Other fields may also be present.

md_stat (md)

md-stat - get underlying devices from an MD device

This call returns a list of the underlying devices which make up the single software RAID array device “md”.

md_stop (md)

md-stop - stop a Linux md (RAID) device

This command deactivates the MD array named “md”. The device is stopped, but it is not destroyed or zeroed.

mkdir (path)

mkdir - create a directory

Create a directory named “path”.

mkdir_mode (path, mode)

mkdir-mode - create a directory with a particular mode

This command creates a directory, setting the initial permissions of the directory to “mode”.

mkdir_p (path)

mkdir-p - create a directory and parents

Create a directory named “path”, creating any parent directories as necessary. This is like the “`mkdir -p`” shell command.

mkfifo (mode, path)

mkfifo - make FIFO (named pipe)

This call creates a FIFO (named pipe) called “path” with mode “mode”. It is just a convenient wrapper around “`mknod`”.

mkfs (*fstype, device, blocksize=None, features=None, inode=None, sectorsize=None*)

mkfs - make a filesystem This function creates a filesystem on “device”. The filesystem type is “fstype”, for example “ext3”.

mkfs_opts (*fstype, device, blocksize=None, features=None, inode=None, sectorsize=None*)

same with mkfs

mklost_and_found (*mountpoint*)

mklost-and-found - make lost+found directory on an ext2/3/4 filesystem

Make the “lost+found” directory, normally in the root directory of an ext2/3/4 filesystem. “mountpoint” is the directory under which we try to create the “lost+found” directory.

mkmountpoint (*exemptpath*)

mkmountpoint - create a mountpoint

“mkmountpoint” and “rmmountpoint” are specialized calls that can be used to create extra mountpoints before mounting the first filesystem.

mknod (*mode, devmajor, devminor, path*)

mknod - make block, character or FIFO devices

This call creates block or character special devices, or named pipes (FIFOs).

mknod_b (*mode, devmajor, devminor, path*)

mknod-b - make block device node

This call creates a block device node called “path” with mode “mode” and device major/minor “devmajor” and “devminor”. It is just a convenient wrapper around “mknod”.

mknod_c (*mode, devmajor, devminor, path*)

mknod-c - make char device node

This call creates a char device node called “path” with mode “mode” and device major/minor “devmajor” and “devminor”. It is just a convenient wrapper around “mknod”.

mkswap (*device, label=None, uuid=None*)

mkswap - create a swap partition

Create a Linux swap partition on “device”

mkswap_L (*label, device*)

mkswap-L - create a swap partition with a label

Create a swap partition on “device” with label “label”.

mkswap_U (*uuid, device*)

mkswap-U - create a swap partition with an explicit UUID

Create a swap partition on “device” with UUID “uuid”.

mkswap_file (*file*)

mkswap-file - create a swap file

Create a swap file.

modprobe (*modulename*)

modprobe - load a kernel module

This loads a kernel module in the appliance.

more (*filename*)

more - view a file

This is used to view a file.

mount (*device, mountpoint*)

mount - mount a guest disk at a position in the filesystem

Mount a guest disk at a position in the filesystem.

mount_loop (*file, mountpoint*)

mount-loop - mount a file using the loop device

This command lets you mount “file” (a filesystem image in a file) on a mount point. It is entirely equivalent to the command “mount -o loop file mountpoint”.

mount_options (*options, device, mountpoint*)

mount - mount a guest disk at a position in the filesystem

Mount a guest disk at a position in the filesystem.

mount_ro (*device, mountpoint*)

mount-ro - mount a guest disk, read-only

This is the same as the “mount” command, but it mounts the filesystem with the read-only (*-o ro*) flag.

mount_vfs (*options, vfstype, mountable, mountpoint*)

mount-vfs - mount a guest disk with mount options and vfstype

This is the same as the “mount” command, but it allows you to set both the mount options and the vfstype as for the mount(8) *-o* and *-t* flags.

mountpoints ()

mountpoints - show mountpoints

This call is similar to “mounts”. That call returns a list of devices.

mounts ()

mounts - show mounted filesystems

This returns the list of currently mounted filesystems.

new_session ()

Open new session, closing any existing

nr_devices ()

nr-devices - return number of whole block devices (disks) added

This returns the number of whole block devices that were added

ntfs_3g_probe (*rw, device*)

ntfs-3g-probe - probe NTFS volume

This command runs the ntfs-3g.probe(8) command which probes an NTFS “device” for mountability. (Not all NTFS volumes can be mounted read-write, and some cannot be mounted at all).

ntfsresize_opts (*device, size=None, force=None*)

ntfsresize - resize an NTFS filesystem

This command resizes an NTFS filesystem, expanding or shrinking it to the size of the underlying device.

open_session ()

Return session with session_id in this class.

parse_environment ()

parse-environment - parse the environment and set handle flags accordingly

Parse the program’s environment and set flags in the handle accordingly. For example if “LIBGUESTFS_DEBUG=1” then the ‘verbose’ flag is set in the handle.

parse_environment_list (*environment*)

parse-environment-list - parse the environment and set handle flags accordingly

Parse the list of strings in the argument “environment” and set flags in the handle accordingly. For example if “LIBGUESTFS_DEBUG=1” is a string in the list, then the ‘verbose’ flag is set in the handle.

part_add (*device, prlogex, startsect, endsect*)

part-add - add a partition to the device

This command adds a partition to “device”. If there is no partition table on the device, call “part_init” first.

part_del (*device, partnum*)

part-del device partnum

This command deletes the partition numbered “partnum” on “device”.

Note that in the case of MBR partitioning, deleting an extended partition also deletes any logical partitions it contains.

part_disk (*device, parttype*)

part-disk - partition whole disk with a single primary partition

This command is simply a combination of “part_init” followed by “part_add” to create a single primary partition covering the whole disk.

part_get_bootable (*device, partnum*)

part-get-bootable - return true if a partition is bootable

This command returns true if the partition “partnum” on “device” has the bootable flag set.

part_get_mbr_id (*device, partnum*)

part-get-mbr-id - get the MBR type byte (ID byte) from a partition

Returns the MBR type byte (also known as the ID byte) from the numbered partition “partnum”.

part_get_parttype (*device*)

part-get-parttype - get the partition table type

This command examines the partition table on “device” and returns the partition table type (format) being used.

part_init (*device, parttype*)

part-init - create an empty partition table

This creates an empty partition table on “device” of one of the partition types listed below. Usually “parttype” should be either “msdos” or “gpt” (for large disks).

part_list (*device*)

part-list - list partitions on a device

This command parses the partition table on “device” and returns the list of partitions found.

part_set_bootable (*device, partnum, bootable*)

part-set-bootable device partnum bootable

This sets the bootable flag on partition numbered “partnum” on device “device”. Note that partitions are numbered from 1.

part_set_mbr_id (*device, partnum, idbyte*)

part-set-mbr-id - set the MBR type byte (ID byte) of a partition

Sets the MBR type byte (also known as the ID byte) of the numbered partition “partnum” to “idbyte”. Note that the type bytes quoted in most documentation are in fact hexadecimal numbers, but usually documented without any leading “0x” which might be confusing.

part_set_name (*device, partnum, name*)

part-set-name - set partition name

This sets the partition name on partition numbered “partnum” on device “device”. Note that partitions are numbered from 1.

part_to_dev (*partition*)

part-to-dev - convert partition name to device name

This function takes a partition name (eg. “/dev/sdb1”) and removes the partition number, returning the device name (eg. “/dev/sdb”).

The named partition must exist, for example as a string returned from “list_partitions”.

part_to_partnum (*partition*)

part-to-partnum - convert partition name to partition number

This function takes a partition name (eg. “/dev/sdb1”) and returns the partition number (eg. 1).

The named partition must exist, for example as a string returned from “list_partitions”.

ping_daemon ()

ping-daemon - ping the guest daemon

This is a test probe into the guestfs daemon running inside the hypervisor. Calling this function checks that the daemon responds to the ping message, without affecting the daemon or attached block device(s) in any other way.

pread (*path, count, offset*)

pread - read part of a file

This command lets you read part of a file. It reads “count” bytes of the file, starting at “offset”, from file “path”.

pvcreate (*physvols*)

pvcreate - create an LVM physical volume

This creates an LVM physical volume called “physvols”.

pvremove (*device*)

pvremove - remove an LVM physical volume

This wipes a physical volume “device” so that LVM will no longer recognise it.

The implementation uses the “pvremove” command which refuses to wipe physical volumes that contain any volume groups, so you have to remove those first.

pvresize (*device*)

pvresize - resize an LVM physical volume

This resizes (expands or shrinks) an existing LVM physical volume to match the new size of the underlying device

pvresize_size (*device, size*)

pvresize-size - resize an LVM physical volume (with size)

This command is the same as “pvresize” except that it allows you to specify the new size (in bytes) explicitly.

pvs ()

pvs - list the LVM physical volumes (PVs)

List all the physical volumes detected. This is the equivalent of the pvs(8) command.

pvs_full ()

pvs-full - list the LVM physical volumes (PVs)

List all the physical volumes detected. This is the equivalent of the pvs(8) command. The “full” version includes all fields.

pvuuid (device)

pvuuid - get the UUID of a physical volume

This command returns the UUID of the LVM PV “device”.

quit ()

quit - quit guestfish

read_file (path)

read-file - read a file

This calls returns the contents of the file “path” as a buffer.

readdir (dir)

readdir - read directories entries

This returns the list of directory entries in directory “dir”

reopen ()

reopen - close and reopen libguestfs handle

Close and reopen the libguestfs handle. It is not necessary to use this normally, because the handle is closed properly when guestfish exits. However this is occasionally useful for testing.

resize2fs (device)

resize2fs - resize an ext2, ext3 or ext4 filesystem

This resizes an ext2, ext3 or ext4 filesystem to match the size of the underlying device.

resize2fs_M (device)

resize2fs-M - resize an ext2, ext3 or ext4 filesystem to the minimum size

This command is the same as “resize2fs”, but the filesystem is resized to its minimum size. This works like the -M option to the “resize2fs” command.

resize2fs_size (device, size)

resize2fs-size - resize an ext2, ext3 or ext4 filesystem (with size)

This command is the same as “resize2fs” except that it allows you to specify the new size (in bytes) explicitly.

rm (path)

rm - remove a file

Remove the single file “path”.

rm_rf (path)

rm-rf - remove a file or directory recursively

Remove the file or directory “path”, recursively removing the contents if its a directory. This is like the “rm -rf” shell command.

rmmountpoint (exemptpath)

rmmountpoint - remove a mountpoint

This calls removes a mountpoint that was previously created with “mkmountpoint”. See “mkmountpoint” for full details.

rsync (*src, dest, args*)

rsync - synchronize the contents of two directories

This call may be used to copy or synchronize two directories under the same libguestfs handle. This uses the rsync(1) program which uses a fast algorithm that avoids copying files unnecessarily.

rsync_in (*src, dest, args*)

rsync-in - synchronize host or remote filesystem with filesystem

This call may be used to copy or synchronize the filesystem on the host or on a remote computer with the filesystem within libguestfs. This uses the rsync(1) program which uses a fast algorithm that avoids copying files unnecessarily.

rsync_out (*src, dest, args*)

rsync-out - synchronize filesystem with host or remote filesystem

This call may be used to copy or synchronize the filesystem within libguestfs with a filesystem on the host or on a remote computer. This uses the rsync(1) program which uses a fast algorithm that avoids copying files unnecessarily.

run ()

run/launch - launch the qemu subprocess

Internally libguestfs is implemented by running a virtual machine using qemu.

run_mode**scrub_device** (*device*)

scrub-device - scrub (securely wipe) a device

This command writes patterns over “device” to make data retrieval more difficult

scrub_file (*file*)

scrub-file - scrub (securely wipe) a file

This command writes patterns over a file to make data retrieval more difficult

scrub_freespace (*dir*)

scrub-freespace - scrub (securely wipe) free space

This command creates the directory “dir” and then fills it with files until the filesystem is full, and scrubs the files as for “scrub_file”, and deletes them. The intention is to scrub any free space on the partition containing “dir”

session_id**set_append** (*append*)

set-append - add options to kernel command line

This function is used to add additional options to the libguestfs appliance kernel command line.

set_attach_method (*backend*)

set-attach-method - set the backend

Set the method that libguestfs uses to connect to the backend guestfsd daemon.

set_autosync (*autosync*)

set-autosync autosync

If “autosync” is true, this enables autosync. Libguestfs will make a best effort attempt to make filesystems consistent and synchronized when the handle is closed (also if the program exits without closing handles).

set_backend (*backend*)

set-backend - set the backend

Set the method that libguestfs uses to connect to the backend guestfsd daemon.

set_direct (*direct*)

set-direct - enable or disable direct appliance mode

If the direct appliance mode flag is enabled, then stdin and stdout are passed directly through to the appliance once it is launched.

set_e2attrs (*file, attrs, clear=None*)

set-e2attrs - set ext2 file attributes of a file

This sets or clears the file attributes “attrs” associated with the inode “file”.

set_e2generation (*file, generation*)

set-e2generation - set ext2 file generation of a file

This sets the ext2 file generation of a file.

set_e2label (*device, label*)

set-e2label - set the ext2/3/4 filesystem label

This sets the ext2/3/4 filesystem label of the filesystem on “device” to “label”. Filesystem labels are limited to 16 characters.

set_e2uuid (*device, uuid*)

set-e2uuid - set the ext2/3/4 filesystem UUID

This sets the ext2/3/4 filesystem UUID of the filesystem on “device” to “uuid”. The format of the UUID and alternatives such as “clear”, “random” and “time” are described in the tune2fs(8) manpage.

set_label (*mountable, label*)

set-label - set filesystem label

Set the filesystem label on “mountable” to “label”.

set_memsize (*memsize*)

set-memsize - set memory allocated to the hypervisor

This sets the memory size in megabytes allocated to the hypervisor. This only has any effect if called before “launch”.

set_network (*network*)

set-network - set enable network flag

If “network” is true, then the network is enabled in the libguestfs appliance. The default is false.

set_path (*searchpath*)

set-path - set the search path

Set the path that libguestfs searches for kernel and initrd.img.

set_pgroup (*pgroup*)

set-pgroup - set process group flag

If “pgroup” is true, child processes are placed into their own process group.

set_program (*program*)

set-program - set the program name

Set the program name. This is an informative string which the main program may optionally set in the handle.

set_qemu (*hv*)

set-qemu - set the hypervisor binary (usually qemu)

Set the hypervisor binary (usually qemu) that we will use.

set_recovery_proc (*recoveryproc*)

set-recovery-proc - enable or disable the recovery process

If this is called with the parameter “false” then “launch” does not create a recovery process. The purpose of the recovery process is to stop runaway hypervisor processes in the case where the main program aborts abruptly.

set_smp (*smp*)

set-smp - set number of virtual CPUs in appliance

Change the number of virtual CPUs assigned to the appliance. The default is 1. Increasing this may improve performance, though often it has no effect.

set_trace (*trace*)

set-trace - enable or disable command traces

If the command trace flag is set to 1, then libguestfs calls, parameters and return values are traced.

set_uuid (*device, uuid*)

set-uuid - set the filesystem UUID

Set the filesystem UUID on “device” to “uuid”.

set_verbose (*verbose*)

set-verbose - set verbose mode

If “verbose” is true, this turns on verbose messages.

setenv (*VAR, value*)

setenv - set an environment variable

Set the environment variable “VAR” to the string “value”.

sfdisk (*device, cyls, heads, sectors, lines*)

sfdisk - create partitions on a block device

This is a direct interface to the sfdisk(8) program for creating partitions on block devices.

This function is deprecated. In new code, use the “part-add” call instead.

Deprecated functions will not be removed from the API, but the fact that they are deprecated indicates that there are problems with correct use of these functions.

sfdiskM (*device, lines*)

sfdiskM - create partitions on a block device

This is a simplified interface to the “sfdisk” command, where partition sizes are specified in megabytes only (rounded to the nearest cylinder) and you don’t need to specify the cyls, heads and sectors parameters which were rarely if ever used anyway.

This function is deprecated. In new code, use the “part-add” call instead.

sfdisk_N (*device, partnum, cyls, heads, sectors, line*)

sfdisk-N - modify a single partition on a block device

This runs sfdisk(8) option to modify just the single partition “n” (note: “n” counts from 1).

For other parameters, see “sfdisk”. You should usually pass 0 for the cyls/heads/sectors parameters.

This function is deprecated. In new code, use the “part-add” call instead.

sfdisk_disk_geometry (*device*)

sfdisk-disk-geometry - display the disk geometry from the partition table

This displays the disk geometry of “device” read from the partition table. Especially in the case where the underlying block device has been resized, this can be different from the kernel’s idea of the geometry

sfdisk_kernel_geometry (*device*)

sfdisk-kernel-geometry - display the kernel geometry

This displays the kernel's idea of the geometry of "device".

sfdisk_l (*device*)

sfdisk-l - display the partition table

This displays the partition table on "device", in the human-readable output of the sfdisk(8) command. It is not intended to be parsed.

This function is deprecated. In new code, use the "part-list" call instead.

sh (*cmd*)

sh - run a command via the shell

This call runs a command from the guest filesystem via the guest's "/bin/sh".

sh_lines (*cmd*)

sh-lines - run a command via the shell returning lines

This is the same as "sh", but splits the result into a list of lines.

shutdown ()

shutdown - shutdown the hypervisor

This is the opposite of "launch". It performs an orderly shutdown of the backend process(es). If the autosync flag is set (which is the default) then the disk image is synchronized.

sleep (*secs*)

sleep - sleep for some seconds

Sleep for "secs" seconds.

sparse (*filename, size*)

sparse - create a sparse disk image and add

This creates an empty sparse file of the given size, and then adds so it can be further examined.

stat (*path*)

stat - get file information

Returns file information for the given "path".

statvfs (*path*)

statvfs - get file system statistics

Returns file system statistics for any mounted file system. "path" should be a file or directory in the mounted file system (typically it is the mount point itself, but it doesn't need to be).

strings (*path*)

strings - print the printable strings in a file

This runs the strings(1) command on a file and returns the list of printable strings found.

supported ()

supported - list supported groups of commands

This command returns a list of the optional groups known to the daemon, and indicates which ones are supported by this build of the libguestfs appliance.

swapoff_device (*device*)

swapoff-device - disable swap on device

This command disables the libguestfs appliance swap device or partition named "device". See "swapon_device".

swapoff_file (*file*)

swapoff-file - disable swap on file

This command disables the libguestfs appliance swap on file.

swapoff_label (*label*)

swapoff-label - disable swap on labeled swap partition

This command disables the libguestfs appliance swap on labeled swap partition.

swapoff_uuid (*uuid*)

swapoff-uuid - disable swap on swap partition by UUID

This command disables the libguestfs appliance swap partition with the given UUID.

swapon_device (*device*)

swapon-device - enable swap on device

This command enables the libguestfs appliance to use the swap device or partition named “device”. The increased memory is made available for all commands, for example those run using “command” or “sh”.

swapon_file (*file*)

swapon-file - enable swap on file

This command enables swap to a file. See “swapon_device” for other notes.

swapon_label (*label*)

swapon-label - enable swap on labeled swap partition

This command enables swap to a labeled swap partition. See “swapon_device” for other notes.

swapon_uuid (*uuid*)

swapon-uuid - enable swap on swap partition by UUID

This command enables swap to a swap partition with the given UUID. See “swapon_device” for other notes.

sync ()

lsync - sync disks, writes are flushed through to the disk image

This syncs the disk, so that any writes are flushed through to the underlying disk image.

syslinux (*device*, *directory=None*)

syslinux - install the SYSLINUX bootloader

Install the SYSLINUX bootloader on “device”.

tail (*path*)

tail - return last 10 lines of a file

This command returns up to the last 10 lines of a file as a list of strings.

tar_in (*tarfile*, *directory*)

tar-in - unpack tarfile to directory

This command uploads and unpacks local file “tarfile” (an *uncompressed* tar file) into “directory”.

tar_in_opts (*tarfile*, *directory*, *compress=None*)

tar-in-opts - unpack tarfile to directory

This command uploads and unpacks local file “tarfile” (an *compressed* tar file) into “directory”.

tar_out (*directory*, *tarfile*)

tar-out - pack directory into tarfile

This command packs the contents of “directory” and downloads it to local file “tarfile”.

time (*command*, *args=None*)

time - print elapsed time taken to run a command

Run the command as usual, but print the elapsed time afterwards. This can be useful for benchmarking operations.

touch (*path*)

touch - update file timestamps or create a new file

Touch acts like the touch(1) command. It can be used to update the timestamps on a file, or, if the file does not exist, to create a new zero-length file.

tune2fs (*device*, *force=None*, *maxmountcount=None*, *mountcount=None*, *errorbehavior=None*, *group=None*, *intervalbetweenchecks=None*, *reservedblockspercentage=None*, *lastmounteddirectory=None*, *reservedblockscount=None*, *user=None*)

tune2fs - adjust ext2/ext3/ext4 filesystem parameters

This call allows you to adjust various filesystem parameters of an ext2/ext3/ext4 filesystem called “device”.

tune2fs_l (*device*)

tune2fs-l - get ext2/ext3/ext4 superblock details

This returns the contents of the ext2, ext3 or ext4 filesystem superblock on “device”.

umask (*mask*)

umask - set file mode creation mask (umask)

This function sets the mask used for creating new files and device nodes to “mask & 0777”.

umount (*pathordevice*, *force=None*, *lazyunmount=None*)

umount - unmount a filesystem

This unmounts the given filesystem. The filesystem may be specified either by its mountpoint (path) or the device which contains the filesystem.

umount_all ()

umount-all - unmount all filesystems

This unmounts all mounted filesystems. Some internal mounts are not unmounted by this call.

unsetenv (*VAR*)

unsetenv - unset an environment variable

Remove “VAR” from the environment.

upload (*filename*, *remotefilename*)

upload - upload a file from the local machine

Upload local file “filename” to “remotefilename” on the filesystem.

upload_offset (*filename*, *remotefilename*, *offset*)

upload - upload a file from the local machine with offset

Upload local file “filename” to “remotefilename” on the filesystem.

utimens (*path*, *atsecs*, *atnsecs*, *mtsecs*, *mtnsecs*)

utimens - set timestamp of a file with nanosecond precision

This command sets the timestamps of a file with nanosecond precision.

utsname ()

utsname - appliance kernel version

This returns the kernel version of the appliance, where this is available. This information is only useful for debugging. Nothing in the returned structure is defined by the API.

version ()

version - get the library version number

Return the libguestfs version number that the program is linked against.

vfs_label (mountable)

vfs-label - get the filesystem label

This returns the label of the filesystem on “mountable”.

vfs_type (mountable)

vfs-type - get the Linux VFS type corresponding to a mounted device

Gets the filesystem type corresponding to the filesystem on “mountable”

vfs_uuid (mountable)

vfs-uuid - get the filesystem UUID

This returns the filesystem UUID of the filesystem on “mountable”.

vg_activate (activate, volgroups)

vg-activate - activate or deactivate some volume groups

This command activates or (if “activate” is false) deactivates all logical volumes in the listed volume groups “volgroups”

vg_activate_all (activate)

vg-activate-all - activate or deactivate all volume groups

This command activates or (if “activate” is false) deactivates all logical volumes in all volume groups.

vgcreate (volgroup, physvols)

vgcreate - create an LVM volume group

This creates an LVM volume group called “volgroup” from the non-empty list of physical volumes “physvols”.

vglvuuids (vgname)

vglvuuids - get the LV UUIDs of all LVs in the volume group

Given a VG called “vgname”, this returns the UUIDs of all the logical volumes created in this volume group.

vgpvuuids (vgname)

vgpvuuids - get the PV UUIDs containing the volume group

Given a VG called “vgname”, this returns the UUIDs of all the physical volumes that this volume group resides on.

vgremove (vgname)

vgremove - remove an LVM volume group

Remove an LVM volume group “vgname”, (for example “VG”).

vgrename (volgroup, newvolgroup)

vgrename - rename an LVM volume group

Rename a volume group “volgroup” with the new name “newvolgroup”.

vgs ()

vgs - list the LVM volume groups (VGs)

List all the volumes groups detected.

vgs_full ()

vgs-full - list the LVM volume groups (VGs)

List all the volumes groups detected. This is the equivalent of the vgs(8) command. The “full” version includes all fields.

vgscan ()

vgscan - rescan for LVM physical volumes, volume groups and logical volumes

This rescans all block devices and rebuilds the list of LVM physical volumes, volume groups and logical volumes.

vguuid (vgname)

vguuid - get the UUID of a volume group

This command returns the UUID of the LVM VG named “vgname”

write (path, content)

write - create a new file

This call creates a file called “path”. The content of the file is the string “content” (which can contain any 8 bit data).

write_append (path, content)

write-append - append content to end of file

This call appends “content” to the end of file “path”. If “path” does not exist, then a new file is created.

zegrep (regex, path)

zegrep - return lines matching a pattern

This calls the external “zegrep” program and returns the matching lines.

zegrepi (regex, path)

zegrepi - return lines matching a pattern

This calls the external “zegrep -i” program and returns the matching lines.

zero (device)

zero - write zeroes to the device

This command writes zeroes over the first few blocks of “device”.

zero_device (device)

zero-device - write zeroes to an entire device

This command writes zeroes over the entire “device”. Compare with “zero” which just zeroes the first few blocks of a device.

zfgrep (pattern, path)

zfgrep - return lines matching a pattern

This calls the external “zfgrep” program and returns the matching lines.

zfgrepi (pattern, path)

zfgrepi - return lines matching a pattern

This calls the external “zfgrep -i” program and returns the matching lines.

zgrep (regex, path)

zgrep - return lines matching a pattern

This calls the external “zgrep” program and returns the matching lines.

zgrep*i* (*regex*, *path*)
 zgrep - return lines matching a pattern

This calls the external “zgrep -i” program and returns the matching lines.

class `virttest.utils_libguestfs.GuestfishRemote` (*guestfs_exec=None*, *a_id=None*)
 Bases: `object`

Remote control of guestfish.

ERROR_REGEX_LIST = ['libguestfs: error:\\s*']

cmd (*cmd*, *ignore_status=False*)
 Mimic `utils.run()`

cmd_result (*cmd*, *ignore_status=False*)
 Mimic `utils.run()`

cmd_status_output (*cmd*, *ignore_status=None*, *verbose=None*, *timeout=60*)
 Send a guestfish command and return its exit status and output.

Parameters

- **cmd** – guestfish command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return

Returns A tuple (status, output) where status is the exit status and output is the output of `cmd`

Raises `LibguestfsCmdError` – Raised if commands execute failed

get_id()

class `virttest.utils_libguestfs.GuestfishSession` (*guestfs_exec=None*, *a_id=None*,
prompt='><fs>\\s'*)

Bases: `virttest.aexpect.ShellSession`

A shell session of guestfish.

ERROR_REGEX_LIST = ['libguestfs: error:\\s*']

cmd_result (*cmd*, *ignore_status=False*)
 Mimic `utils.run()`

cmd_status_output (*cmd*, *timeout=60*, *internal_timeout=None*, *print_func=None*)
 Send a guestfish command and return its exit status and output.

Parameters

- **cmd** – guestfish command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to `read_nonblocking`
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple (status, output) where status is the exit status and output is the output of `cmd`

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output
- **ShellStatusError** – Raised if the exit status cannot be obtained

- **ShellError** – Raised if an unknown error occurs

class `virttest.utils_libguestfs.LibguestfsBase` (*lgf_exec='/bin/true', ignore_status=True, debug=False, timeout=60, uri=None*)

Bases: `virttest.propcan.PropCanBase`

Base class of libguestfs tools.

debug

get_uri ()

Accessor method for 'uri' property that must exist

ignore_status

lgf_exec

set_debug (*debug*)

Accessor method for 'debug' property that logs message on change

set_ignore_status (*ignore_status*)

Enforce setting ignore_status as a boolean.

set_timeout (*timeout*)

Accessor method for 'timeout' property, timeout should be digit

timeout

uri

exception `virttest.utils_libguestfs.LibguestfsCmdError` (*details=''*)

Bases: `exceptions.Exception`

Error of libguestfs-tool command.

`virttest.utils_libguestfs.guestmount` (*disk_or_domain, mountpoint, inspector=False, read-only=False, **dargs*)

guestmount - Mount a guest filesystem on the host using FUSE and libguestfs.

Parameters

- **disk_or_domain** – a disk or a domain to be mounted If you need to mount a disk, set `is_disk` to True in dargs
- **mountpoint** – the mountpoint of filesystems
- **inspector** – mount all filesystems automatically
- **readonly** – if mount filesystem with readonly option

`virttest.utils_libguestfs.lgf_cmd_check` (*cmd*)

To check whether the cmd is supported on this host.

Parameters *cmd* – the cmd to use a libguest tool.

Returns None if the cmd is not exist, otherwise return its path.

`virttest.utils_libguestfs.lgf_command` (*cmd, ignore_status=True, debug=False, timeout=60*)

Interface of libguestfs tools' commands.

Parameters *cmd* – Command line to execute.

Returns CmdResult object.

Raise LibguestfsCmdError if non-zero exit status and ignore_status=False

```
virttest.utils_libguestfs.libguest_test_tool_cmd(qemuarg=None,      gemudi-
rarg=None,      timeoutarg=None,
ignore_status=True,  debug=False,
timeout=60)
```

Execute libguest-test-tool command.

Parameters

- **qemuarg** – the qemu option
- **gemudirarg** – the qemudir option
- **timeoutarg** – the timeout option

Returns a CmdResult object

Raise raise LibguestfsCmdError

```
virttest.utils_libguestfs.virt_cat_cmd(disk_or_domain,  file_path,  options=None,  ig-
nignore_status=True, debug=False, timeout=60)
```

Execute virt-cat command to print guest's file detail.

Parameters

- **disk_or_domain** – a img path or a domain name.
- **file_path** – the file to print detail
- **options** – the options of virt-cat.

Returns a CmdResult object.

```
virttest.utils_libguestfs.virt_clone_cmd(original,  newname=None,  autoclone=False,
**dargs)
```

Clone existing virtual machine images.

Parameters

- **original** – Name of the original guest to be cloned.
- **newname** – Name of the new guest virtual machine instance.
- **autoclone** – Generate a new guest name, and paths for new storage.
- **dargs** – Standardized function API keywords. There are many options not listed, they can be passed in dargs.

```
virttest.utils_libguestfs.virt_cmd_contain_opt(virt_cmd, opt)
```

Check if opt is supported by virt-command

```
virttest.utils_libguestfs.virt_copy_in(disk_or_domain, file, destination, is_disk=False, ig-
nignore_status=True, debug=False, timeout=60)
```

“virt-copy-in” copies files and directories from the local disk into a virtual machine disk image or named libvirt domain. #TODO: expand file to files

```
virttest.utils_libguestfs.virt_copy_out(disk_or_domain, file_path, localdir, is_disk=False,
ignore_status=True, debug=False, timeout=60)
```

“virt-copy-out” copies files and directories out of a virtual machine disk image or named libvirt domain.

```
virttest.utils_libguestfs.virt_df(disk_or_domain, ignore_status=True, debug=False, time-
out=60)
```

“virt-df” is a command line tool to display free space on virtual machine filesystems.

```
virttest.utils.libguestfs.virt_edit_cmd(disk_or_domain, file_path, is_disk=False,
                                         disk_format=None, options=None, extra=None,
                                         expr=None, connect_uri=None, ignore_status=True,
                                         debug=False, timeout=60)
```

Execute virt-edit command to check whether it is ok.

Since virt-edit will need uses' interact, maintain and return a session if there is no raise after command has been executed.

Parameters

- **disk_or_domain** – a img path or a domain name.
- **file_path** – the file need to be edited in img file.
- **is_disk** – whether disk_or_domain is disk or domain
- **disk_format** – when is_disk is true, add a format if it is set.
- **options** – the options of virt-edit.
- **extra** – additional suffix of command.

Returns a session of executing virt-edit command.

```
virttest.utils.libguestfs.virt_filesystems(disk_or_domain, **dargs)
virt-filesystems - List filesystems, partitions, block devices, LVM in a virtual machine or disk image
```

Parameters **disk_or_domain** – a disk or a domain to be mounted If you need to mount a disk, set is_disk to True in dargs

```
virttest.utils.libguestfs.virt_format(disk, filesystem=None, image_format=None,
                                      lvm=None, partition=None, wipe=False, ignore_status=False,
                                      debug=False, timeout=60)
```

Virt-format takes an existing disk file (or it can be a host partition, LV etc), erases all data on it, and formats it as a blank disk.

```
virttest.utils.libguestfs.virt_inspector(disk_or_domain, is_disk=False, ignore_status=True,
                                          debug=False, timeout=60)
```

virt-inspector2 examines a virtual machine or disk image and tries to determine the version of the operating system and other information about the virtual machine.

```
virttest.utils.libguestfs.virt_list_filesystems(disk_or_domain, format=None, long=False, all=False,
                                                ignore_status=True, debug=False, timeout=60)
```

“virt-list-filesystems” is a command line tool to list the filesystems that are contained in a virtual machine or disk image.

Parameters **disk_or_domain** – a disk or a domain to be mounted

```
virttest.utils.libguestfs.virt_list_partitions(disk_or_domain, long=False, total=False,
                                                human_readable=False, ignore_status=True,
                                                debug=False, timeout=60)
```

“virt-list-partitions” is a command line tool to list the partitions that are contained in a virtual machine or disk image.

Parameters **disk_or_domain** – a disk or a domain to be mounted


```
virttest.utils_libguestfs.virt_list_partitions_cmd(disk_or_domain, long=False,
                                                    total=False, human_readable=False,
                                                    ignore_status=True, debug=False,
                                                    timeout=60)
```

“virt-list-partitions” is a command line tool to list the partitions that are contained in a virtual machine or disk image.

Parameters **disk_or_domain** – a disk or a domain to be mounted

```
virttest.utils_libguestfs.virt_ls_cmd(disk_or_domain, file_dir_path, is_disk=False,
                                       options=None, extra=None, connect_uri=None,
                                       ignore_status=True, debug=False, timeout=60)
```

Execute virt-ls command to check whether file exists.

Parameters

- **disk_or_domain** – a img path or a domain name.
- **file_dir_path** – the file or directory need to check.

```
virttest.utils_libguestfs.virt_resize_cmd(indisk, outdisk, **dargs)
```

Resize a virtual machine disk.

Parameters

- **indisk** – The source disk to be resized
- **outdisk** – The destination disk.

```
virttest.utils_libguestfs.virt_sparsify_cmd(indisk, outdisk, compress=False,
                                             convert=None, format=None,
                                             ignore_status=True, debug=False,
                                             timeout=60)
```

Make a virtual machine disk sparse.

Parameters

- **indisk** – The source disk to be sparsified.
- **outdisk** – The destination disk.

```
virttest.utils_libguestfs.virt_sysprep_cmd(disk_or_domain, options=None, extra=None,
                                             ignore_status=True, debug=False,
                                             timeout=600)
```

Execute virt-sysprep command to reset or unconfigure a virtual machine.

Parameters

- **disk_or_domain** – a img path or a domain name.
- **options** – the options of virt-sysprep.

Returns a CmdResult object.

```
virttest.utils_libguestfs.virt_sysprep_operations()
```

Get virt-sysprep support operation

```
virttest.utils_libguestfs.virt_tar_in(disk_or_domain, tar_file, destination, is_disk=False,
                                       ignore_status=True, debug=False, timeout=60)
```

“virt-tar-in” unpacks an uncompressed tarball into a virtual machine disk image or named libvirt domain.

```
virttest.utils_libguestfs.virt_tar_out(disk_or_domain, directory, tar_file, is_disk=False,
                                       ignore_status=True, debug=False, timeout=60)
```

“virt-tar-out” packs a virtual machine disk image directory into a tarball.

virttest.utils_libguestfs_unittest module

```
class virttest.utils_libguestfs_unittest.LibguestfsTest (methodName='runTest')
    Bases: unittest.case.TestCase

    test_lgf_cmd()

    test_lgf_cmd_check()

    test_lgf_cmd_check_raises()

class virttest.utils_libguestfs_unittest.SlotsCheckTest (methodName='runTest')
    Bases: unittest.case.TestCase

    test_Guestfish_slots()
        Test Guestfish slots

    test_LibguestfsBase_default_slots()
        Default slots' value check

    test_LibguestfsBase_update_slots()
        Update slots
```

virttest.utils_libvirt module

Module to control libvirt service.

```
class virttest.utils_libvirt.Libvirt (session=None)
    Bases: object

    Class to manage libvirt service on host or guest.

    is_running()

    restart (reset_failed=True)

    start (reset_failed=True)

    stop()

class virttest.utils_libvirt.LibvirtSession (gdb=False, logging_handler=None, logging_pattern='.*')
    Bases: object

    Interaction libvirt daemon session by directly call the libvirt command. With gdb debugging feature can be optionally started.

    back_trace()
        Get the backtrace from gdb session.

    cont()
        Continue a stopped libvirt session.

    exit()
        Exit the libvirt session.

    insert_break (break_func)
        Insert a function breakpoint.

        Parameters break_func – Function at which breakpoint inserted

    is_working()
        Check if libvirt is start by return status of 'virsh list'
```

kill()

Kill the libvirtd session.

restart (*arg_str*='', *wait_for_working*=True)

Restart the libvirtd session.

Parameters

- **arg_str** – Argument passing to the session
- **wait_for_working** – Whether wait for libvirtd finish loading

set_callback (*callback_type*, *callback_func*, *callback_params*=None)

Set a customized gdb callback function.

start (*arg_str*='', *wait_for_working*=True)

Start libvirtd session.

Parameters

- **arg_str** – Argument passing to the session
- **wait_for_working** – Whether wait for libvirtd finish loading

wait_for_stop (*timeout*=60, *step*=0.1)

Wait for libvirtd to stop.

Parameters

- **timeout** – Max wait time
- **step** – Checking interval

wait_for_termination (*timeout*=60)

Wait for libvirtd gdb session to exit.

Parameters timeout – Max wait time

wait_for_working (*timeout*=60)

Wait for libvirtd to work.

Parameters timeout – Max wait time

virttest.utils_libvirtd.deprecation_warning()

As the `utils_libvirtd.libvirtd_xxx` interfaces are deprecated, this function are printing the warning to user.

virttest.utils_libvirtd.libvirtd_is_running()

virttest.utils_libvirtd.libvirtd_restart()

virttest.utils_libvirtd.libvirtd_start()

virttest.utils_libvirtd.libvirtd_stop()

virttest.utils_libvirtd.service_libvirtd_control (*action*, *session*=None)

virttest.utils_misc module

Virtualization test utility functions.

copyright 2008-2009 Red Hat Inc.

class `virttest.utils_misc.Flag`

Bases: `str`

Class for easy merge cpuflags.

```
aliases = {}
```

class `virttest.utils_misc.ForAll`
Bases: `list`

class `virttest.utils_misc.ForAllP`
Bases: `list`
Parallel version of ForAll

class `virttest.utils_misc.ForAllPSE`
Bases: `list`
Parallel version of and suppress exception.

class `virttest.utils_misc.KSMController`
Bases: `object`
KSM Manager

get_ksm_feature (*feature*)
Get ksm feature's value.

get_ksmtuned_pid ()
Return ksmtuned process id(0 means not running).

get_writable_features ()
Get writable features for setting

is_ksm_running ()
Verify whether ksm is running.

is_module_loaded ()
Check whether ksm module has been loaded.

load_ksm_module ()
Try to load ksm module.

restart_ksm (*pages_to_scan=None, sleep_ms=None*)
Restart ksm service

restart_ksmtuned ()
Restart ksmtuned service

set_ksm_feature (*feature_args*)
Set ksm features.

Parameters **feature_args** – a dict include features and their's value.

start_ksm (*pages_to_scan=None, sleep_ms=None*)
Start ksm function.

start_ksmtuned ()
Start ksmtuned service

stop_ksm ()
Stop ksm function.

stop_ksmtuned ()
Stop ksmtuned service

unload_ksm_module ()
Try to unload ksm module.

exception `virttest.utils_misc.KSMError`

Bases: `exceptions.Exception`

Base exception for KSM setup

exception `virttest.utils_misc.KSMNotSupportedError`

Bases: `virttest.utils_misc.KSMError`

Thrown when host does not support KSM.

exception `virttest.utils_misc.KSMTunedError`

Bases: `virttest.utils_misc.KSMError`

Thrown when KSMTuned Error happen.

exception `virttest.utils_misc.KSMTunedNotSupportedError`

Bases: `virttest.utils_misc.KSMTunedError`

Thrown when host does not support KSM Tune.

exception `virttest.utils_misc.LogLockError`

Bases: `exceptions.Exception`

class `virttest.utils_misc.NumaInfo` (*all_nodes_path=None, online_nodes_path=None*)

Bases: `object`

Numa topology for host. Also provide the function for check the memory status of the node.

get_all_nodes (*all_nodes_path=None*)

Get all node ids in host.

Returns All node ids in host

Return type *list*

get_node_distance (*node_id*)

Get the distance from the give node to other nodes include itself.

Parameters *node_id* (*string*) – Node that you want to check

Returns A list in of distance for the node in positive-sequence

Return type *list*

get_online_nodes (*online_nodes_path=None*)

Get node ids online in host

Returns The ids of node which is online

Return type *list*

read_from_node_meminfo (*node_id, key*)

Get specific value of a given node from meminfo file

Parameters

- *node_id* (*string*) – The node you want to check
- *key* (*string*) – The value you want to check such as MemTotal etc.

Returns The value in KB

Return type *string*

class `virttest.utils_misc.NumaNode` (*i=-1, all_nodes_path=None, online_nodes_path=None*)

Bases: `object`

Numa node to control processes and shared memory.

free_cpu (*i*, *thread=None*)
Release pin of one node.

Parameters

- **i** – Index of the node.
- **thread** – Thread ID, remove all threads if thread ID isn't set

get_cpu_topology (*cpu_id*)
Return cpu info dict get from sysfs.

Parameters **cpu_id** – integer, cpu id number

Returns topology dict of certain cpu

get_node_cpus (*i*)
Get cpus of a specific node

Parameters **i** – Index of the CPU inside the node.

pin_cpu (**args*, ***kwargs*)
Pin one process to a single cpu.

Parameters

- **process** – Process ID.
- **cpu** – CPU ID, pin thread to free CPU if cpu ID isn't set

show ()
Display the record dict in a convenient way.

class `virttest.utils_misc.SELinuxBoolean` (*params*)
Bases: `object`

SELinuxBoolean class for managing SELinux boolean value.

cleanup (*keep_authorized_keys=False*)
Cleanup SELinux boolean value.

get_sebool_local ()
Get SELinux boolean value from local host.

get_sebool_remote ()
Get SELinux boolean value from remote host.

setup ()
Set SELinux boolean value.

setup_local ()
Set SELinux boolean value on the local

setup_remote ()
Set SELinux boolean value on remote host.

exception `virttest.utils_misc.UnsupportedCPU`
Bases: `autotest.client.shared.error.TestError`

class `virttest.utils_misc.VFIOController` (*load_modules=True*, *al-*
low_unsafe_interrupts=True)
Bases: `object`

Control Virtual Function for testing

add_device_to_iommu_group (*pci_id*)
Add one single device to iommu group.

bind_device_to_iommu_group (*pci_id*)

Bind device to iommu group.

check_iommu ()

Check whether iommu group is available.

check_vfio_id (*group_id*)

Check whether given vfio group has been established.

get_iommu_group_devices (*group_id*)

Get all devices in one group by its id.

get_pci_iommu_group_id (*pci_id*, *device_type*='')

Get pci devices iommu group id

exception `virttest.utils_misc.VFIOError` (*err*)

Bases: `exceptions.Exception`

class `virttest.utils_misc.VirtLoggingConfig` (*use_console=True*)

Bases: `autotest.client.shared.logging_config.LoggingConfig`

Used with the sole purpose of providing convenient logging setup for the KVM test auxiliary programs.

configure_logging (*results_dir=None*, *verbose=False*)

`virttest.utils_misc.add_identities_into_ssh_agent` ()

Adds RSA or DSA identities to the authentication agent

`virttest.utils_misc.add_ker_cmd` (*kernel_cmdline*, *kernel_param*, *remove_similar=False*)

Add a parameter to kernel command line content

Parameters

- **kernel_cmdline** (*string*) – Original kernel command line.
- **kernel_param** (*string*) – parameter want to change include the value.
- **remove_similar** (*bool*) – remove the value of the parameter that already in kernel cmd line or not.

Returns kernel command line

Return type `string`

`virttest.utils_misc.archive_as_tarball` (*source_dir*, *dest_dir*, *tarball_name=None*, *compression='bz2'*, *verbose=True*)

Saves the given source directory to the given destination as a tarball

If the name of the archive is omitted, it will be taken from the *source_dir*. If it is an absolute path, *dest_dir* will be ignored. But, if both the destination directory and *tarball_name* is given, and the latter is not an absolute path, they will be combined.

For archiving directory '/tmp' in '/net/server/backup' as file 'tmp.tar.bz2', simply use:

```
>>> utils_misc.archive_as_tarball('/tmp', '/net/server/backup')
```

To save the file it with a different name, say 'host1-tmp.tar.bz2' and save it under '/net/server/backup', use:

```
>>> utils_misc.archive_as_tarball('/tmp', '/net/server/backup',
                                   'host1-tmp')
```

To save with gzip compression instead (resulting in the file '/net/server/backup/host1-tmp.tar.gz'), use:

```
>>> utils_misc.archive_as_tarball('/tmp', '/net/server/backup',
                                   'host1-tmp', 'gz')
```

`virttest.utils_misc.aton(sr)`

Transform a string to a number(include float and int). If the string is not in the form of number, just return false.

Parameters `sr` – string to transform

Returns float, int or False for failed transform

`virttest.utils_misc.bind_device_driver(pci_id, driver_type)`

Bind device driver.

Parameters `driver_type` – Supported drivers: igb, lpfc, vfio-pci

`virttest.utils_misc.bitlist_to_string(data)`

Transform from bit list to ASCII string.

Parameters `data` – Bit list to be transformed

`virttest.utils_misc.check_device_driver(pci_id, driver_type)`

Check whether device's driver is same as expected.

`virttest.utils_misc.check_if_vm_vcpu_match(vcpu_desire, vm)`

This checks whether the VM vCPU quantity matches the value desired.

`virttest.utils_misc.check_module(module_name, submodules=[])`

Check whether module and its submodules work.

`virttest.utils_misc.close_log_file(filename)`

`virttest.utils_misc.convert_ipv4_to_ipv6(ipv4)`

Translates a passed in string of an ipv4 address to an ipv6 address.

Parameters `ipv4` – a string of an ipv4 address

`virttest.utils_misc.cpu_str_to_list(origin_str)`

Convert the cpu string to a list. The string may include comma and hyphen.

Parameters `origin_str` (*string*) – the cpu info string read from system

Returns A list of the cpu ids

Return type *list*

`virttest.utils_misc.create_x509_dir(path, cacert_subj, server_subj, passphrase, secure=False, bits=1024, days=1095)`

Creates directory with freshly generated: ca-cart.pem, ca-key.pem, server-cert.pem, server-key.pem,

Parameters

- **path** – defines path to directory which will be created
- **cacert_subj** – ca-cert.pem subject
- **server_key.csr** – subject
- **passphrase** – passphrase to ca-key.pem
- **secure** – defines if the server-key.pem will use a passphrase
- **bits** – bit length of keys
- **days** – cert expiration

Raises

- **ValueError** – openssl not found or rc != 0
- **OSError** – if os.makedirs() fails

`virttest.utils_misc.delete_pid_file_if_exists(program_name, pid_files_dir=None)`

Tries to remove <program_name>.pid from the main autotest directory.

`virttest.utils_misc.display_attributes(instance)`

Inspects a given class instance attributes and displays them, convenient for debugging.

`virttest.utils_misc.extract_qemu_cpu_models(qemu_cpu_help_text)`

Get all cpu models from qemu -cpu help text.

Parameters `qemu_cpu_help_text` – text produced by <qemu> -cpu ‘?’

Returns list of cpu models

`virttest.utils_misc.find_command(cmd)`

Try to find a command in the PATH, paranoid version.

Parameters `cmd` – Command to be found.

Raise `ValueError` in case the command was not found.

`virttest.utils_misc.find_free_port(start_port, end_port, address='localhost')`

Return a host free port in the range [start_port, end_port].

Parameters

- **start_port** – First port that will be checked.
- **end_port** – Port immediately after the last one that will be checked.

`virttest.utils_misc.find_free_ports(start_port, end_port, count, address='localhost')`

Return count of host free ports in the range [start_port, end_port].

Parameters

- **count** – Initial number of ports known to be free in the range.
- **start_port** – First port that will be checked.
- **end_port** – Port immediately after the last one that will be checked.

`virttest.utils_misc.find_substring(string, pattern1, pattern2=None)`

Return the match of pattern1 in string. Or return the match of pattern2 if pattern1 is not matched.

Parameters

- **string** – string
- **pattern1** – first pattern want to match in string, must set.
- **pattern2** – second pattern, it will be used if pattern1 not match, optional.

Returns Match substring or None

`virttest.utils_misc.format_guest_disk(session, did, mountpoint, size, fstype, ostype)`

Create a partition on disk in guest and format and mount it.

Parameters

- **session** – session object to guest.
- **did** – disk ID in guest.
- **mountpoint** – mount point for the disk.
- **fstype** – filesystem type for the disk.
- **ostype** – guest os type ‘windows’ or ‘linux’.

Return Boolean disk usable or not.

`virttest.utils_misc.format_linux_disk(session, did, mountpoint, size, fstype='ext3')`

Create a partition on disk in linux guest and format and mount it.

Parameters

- **session** – session object to guest.
- **did** – disk serial, kname or wwn.
- **mountpoint** – mount point for the disk.
- **fstype** – filesystem type for the disk.
- **ostype** – guest os type 'windows' or 'linux'.

Return Boolean disk usable or not.

`virttest.utils_misc.format_str_for_message(sr)`

Format str so that it can be appended to a message. If str consists of one line, prefix it with a space. If str consists of multiple lines, prefix it with a newline.

Parameters **str** – string that will be formatted.

`virttest.utils_misc.format_windows_disk(session, did, mountpoint=None, size=None, fstype='ntfs')`

Create a partition on disk in windows guest and format it.

Parameters

- **session** – session object to guest.
- **did** – disk index which show in 'diskpart list disk'.
- **mountpoint** – mount point for the disk.
- **fstype** – filesystem type for the disk.
- **ostype** – guest os type 'windows' or 'linux'.

Return Boolean disk usable or not.

`virttest.utils_misc.generate_random_id()`

Return a random string suitable for use as a qemu id.

`virttest.utils_misc.generate_random_string(length, ignore_str='!\"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~', convert_str='')`

Return a random string using alphanumeric characters.

Parameters

- **length** – Length of the string that will be generated.
- **ignore_str** – Characters that will not include in generated string.
- **convert_str** – Characters that need to be escaped (prepend "").

Returns The generated random string.

`virttest.utils_misc.generate_tmp_file_name(file_name, ext=None, directory='/tmp/')`

Returns a temporary file name. The file is not created.

`virttest.utils_misc.get_archive_tarball_name(source_dir, tarball_name, compression)`

Get the name for a tarball file, based on source, name and compression

`virttest.utils_misc.get_cpu_flags(cpu_info='')`

Returns a list of the CPU flags

```
virttest.utils_misc.get_cpu_info (session=None)
```

Return information about the CPU architecture

Parameters `session` – session Object

Returns A dict of cpu information

```
virttest.utils_misc.get_cpu_status (cpu_num)
```

Get cpu status to check it's enable or disable

```
virttest.utils_misc.get_cpu_vendor (cpu_info='', verbose=True)
```

Returns the name of the CPU vendor

```
virttest.utils_misc.get_dev_major_minor (dev)
```

Get the major and minor numbers of the device @return: Tuple(major, minor) numbers of the device

```
virttest.utils_misc.get_dev_pts_max_id ()
```

Get the maxi ID of pseudoterminal interfaces for /dev/pts

:param None

```
virttest.utils_misc.get_free_disk (session, mount)
```

Get FreeSpace for given mount point.

Parm session shell Object.

Parm mount mount point(eg. C:, /mnt)

Return string freespace M-bytes

```
virttest.utils_misc.get_free_mem (session, os_type)
```

Get Free memory for given OS.

Parm session shell Object.

Parm os_type os type (eg. linux or windows)

Return string freespace M-bytes

```
virttest.utils_misc.get_full_pci_id (pci_id)
```

Get full PCI ID of pci_id.

Parameters `pci_id` – PCI ID of a device.

```
virttest.utils_misc.get_hash_from_file (hash_path, dvd_basename)
```

Get the a hash from a given DVD image from a hash file (Hash files are usually named MD5SUM or SHA1SUM and are located inside the download directories of the DVDs)

Parameters

- **hash_path** – Local path to a hash file.
- **cd_image** – Basename of a CD image

```
virttest.utils_misc.get_host_cpu_models ()
```

Get cpu model from host cpuinfo

```
virttest.utils_misc.get_image_info (image_file)
```

Get image information and put it into a dict. Image information like this:

```
*****
image: /path/vml_6.3.img
file format: raw
virtual size: 10G (10737418240 bytes)
disk size: 888M
....
```

```
image: /path/vm2_6.3.img
file format: raw
virtual size: 1.0M (1024000 bytes)
disk size: 196M
....
*****
```

And the image info dict will be like this

```
image_info_dict = {'format': 'raw',
                   'vsize' : '10737418240',
                   'dsize' : '931135488',
                   'csize' : '65536'}
```

`virttest.utils_misc.get_log_file_dir()`

get the base directory for log files created by `log_line()`.

`virttest.utils_misc.get_module_params(sys_path, module_name)`

Get the kvm module params :param sys_path: sysfs path for modules info :param module_name: module to check

`virttest.utils_misc.get_node_cpus(i=0)`

Get cpu ids of one node

Returns the cpu lists

Return type *list*

`virttest.utils_misc.get_open_fds(pid)`

`virttest.utils_misc.get_path(base_path, user_path)`

Translate a user specified path to a real path. If user_path is relative, append it to base_path. If user_path is absolute, return it as is.

Parameters

- **base_path** – The base path of relative user specified paths.
- **user_path** – The user specified path.

`virttest.utils_misc.get_pci_devices_in_group(str_flag='')`

Get PCI Devices. Classify pci devices according to its bus and slot, devices with same bus and slot will be put together. The format will be {'domain:bus:slot': 'device_function',...}

Parameters **str_flag** – the match string to filter devices.

`virttest.utils_misc.get_pci_group_by_id(pci_id, device_type='')`

Fit pci_id to a group list which has same domain:bus:slot.

Parameters

- **pci_id** – pci id of a device: domain:bus:slot.function or domain:bus:slot even bus:slot
- **device_type** – string which can stand device like 'Ethernet', 'Fibre'

`virttest.utils_misc.get_pci_vendor_device(pci_id)`

Get vendor and device number by pci id.

Returns a 'vendor device' list include all matched devices

`virttest.utils_misc.get_pid_cpu(pid)`

Get the process used cpus.

Parameters **pid** – process id

Returns A list include all cpus the process used

Return type *list*

`virttest.utils_misc.get_pid_from_file(program_name, pid_files_dir=None)`

Reads the pid from <program_name>.pid in the autotest directory.

:param program_name the name of the program :return: the pid if the file exists, None otherwise.

`virttest.utils_misc.get_pid_path(program_name, pid_files_dir=None)`

`virttest.utils_misc.get_qemu_best_cpu_model(params)`

Try to find out the best CPU model available for qemu.

This function can't be in qemu_vm, because it is used in env_process, where there's no vm object available yet, and env content is synchronized in multi host testing.

1. Get host CPU model
2. Verify if host CPU model is in the list of supported qemu cpu models
3. If so, return host CPU model
4. If not, return the default cpu model set in params, if none defined, return 'qemu64'.

`virttest.utils_misc.get_qemu_binary(params)`

Get the path to the qemu binary currently in use.

`virttest.utils_misc.get_qemu_cpu_models(qemu_binary)`

Get listing of CPU models supported by QEMU

Get list of CPU models by parsing the output of <qemu> -cpu '?'

`virttest.utils_misc.get_qemu_dst_binary(params)`

Get the path to the qemu dst binary currently in use.

`virttest.utils_misc.get_qemu_img_binary(params)`

Get the path to the qemu-img binary currently in use.

`virttest.utils_misc.get_qemu_io_binary(params)`

Get the path to the qemu-io binary currently in use.

`virttest.utils_misc.get_support_machine_type(qemu_binary='/usr/libexec/qemu-kvm')`

Get the machine type the host support, return a list of machine type

`virttest.utils_misc.get_test_entrypoint_func(name, module)`

Returns the test entry point function for a loaded module

Parameters

- **name** (*str*) – the name of the test. Usually supplied on a cartesian config file using the “type” key
- **module** (*module*) – a loaded python module for containing the code for the test named on name

Raises ValueError if module does not have a suitable function

Returns the test entry point function

Return type *func*

`virttest.utils_misc.get_thread_cpu(thread)`

Get the light weight process(thread) used cpus.

Parameters **thread** (*string*) – thread checked

Returns A list include all cpus the thread used

Return type *list*

```
virttest.utils_misc.get_vendor_from_pci_id(pci_id)
```

Check out the device vendor ID according to pci_id.

Parameters **pci_id** – PCI ID of a device.

```
virttest.utils_misc.get_virt_test_open_fds()
```

```
virttest.utils_misc.get_windows_drive_letters(session)
```

Get drive letters has been assigned

Parameters **session** – session object to guest

Return list letters has been assigned

```
virttest.utils_misc.get_winutils_vol(session, label='WIN_UTILS')
```

Return Volume ID of winutils CDROM ISO file should be create via command `mkisofs -V $label -o winutils.iso`.

Parameters

- **session** – session Object
- **label** – volume ID of WIN_UTILS.iso

Returns volume ID

```
virttest.utils_misc.install_disktest_on_vm(test, vm, src_dir, dst_dir)
```

Install stress to vm.

Parameters

- **vm** – virtual machine.
- **src_dir** – Source path.
- **dst_dir** – Installation path.

```
virttest.utils_misc.install_host_kernel(job, params)
```

Install a host kernel, given the appropriate params.

Parameters

- **job** – Job object.
- **params** – Dict with host kernel install params.

```
virttest.utils_misc.is_mounted(src, mount_point, fstype, perm=None, verbose=False,
                               fstype_mtab=None)
```

Check mount status from /etc/mtab

Parameters

- **src** (*string*) – mount source
- **mount_point** (*string*) – mount point
- **fstype** (*string*) – file system type
- **perm** (*string*) – mount permission
- **verbose** (*Boolean*) – if display mtab content
- **fstype_mtab** (*str*) – file system type in mtab could be different

Returns if the src is mounted as expect

Return type Boolean

`virttest.utils_misc.is_port_free(port, address)`

Return True if the given port is available for use.

Parameters `port` – Port number

`virttest.utils_misc.kill_process_by_pattern(pattern)`

Send SIGTERM signal to a process with matched pattern. :param pattern: normally only matched against the process name

`virttest.utils_misc.kill_process_tree(pid, sig=9)`

Signal a process and all of its children.

If the process does not exist – return.

Parameters

- **pid** – The pid of the process to signal.
- **sig** – The signal to send to the processes.

`virttest.utils_misc.kvm_flags_to_stresstests(flags)`

Covert [cpu flags] to [tests]

Parameters `cpuflags` – list of cpuflags

Returns Return tests like string.

`virttest.utils_misc.lock_file(filename, mode=2)`

`virttest.utils_misc.log_last_traceback(msg=None, log=<function error>)`

Writes last traceback into specified log.

Warning This function is being moved into `autotest` and your code should use `autotest.client.shared.base_utils` function instead.

Parameters

- **msg** – Override the default message. ["Original traceback"]
- **log** – Where to log the traceback [logging.error]

`virttest.utils_misc.log_line(filename, line)`

Write a line to a file.

Parameters

- **filename** – Path of file to write to, either absolute or relative to the dir set by `set_log_file_dir()`.
- **line** – Line to write.

`virttest.utils_misc.monotonic_time()`

Get monotonic time

`virttest.utils_misc.mount(src, mount_point, fstype, perm=None, verbose=False, fstype_mtab=None)`

Mount the src into mount_point of the host.

Src mount source

Mount_point mount point

Fstype file system type

Perm mount permission

Parameters `fstype_mtab` (*str*) – file system type in mtab could be different

`virttest.utils_misc.normalize_data_size` (*value_str*, *order_magnitude*=*'M'*, *factor*=*'1024'*)
Normalize a data size in one order of magnitude to another (MB to GB, for example).

Parameters

- **value_str** – a string include the data and unit
- **order_magnitude** – the magnitude order of result
- **factor** – the factor between two relative order of magnitude. Normally could be 1024 or 1000

`virttest.utils_misc.parallel` (*targets*)

Run multiple functions in parallel.

Parameters **targets** – A sequence of tuples or functions. If it's a sequence of tuples, each tuple will be interpreted as (target, args, kwargs) or (target, args) or (target,) depending on its length. If it's a sequence of functions, the functions will be called without arguments.

Returns A list of the values returned by the functions called.

`virttest.utils_misc.pid_exists` (*pid*)

Return True if a given PID exists.

Parameters **pid** – Process ID number.

`virttest.utils_misc.process_or_children_is_defunct` (*ppid*)

Verify if any processes from PPID is defunct.

Attempt to verify if parent process and any children from PPID is defunct (zombie) or not. :param ppid: The parent PID of the process to verify.

`virttest.utils_misc.program_is_alive` (*program_name*, *pid_files_dir*=*None*)

Checks if the process is alive and not in Zombie state.

:param program_name the name of the program :return: True if still alive, False otherwise

`virttest.utils_misc.qemu_has_option` (*option*, *qemu_path*=*'/usr/bin/qemu-kvm'*)

Helper function for command line option wrappers

Parameters

- **option** – Option need check.
- **qemu_path** – Path for qemu-kvm.

`virttest.utils_misc.rm_ker_cmd` (*kernel_cmdline*, *kernel_param*)

Remove a parameter from kernel command line content

Parameters

- **kernel_cmdline** (*string*) – Original kernel command line.
- **kernel_param** (*string*) – parameter want to change include the value.

Returns kernel command line

Return type *string*

`virttest.utils_misc.run_tests` (*parser*, *job*)

Runs the sequence of KVM tests based on the list of dictionaries generated by the configuration system, handling dependencies.

Parameters

- **parser** – Config parser object.
- **job** – Autotest job object.

Returns True, if all tests ran passed, False if any of them failed.

`virttest.utils_misc.safe_kill(pid, signal)`

Attempt to send a signal to a given process that may or may not exist.

Parameters **signal** – Signal number.

`virttest.utils_misc.selinux_enforcing()`

Deprecated function

Returns True if SELinux is in enforcing mode, False if permissive/disabled

Alias to `utils_selinux.is_enforcing()`

`virttest.utils_misc.set_cpu_status(cpu_num, enable=True)`

Set assigned cpu to be enable or disable

`virttest.utils_misc.set_log_file_dir(directory)`

Set the base directory for log files created by `log_line()`.

Parameters **dir** – Directory for log files.

`virttest.utils_misc.signal_program(program_name, sig=15, pid_files_dir=None)`

Sends a signal to the process listed in `<program_name>.pid`

:param `program_name` the name of the program :param `sig` signal to send

`virttest.utils_misc.string_to_bitlist(data)`

Transform from ASCII string to bit list.

Parameters **data** – String to be transformed

`virttest.utils_misc.strip_console_codes(output, custom_codes=None)`

Remove the Linux console escape and control sequences from the console output. Make the output readable and can be used for result check. Now only remove some basic console codes using during boot up.

Parameters

- **output** (*string*) – The output from Linux console
- **custom_codes** – The codes added to the console codes which is not covered in the default codes

Returns the string without any special codes

Return type *string*

`virttest.utils_misc.umount(src, mount_point, fstype, verbose=False, fstype_mtab=None)`

Unmount the `src` mounted in `mount_point`.

Src mount source

Mount_point mount point

Type file system type

Parameters **fstype_mtab** (*str*) – file system type in mtab could be different

`virttest.utils_misc.unbind_device_driver(pci_id)`

Unbind device current driver.

`virttest.utils_misc.unique(llist)`

Return a list of the elements in list, but without duplicates.

Parameters `list` – List with values.

Returns List with non duplicate elements.

`virttest.utils_misc.unlock_file(lockfile)`

`virttest.utils_misc.valued_option_dict(options, split_pattern, start_count=0, dict_split=None)`

Divide the valued options into key and value

Parameters

- **options** – the valued options get from cfg
- **split_pattern** – patten used to split options
- **dict_split** – patten used to split sub options and insert into dict
- **start_count** – the start_count to insert option_dict

Returns dict include option and its value

`virttest.utils_misc.verify_host_dmesg(dmesg_log_file=None, trace_re=None)`

Find host call trace in dmesg log.

Parameters

- **dmesg_log_file** – The file used to save host dmesg. If None, will save host dmesg to logging.debug.
- **trace_re** – re string used to filter call trace.

`virttest.utils_misc.verify_running_as_root()`

Verifies whether we're running under UID 0 (root).

Raise `error.TestNAError`

`virttest.utils_misc.wait_for(func, timeout, first=0.0, step=1.0, text=None)`

Wait until func() evaluates to True.

If func() evaluates to True before timeout expires, return the value of func(). Otherwise return None.

Parameters

- **timeout** – Timeout in seconds
- **first** – Time to sleep before first attempt
- **steps** – Time to sleep between attempts in seconds
- **text** – Text to print while waiting, for debug purposes

`virttest.utils_misc.write_pid(program_name, pid_files_dir=None)`

Try to drop <program_name>.pid in the main autotest directory.

Args: `program_name`: prefix for file name

`virttest.utils_misc.yum_install(pkg_list, session=None, timeout=300)`

Try to install packages on system

Parameters `pkg_list` – list of packages

Session session Object

Returns True if all packages installed, False if any error

virttest.utils_misc_unittest module

```
class virttest.utils_misc_unittest.FakeCmd(cmd)
    Bases: object

    get_stdout(cmd)

class virttest.utils_misc_unittest.TestNumaNode(methodName='runTest')
    Bases: unittest.case.TestCase

    setUp()
    tearDown()
    test_bitlist_to_string()
    test_free_cpu()
    test_get_node_cpus()
    test_pin_cpu()
    test_string_to_bitlist()

class virttest.utils_misc_unittest.TestUtilsMisc(methodName='runTest')
    Bases: unittest.case.TestCase

    test_cpu_vendor_amd()
    test_cpu_vendor_intel()
    test_get_archive_tarball_name()
    test_get_archive_tarball_name_absolute()
    test_get_archive_tarball_name_from_dir()
    test_git_repo_param_helper()
    test_normalize_data_size()
    test_vendor_unknown()

virttest.utils_misc_unittest.utils_run(cmd)
```

virttest.utils_net module

```
exception virttest.utils_net.BRAddIfError(ifname, brname, details)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.BRDelIfError(ifname, brname, details)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.BRIpError(brname)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.BRNotExistError(brname, details)
    Bases: virttest.utils_net.NetError

class virttest.utils_net.Bridge
    Bases: object

    add_bridge(brname)
        Add a bridge in host
```

add_port (*brname, ifname*)

Add a device to bridge

Parameters

- **ifname** – Name of TAP device
- **brname** – Name of the bridge

del_bridge (*brname*)

Delete a bridge in host

del_port (*brname, ifname*)

Remove a TAP device from bridge

Parameters

- **ifname** – Name of TAP device
- **brname** – Name of the bridge

get_stp_status (*brname*)

get STP status

get_structure ()

Get bridge list.

list_br ()

list_iface ()

Return all interfaces used by bridge.

port_to_br (*port_name*)

Return bridge which contain port.

Parameters **port_name** – Name of port.

Returns Bridge name or None if there is no bridge which contain port.

class `virttest.utils_net.DbNet` (*params, vm_name, db_filename, db_key*)

Bases: `virttest.utils_net.VMNet`

Networking information from database

Database specification- database values are python string-formatted lists of dictionaries

db_entry (*db_key=None*)

Returns a python list of dictionaries from locked DB string-format entry

lock_db ()

mac_index ()

Generator of mac addresses found in database

save_to_db (*db_key=None*)

Writes string representation out to database

unlock_db ()

update_db ()

exception `virttest.utils_net.DbNoLockError` (**args*)

Bases: `virttest.utils_net.NetError`

exception `virttest.utils_net.DelLinkError` (*ifname, details=None*)

Bases: `virttest.utils_net.NetError`

```
exception virttest.utils_net.HwAddrGetError (ifname)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.HwAddrSetError (ifname, mac)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.HwOperstarteGetError (ifname, details=None)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.IPAddrGetError (mac_addr, details=None)
    Bases: virttest.utils_net.NetError

class virttest.utils_net.IPAddress (ip_str='', info='')
    Bases: object

    Class to manipulate IPv4 or IPv6 address.

    canonicalize (ip_str)
        Parse an IP string for listen to IPAddress content.

    listening_on (port, max_retry=30)
        Check whether a port is used for listening.

class virttest.utils_net.IPv6Manager (*args, **dargs)
    Bases: virttest.propcan.PropCanBase

    Setup and cleanup IPv6 environment.

    auto_recover

    static check_connectivity (client_ifname, server_ipv6, count=5)
        Check IPv6 network connectivity :param client_ifname: client network interface name :param server_ipv6:
        server IPv6 address ::param count: sending packets counts, default is 5

    check_ipv6_connectivity

    cleanup ()
        Cleanup IPv6 network environment.

    client

    client_ifname

    client_ipv6_addr

    close_session ()
        If the session exists then close it.

    flush_ip6tables ()
        Refresh IPv6 firewall rules

    get_addr_list (runner=None)
        Get IPv6 address list from local and remote host.

    get_session ()
        Make sure the session is alive and available

    port

    prompt

    runner

    server_ifname

    server_ip
```

```
server_ipv6_addr
server_pwd
server_user
session
setup ()
    Setup IPv6 network environment.
exception virttest.utils_net.IfChangeAddrError (ifname, ipaddr, details)
    Bases: virttest.utils_net.NetError
exception virttest.utils_net.IfChangeBrError (ifname, old_brname, new_brname, details)
    Bases: virttest.utils_net.NetError
exception virttest.utils_net.IfNotInBridgeError (ifname, details)
    Bases: virttest.utils_net.NetError
class virttest.utils_net.Interface (name)
    Bases: object
    Class representing a Linux network device.
    dellink ()
        Delete the interface. Equivalent to 'ip link delete NAME'.
    down (*args, **argkw)
    get_index (*args, **argkw)
    get_ip (*args, **argkw)
    get_mac (*args, **argkw)
    get_netmask (*args, **argkw)
    get_stats (*args, **argkw)
    is_brport ()
        Check Whether this Interface is a bridge port_to_br
    is_up (*args, **argkw)
    set_ip (*args, **argkw)
    set_mac (*args, **argkw)
    set_netmask (*args, **argkw)
    up (*args, **argkw)
class virttest.utils_net.LibvirtIface (*args, **dargs)
    Bases: virttest.utils_net.VirtIface
    Networking information specific to libvirt
class virttest.utils_net.Macvtap (tapname=None)
    Bases: virttest.utils_net.Interface
    class of macvtap, base Interface
    create (device, mode='vepa')
        Create a macvtap device, only when the device does not exist.
    Parameters
```

- **device** – Macvtap device to be created.
- **mode** – Creation mode.

delete()

get_device()

get_tapname()

ip_link_ctl(*params*, *ignore_status=False*)

open()

exception `virttest.utils_net.MacvtapCreationError` (*ifname*, *base_interface*, *details=None*)

Bases: `virttest.utils_net.NetError`

exception `virttest.utils_net.MacvtapGetBaseInterfaceError` (*ifname=None*, *details=None*)

Bases: `virttest.utils_net.NetError`

exception `virttest.utils_net.NetError` (**args*)

Bases: `exceptions.Exception`

exception `virttest.utils_net.OpenflowSwitchError` (*brname*)

Bases: `virttest.utils_net.NetError`

class `virttest.utils_net.ParamsNet` (*params*, *vm_name*)

Bases: `virttest.utils_net.VMNet`

Networking information from Params

Params contents specification- `vms = <vm names...> nics = <nic names...> nics_<vm name> = <nic names...> # attr: mac, ip, model, nettype, netdst, etc. <attr> = value <attr>_<nic name> = value`

mac_index()

Generator over mac addresses found in params

reset_ip(*index_or_name*)

Reset to ip from params if defined and valid, or undefine.

reset_mac(*index_or_name*)

Reset to mac from params if defined and valid, or undefine.

class `virttest.utils_net.QemuIface` (**args*, ***dargs*)

Bases: `virttest.utils_net.VirtIface`

Networking information specific to Qemu

device_id

ifname

netdev_extra_params

netdev_id

nic_extra_params

queues

romfile

tapfd_ids

tapfds

```
tftp
vectors
vhostfds
vlan

exception virttest.utils_net.TAPBringDownError (ifname)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.TAPBringUpError (ifname)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.TAPCreationError (ifname, details=None)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.TAPModuleError (devname, action='open', details=None)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.TAPNotExistError (ifname)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.VMIPv6AddressError (error_info)
    Bases: virttest.utils_net.NetError

exception virttest.utils_net.VMIPv6NeighNotFound (ipv6_address)
    Bases: virttest.utils_net.NetError

class virttest.utils_net.VMNet (container_class=<class 'virttest.utils_net.VirtIface'>,
                                face_list=[])
    Bases: list
    Collection of networking information.

    DISCARD_WARNINGS = 10

    append (value)

    mac_list ()
        Return a list of all mac addresses used by defined interfaces

    nic_lookup (prop_name, prop_value)
        Return the first index with prop_name key matching prop_value or None

    nic_name_index (name)
        Return the index number for name, or raise KeyError

    nic_name_list ()
        Obtain list of nic names from lookup of contents 'nic_name' key.

    process_mac (value)
        Strips 'mac' key from value if it's not valid

    subclass_pre_init (params, vm_name)
        Subclasses must establish style before calling VMNet. __init__()

exception virttest.utils_net.VMNetError (*args)
    Bases: virttest.utils_net.NetError

class virttest.utils_net.VMNetStyle
    Bases: dict
    Make decisions about needed info from vm_type and driver_type params.

    VMNet_Style_Map = {'default': {'default': {'mac_prefix': '9a', 'container_class': <class 'virttest.utils_net.QemuIface'>}}
```



```
classmethod get_driver_type_map (vm_type_map, driver_type)
classmethod get_style (vm_type, driver_type)
classmethod get_vm_type_map (vm_type)
class virttest.utils_net.VirtIface (*args, **dargs)
    Bases: virttest.propan.PropCan, object
    Networking information for single guest interface and host connection.
    LASTBYTE = 200
    classmethod complete_mac_address (mac)
        Append randomly generated byte strings to make mac complete
        Parameters mac – String or list of mac bytes (possibly incomplete)
        Raise TypeError if mac is not a string or a list
    g_nic_name
    classmethod generate_bytes ()
        Return next byte from ring
    classmethod int_list_to_mac_str (mac_bytes)
        Return string formatting of int mac_bytes
    ip
    mac
    classmethod mac_is_valid (mac)
    classmethod mac_str_to_int_list (mac)
        Convert list of string bytes to int list
    classmethod name_is_valid (nic_name)
        Corner-case prevention where nic_name is not a sane string value
    netdst
    nettype
    nic_model
    nic_name
class virttest.utils_net.VirtNet (params, vm_name, db_key, db_filename='tmp/address_pool')
    Bases: virttest.utils_net.DbNet, virttest.utils_net.ParamsNet
    Persistent collection of VM's networking information.
    free_mac_address (nic_index_or_name)
        Remove the mac value from nic_index_or_name and cache unless static
        Parameters nic_index_or_name – index number or name of NIC
    generate_ifname (nic_index_or_name)
        Return and set network interface name
    generate_mac_address (nic_index_or_name, attempts=1024)
        Set & return valid mac address for nic_index_or_name or raise NetError
        Parameters nic_index_or_name – index number or name of NIC
        Returns MAC address string
```

Raise NetError if mac generation failed

get_mac_address (*nic_index_or_name*)

Return a MAC address for *nic_index_or_name*

Parameters *nic_index_or_name* – index number or name of NIC

Returns MAC address string.

mac_index ()

Generator for all allocated mac addresses (requires db lock)

set_mac_address (*nic_index_or_name*, *mac*)

Set a MAC address to value specified

Parameters *nic_index_or_name* – index number or name of NIC

Raise NetError if mac already assigned

exception `virttest.utils_net.VlanError` (*ifname*, *details*)

Bases: `virttest.utils_net.NetError`

`virttest.utils_net.add_ovs_bridge` (**args*, ***kwargs*)

`virttest.utils_net.add_to_bridge` (**args*, ***kwargs*)

`virttest.utils_net.bring_down_ifname` (*ifname*)

Bring down an interface

Parameters *ifname* – Name of the interface

`virttest.utils_net.bring_up_ifname` (*ifname*)

Bring up an interface

Parameters *ifname* – Name of the interface

`virttest.utils_net.change_iface_bridge` (**args*, ***kwargs*)

`virttest.utils_net.check_add_dnsmasq_to_br` (*br_name*, *tmpdir*)

Add dnsmasq for bridge. dnsmasq could be added only if bridge has assigned ip address.

Parameters

- **bridge_name** – Name of bridge.
- **bridge_ip** – Bridge ip.
- **tmpdir** – Tmp dir for save pid file and ip range file.

Returns When new dnsmasq is started name of pidfile otherwise return None because system dnsmasq is already started on bridge.

`virttest.utils_net.check_listening_port_by_service` (*service*, *port*, *listen_addr='0.0.0.0'*, *runner=None*)

Check TCP/IP listening by service

`virttest.utils_net.check_listening_port_remote_by_service` (*server_ip*, *server_user*, *server_pwd*, *service*, *port*, *listen_addr*)

Check remote TCP/IP listening by service

`virttest.utils_net.clean_tmp_files` ()

Remove the base address pool filename.

```
virttest.utils_net.create_and_open_macvtap (ifname, mode='vepa', queues=1,
                                             base_if=None, mac_addr=None)
```

Create a new macvtap device, open it, and return the fds

Parameters

- **ifname** – macvtap interface name
- **mode** – macvtap type mode (“vepa, bridge,...)
- **queues** – Queue number
- **base_if** – physical interface to create macvtap
- **mac_addr** – macvtap mac address

```
virttest.utils_net.create_macvtap (ifname, mode='vepa', base_if=None, mac_addr=None)
```

Create Macvtap device, return a object of Macvtap

Parameters

- **ifname** – macvtap interface name
- **mode** – macvtap type mode (“vepa, bridge,...)
- **base_if** – physical interface to create macvtap
- **mac_addr** – macvtap mac address

```
virttest.utils_net.del_from_bridge (*args, **kwargs)
```

```
virttest.utils_net.del_net_if_ip (if_name, ip_addr, runner=None)
```

Delete network device ip addresses.

Parameters

- **if_name** – Name of interface.
- **ip_addr** – Interface ip addr in format “ip_address/mask”.

Raise IfChangeAddrError.

```
virttest.utils_net.del_ovs_bridge (*args, **kwargs)
```

```
virttest.utils_net.disable_windows_guest_network (session, connection_id,
                                                    timeout=240)
```

```
virttest.utils_net.enable_windows_guest_network (session, connection_id, timeout=240)
```

```
virttest.utils_net.find_bridge_manager (*args, **kwargs)
```

```
virttest.utils_net.find_current_bridge (*args, **kwargs)
```

```
virttest.utils_net.find_dnsmasq_listen_addresses ()
```

Search all dnsmasq listen addresses.

Parameters

- **bridge_name** – Name of bridge.
- **bridge_ip** – Bridge ip.

Returns List of ip where dnsmasq is listening.

```
virttest.utils_net.generate_mac_address_simple ()
```

```
virttest.utils_net.get_all_ips ()
```

Get all IPv4 and IPv6 addresses from all interfaces.

`virttest.utils_net.get_correspond_ip(remote_ip)`

Get local ip address which is used to contact remote ip.

Parameters `remote_ip` – Remote ip

Returns Local correspond IP.

`virttest.utils_net.get_guest_ip_addr(session, mac_addr, os_type='linux', ip_version='ipv4', linklocal=False)`

Get guest ip addresses by serial session

Parameters

- **session** – serial session
- **mac_addr** – nic mac address of the nic that you want get
- **os_type** – guest os type, windows or linux
- **ip_version** – guest ip version, ipv4 or ipv6
- **linklocal** – Whether ip address is local or remote

Returns ip addresses of network interface.

`virttest.utils_net.get_host_default_gateway()`

Get the Default Gateway in host. :return: a string of the host's default gateway. :rtype: string

`virttest.utils_net.get_host_iface()`

List the nic interface in host. :return: a list of the interfaces in host :rtype: list

`virttest.utils_net.get_host_ip_address(params)`

returns ip address of host specified in host_ip_addr parameter If provided otherwise ip address on interface specified in netdst parameter is returned :param params

`virttest.utils_net.get_ip_address_by_interface(ifname)`

returns ip address by interface :param ifname - interface name :raise NetError - When failed to fetch IP address (ioctl raised IOError.).

Retrieves interface address from socket fd through ioctl call and transforms it into string from 32-bit packed binary by using socket.inet_ntoa().

`virttest.utils_net.get_linux_ifname(session, mac_address='')`

Get the interface name through the mac address.

Parameters

- **session** – session to the virtual machine
- **mac_address** – the macaddress of nic

:raise error. `TestError` in case it was not possible to determine the interface name.

`virttest.utils_net.get_macvtap_base_iface(base_interface=None)`

Get physical interface to create macvtap, if you assigned base interface is valid(not belong to any bridge and is up), will use it; else use the first physical interface, which is not a brport and up.

`virttest.utils_net.get_neigh_attach_interface(neigh_address)`

Get the interface which can reach the neigh_address

`virttest.utils_net.get_neigh_mac(neigh_address)`

Get neighbour mac by his address

`virttest.utils_net.get_neighbours_info(neigh_address='', interface_name=None)`

Get the neighbours information

`virttest.utils_net.get_net_if(runner=None, state=None)`

Parameters

- **runner** – command runner.
- **div_phy_virt** – if set true, will return a tuple division real physical interface and virtual interface

Returns List of network interfaces.

`virttest.utils_net.get_net_if_addrs(if_name, runner=None)`

Get network device ip addresses. ioctl not used because it's not compatible with ipv6 address.

Parameters **if_name** – Name of interface.

Returns List ip addresses of network interface.

`virttest.utils_net.get_net_if_addrs_win(session, mac_addr)`

Try to get windows guest nic address by serial session

Parameters

- **session** – serial session
- **mac_addr** – guest nic mac address

Returns List ip addresses of network interface.

`virttest.utils_net.get_net_if_and_addrs(runner=None)`

Returns Dict of interfaces and their addresses {"ifname": addrs}.

`virttest.utils_net.get_net_if_operstate(ifname, runner=None)`

Get linux host/guest network device operstate.

Parameters **if_name** – Name of the interface.

Raise HwOperstateGetError.

`virttest.utils_net.get_sorted_net_if()`

Get all network interfaces, but sort them among physical and virtual if.

Returns Tuple (physical interfaces, virtual interfaces)

`virttest.utils_net.get_windows_nic_attribute(session, key, value, target, timeout=240, global_switch='nic')`

Get the windows nic attribute using wmic. All the support key you can using wmic to have a check.

Parameters

- **session** – session to the virtual machine
- **key** – the key supported by wmic
- **value** – the value of the key
- **target** – which nic attribute you want to get.

`virttest.utils_net.if_nametoindex(ifname)`

Map an interface name into its corresponding index. Returns 0 on error, as 0 is not a valid index

Parameters **ifname** – interface name

`virttest.utils_net.if_set_macaddress(ifname, mac)`

Set the mac address for an interface

Parameters

- **ifname** – Name of the interface
- **mac** – Mac address

`virttest.utils_net.ipv6_from_mac_addr(mac_addr)`

Returns Ipv6 address for communication in link range.

`virttest.utils_net.is_virtual_network_dev(dev_name)`

Parameters **dev_name** – Device name.

Returns True if dev_name is in virtual/net dir, else false.

`virttest.utils_net.local_runner(cmd, timeout=None)`

`virttest.utils_net.local_runner_status(cmd, timeout=None)`

`virttest.utils_net.neigh_reachable(neigh_address, attach_if=None)`

Check the neighbour is reachable

`virttest.utils_net.open_macvtap(macvtap_object, queues=1)`

Open a macvtap device and returns its file descriptors which are used by `fds=<fd1:fd2:...>` parameter of `qemu`

For single queue, only returns one file descriptor, it's used by `fd=<fd>` legacy parameter of `qemu`

If you not have a switch support vepa in you env, run this type case you need at least two nic on you host [just workaround]

Parameters

- **macvtap_object** – macvtap object
- **queues** – Queue number

`virttest.utils_net.open_tap(devname, ifname, queues=1, vnet_hdr=True)`

Open a tap device and returns its file descriptors which are used by `fds=<fd1:fd2:...>` parameter of `qemu`

For single queue, only returns one file descriptor, it's used by `fd=<fd>` legacy parameter of `qemu`

Parameters

- **devname** – TUN device path
- **ifname** – TAP interface name
- **queues** – Queue number
- **vnet_hdr** – Whether enable the vnet header

`virttest.utils_net.openflow_manager(*args, **kwargs)`

`virttest.utils_net.ovs_br_exists(*args, **kwargs)`

`virttest.utils_net.parse_arp()`

Read `/proc/net/arp`, return a mapping of MAC to IP

Returns dict mapping MAC to IP

`virttest.utils_net.refresh_neigh_table(interface_name=None, neigh_address='ff02::1')`

Refresh host neighbours table, if interface_name is assigned only refresh neighbours of this interface, else refresh the all the neighbours.

`virttest.utils_net.restart_guest_network(session, mac_addr=None, os_type='linux', ip_version='ipv4', timeout=240)`

Restart guest network by serial session

Parameters

- **session** – serial session
- **mac_addr** – nic mac address of the nic that you want restart
- **os_type** – guest os type, windows or linux
- **ip_version** – guest ip version, ipv4 or ipv6
- **timeout** – timeout value for command.

```
virttest.utils_net.restart_windows_guest_network(session, connection_id, timeout=240,
                                                mode='netsh')
```

Restart guest's network via serial console. mode "netsh" can not works in winxp system

Parameters

- **session** – session to virtual machine
- **connection_id** – windows nic connectionid, it means connection name, you Can get connection id string via wmic

```
virttest.utils_net.restart_windows_guest_network_by_devcon(session, netdev_id,
                                                         timeout=240)
```

```
virttest.utils_net.restart_windows_guest_network_by_key(session, key, value, timeout=240, mode='netsh')
```

Restart the guest network by nic Attribute like connectionid, interfaceindex, "netsh" can not work in winxp system. using devcon mode must download devcon.exe and put it under c: :param session: session to virtual machine :param key: the key supported by wmic nic :param value: the value of the key :param timeout: timeout :param mode: command mode netsh or devcon

```
virttest.utils_net.set_guest_network_status_by_devcon(session, status, netdev_id, timeout=240)
```

using devcon to enable/disable the network device. using it must download the devcon.exe, and put it under c:

```
virttest.utils_net.set_net_if_ip(if_name, ip_addr, runner=None)
```

Set network device ip addresses. ioctl not used because there is incompatibility with ipv6.

Parameters

- **if_name** – Name of interface.
- **ip_addr** – Interface ip addr in format "ip_address/mask".

Raise IfChangeAddrError.

```
virttest.utils_net.set_win_guest_nic_status(session, connection_id, status, timeout=240)
```

Set windows guest nic ENABLED/DISABLED

:param session : session to virtual machine :param connection_id : windows guest nic netconnectionid :param status : set nic ENABLED/DISABLED

```
virttest.utils_net.update_mac_ip_address(vm, params, timeout=None)
```

Get mac and ip address from guest then update the mac pool and address cache

Parameters

- **vm** – VM object
- **params** – Dictionary with the test parameters.

```
virttest.utils_net.verify_ip_address_ownership(ip, macs, timeout=60.0)
```

Use arping and the ARP cache to make sure a given IP address belongs to one of the given MAC addresses.

Parameters

- **ip** – An IP address.

- **macs** – A list or tuple of MAC addresses.

Returns True if ip is assigned to a MAC address in macs.

`virttest.utils_net.vnet_hdr_probe(tapfd)`

Check if the IFF_VNET_HDR is support by tun.

Parameters `tapfd` – the file descriptor of /dev/net/tun

`virttest.utils_net.vnet_mq_probe(tapfd)`

Check if the IFF_MULTI_QUEUE is support by tun.

Parameters `tapfd` – the file descriptor of /dev/net/tun

`virttest.utils_net.warp_init_del(func)`

virttest.utils_net_unittest module

class `virttest.utils_net_unittest.FakeVm(vm_name, params)`

Bases: `object`

get_params()

is_alive()

class `virttest.utils_net_unittest.TestBridge(methodName='runTest')`

Bases: `unittest.case.TestCase`

class `FakeCmd(*args, **kwargs)`

Bases: `object`

get_stdout()

iter = 0

`TestBridge.setUp()`

`TestBridge.tearDown()`

`TestBridge.test_getstructure()`

class `virttest.utils_net_unittest.TestLibvirtIface(methodName='runTest')`

Bases: `virttest.utils_net_unittest.TestVirtIface`

setUp()

class `virttest.utils_net_unittest.TestQemuIface(methodName='runTest')`

Bases: `virttest.utils_net_unittest.TestVirtIface`

setUp()

class `virttest.utils_net_unittest.TestVirtIface(methodName='runTest')`

Bases: `unittest.case.TestCase`

class `VirtIface(*args, **kwargs)`

Bases: `virttest.propcan.PropCan, object`

Networking information for single guest interface and host connection.

LASTBYTE = 200

classmethod `complete_mac_address(mac)`

Append randomly generated byte strings to make mac complete

Parameters `mac` – String or list of mac bytes (possibly incomplete)

Raise `TypeError` if mac is not a string or a list


```
g_nic_name
classmethod generate_bytes ()
    Return next byte from ring
classmethod int_list_to_mac_str (mac_bytes)
    Return string formatting of int mac_bytes

ip
mac
classmethod mac_is_valid (mac)
classmethod mac_str_to_int_list (mac)
    Convert list of string bytes to int list
classmethod name_is_valid (nic_name)
    Corner-case prevention where nic_name is not a sane string value

netdst
nettype
nic_model
nic_name

TestVirtIface.loop_assert (virtiface, test_keys, what_func)
TestVirtIface.setUp ()
TestVirtIface.test_apendex_set ()
    Verify container ignores unknown key names
TestVirtIface.test_full_set ()
TestVirtIface.test_half_set ()
TestVirtIface.test_mac_completer ()

class virttest.utils_net_unittest.TestVmNet (methodName='runTest')
    Bases: unittest.case.TestCase
    setUp ()
    test_VirtIface_container ()
    test_string_container ()

class virttest.utils_net_unittest.TestVmNetStyle (methodName='runTest')
    Bases: unittest.case.TestCase
    get_style (vm_type, driver_type)
    setUp ()
    test_default_default ()
    test_libvirt ()

class virttest.utils_net_unittest.TestVmNetSubclasses (methodName='runTest')
    Bases: unittest.case.TestCase
    counter = 0
    db_filename = '/dev/shm/UnitTest_AddressPool'
    db_item_count = 0
```

```
fakevm_generator()
mac_prefix = '01:02:03:04:05:'
nettests_cartesian = '\n variants:\n - onevm:\n vms=vm1\n - twovms:\n vms=vm1 vm2\n - threevms:\n vms=vm1 vm2 vm3\n'
print_and_inc()
setUp()
    Runs before every test
test_01_Params()
    Load Cartesian combinatorial result verifies against all styles of VM.
    Note: There are some cases where the key should NOT be set, in this case an exception is caught
        prior to verifying
test_02_db()
    Load Cartesian combinatorial result from params into database
test_03_db()
    Load from database created in test_02_db, verify data against params
test_04_VirtNet()
    Populate database with max - 1 mac addresses
test_05_VirtNet()
    Load max - 1 entries from db, overriding params.
    DEPENDS ON test_04_VirtNet running first
test_06_VirtNet()
    Generate last possibly mac and verify value.
    DEPENDS ON test_05_VirtNet running first
test_07_VirtNet()
    Release mac from beginning, middle, and end, re-generate + verify value
test_08_ifname()
test_99_ifname()
test_cmp_Virtnet()
zero_counter (increment=100)
```

virttest.utils_netperf module

```
class virttest.utils_netperf.Netperf(address, netperf_path, md5sum='', netperf_source='',
                                     client='ssh', port='22', username='root', password='redhat',
                                     compile_option='-enable-demo=yes', install=True)
    Bases: object
    is_target_running(target)
    stop(target)
class virttest.utils_netperf.NetperfClient(address, netperf_path, md5sum='', netperf_source='',
                                           client='ssh', port='22', username='root', password='redhat',
                                           compile_option='', install=True)
    Bases: virttest.utils_netperf.Netperf
```

bg_start (*server_address*, *test_option*='', *session_num*=1, *cmd_prefix*='', *package_sizes*='')
 Run netperf background, for stress test do not have output

Parameters

- **server_address** – Remote netserver address
- **netperf_path** – netperf test option (global/test option)
- **timeout** – Netperf test timeout(-l)
- **cmd_prefix** – Prefix in netperf command
- **package_sizes** – Package sizes test in netperf command.

is_netperf_running ()

start (*server_address*, *test_option*='', *timeout*=1200, *cmd_prefix*='', *package_sizes*='')
 Run netperf test

Parameters

- **server_address** – Remote netserver address
- **netperf_path** – Netperf test option (global/test option)
- **timeout** – Netperf test timeout(-l)
- **cmd_prefix** – Prefix in netperf command
- **package_sizes** – Package sizes test in netperf command.

Returns return test result

stop ()

exception `virttest.utils_netperf.NetperfError`

Bases: `exceptions.Exception`

class `virttest.utils_netperf.NetperfPackage` (*address*, *netperf_path*, *md5sum*='', *netperf_source*='', *client*='ssh', *port*='22', *username*='root', *password*='123456')

Bases: `virttest.remote.RemotePackage`

env_cleanup (*clean_all*=True)

install (*install*, *compile_option*)

pack_compile (*compile_option*='')

pull_file (*netperf_source*=None)

Copy file from remote to local.

exception `virttest.utils_netperf.NetperfPackageError` (*error_info*)

Bases: `virttest.utils_netperf.NetperfError`

class `virttest.utils_netperf.NetperfServer` (*address*, *netperf_path*, *md5sum*='', *netperf_source*='', *client*='ssh', *port*='22', *username*='root', *password*='redhat', *compile_option*='-enable-demo=yes', *install*=True)

Bases: `virttest.utils_netperf.Netperf`

is_server_running ()

start (*restart*=False)

Start/Restart netserver

Parameters **restart** – if restart=True, will restart the netserver

stop()

exception `virttest.utils_netperf.NetperfTestError` (*error_info*)

Bases: `virttest.utils_netperf.NetperfError`

exception `virttest.utils_netperf.NetserverError` (*error_info*)

Bases: `virttest.utils_netperf.NetperfError`

virttest.utils_params module

exception `virttest.utils_params.ParamNotFound`

Bases: `autotest.client.shared.error.TestNAError`

class `virttest.utils_params.Params` (*args, **kwargs)

Bases: `UserDict.IterableUserDict`

A dict-like object passed to every test.

lock = <thread.lock object>

object_counts (*count_key*, *base_name*)

This is a generator method: to give it the name of a count key and a base_name, and it returns an iterator over all the values from params

object_params (*obj_name*)

Return a dict-like object containing the parameters of an individual object.

This method behaves as follows: the suffix ‘_’ + obj_name is removed from all key names that have it. Other key names are left unchanged. The values of keys with the suffix overwrite the values of their suffixless versions.

Parameters **obj_name** – The name of the object (objects are listed by the objects() method).

objects (*key*)

Return the names of objects defined using a given key.

Parameters **key** – The name of the key whose value lists the objects (e.g. ‘nics’).

virttest.utils_params_unittest module

class `virttest.utils_params_unittest.TestParams` (*methodName*=‘runTest’)

Bases: `unittest.case.TestCase`

setUp()

testGetItem()

testGetItemMissing()

testObjects()

testObjectsParams()

virttest.utils_sasl module

tools to manage sasl.

```
class virttest.utils_sasl.SASL(*args, **dargs)
    Bases: virttest.propcan.PropCanBase

    Base class of a connection between server and client.

    auto_recover

    cleanup(remote=True)
        Clear created sasl users

    client

    close_session()
        If session exists then close it

    get_session()
        Make sure the session is alive and available

    linesep

    list_users(remote=True, sasldb_path='/etc/libvirt/passwd.db')
        List users in sasldb

    port

    prompt

    sasl_pwd_cmd

    sasl_user_cmd

    sasl_user_pwd

    server_ip

    server_pwd

    server_user

    session

    setup(remote=True)
        Create sasl users with password

class virttest.utils_sasl.VirshSessionSASL(params)
    Bases: virttest.virsh.VirshSession

    A wrap class for virsh session which used SASL infrastructure.
```

virttest.utils_selinux module

selinux test utility functions.

```
exception virttest.utils_selinux.RestoreconError
    Bases: virttest.utils_selinux.SelinuxError

exception virttest.utils_selinux.SeCmdError(cmd, detail)
    Bases: virttest.utils_selinux.SelinuxError

    Error in executing cmd.

exception virttest.utils_selinux.SelinuxError
    Bases: exceptions.Exception

    Error selinux utility functions.
```

exception `virttest.utils_selinux.SemanageError`

Bases: `virttest.utils_selinux.SelinuxError`

Error when semanage binary is not found

`virttest.utils_selinux.apply_defcon(pathname, dirdesc=False)`

Apply default contexts to pathname, possibly descending into sub-dirs also.

Parameters

- **pathname** – Absolute path to file, directory, or symlink
- **dirdesc** – True to descend into sub-directories

Returns List of changes applied tuple(pathname, from context, to context)

`virttest.utils_selinux.del_defcon(context_type, pathregex)`

Remove the default local SELinux policy type for a file/path

Parameters **context** – The selinux context (only type is used)

Pramm pathregex Pathname regex e.g. `r"/foo/bar/baz(/.*)?"`

Raises

- **SelinuxError** – if semanage command not found
- **SeCmdError** – if semanage exits non-zero

`virttest.utils_selinux.diff_defcon(pathname, dirdesc=False)`

Return a list of tuple(pathname, from, to) for current & default contexts

Parameters

- **pathname** – Absolute path to file, directory, or symlink
- **dirdesc** – True to descend into sub-directories

Returns List of tuple(pathname, from context, to context)

`virttest.utils_selinux.find_defcon(defcon, pathname)`

Returns the context type of first match to pathname or None

`virttest.utils_selinux.find_defcon_idx(defcon, pathname)`

Returns the index into defcon where pathname matches or None

`virttest.utils_selinux.find_pathregex(defcon, pathname)`

Returns the regular expression in defcon matching pathname

`virttest.utils_selinux.get_context_from_str(context)`

Get the context in a context.

Parameters **context** – SELinux context string

Raises **SelinuxError** – if there is no context in context.

`virttest.utils_selinux.get_context_of_file(filename)`

Get the context of file.

Raises **SeCmdError** – if execute 'getfattr' failed.

`virttest.utils_selinux.get_context_of_process(pid)`

Get context of process.

`virttest.utils_selinux.get_defcon(local=False)`

Return list of dictionaries containing SELinux default file context types

Parameters **local** – Only return locally modified default contexts

Returns list of dictionaries of default context attributes

`virttest.utils_selinux.get_status()`

Get the status of selinux.

Returns string of status in STATUS_LIST.

Raises

- **SeCmdError** – if execute ‘getenforce’ failed.
- **SelinuxError** – if ‘getenforce’ command exit 0, but the output is not expected.

`virttest.utils_selinux.get_type_from_context(context)`

Return just the type component of a full context string

Parameters `context` – SELinux context string

Returns Type component of SELinux context string

`virttest.utils_selinux.is_disabled()`

Return True if the selinux is disabled.

`virttest.utils_selinux.is_enforcing()`

Return true if the selinux is enforcing.

`virttest.utils_selinux.is_not_disabled()`

Return True if the selinux is not disabled.

`virttest.utils_selinux.is_permissive()`

Return true if the selinux is permissive.

`virttest.utils_selinux.set_context_of_file(filename, context)`

Set context of file.

Raises

- **SeCmdError** – if failed to execute chcon.
- **SelinuxError** – if command chcon execute normally, but the context of file is not setted to context.

`virttest.utils_selinux.set_defcon(context_type, pathregex, context_range=None)`

Set the default context of a file/path in local SELinux policy

Parameters

- **context_type** – The selinux context (only type is used)
- **pathregex** – Pathname regex e.g. `r”/foo/bar/baz(/.)*?”`
- **context_range** – MLS/MCS Security Range e.g. `s0:c87,c520`

Raises

- **SelinuxError** – if semanage command not found
- **SeCmdError** – if semanage exits non-zero

`virttest.utils_selinux.set_status(status)`

Set status of selinux.

Parameters `status` – status want to set selinux.

Raises

- **SelinuxError** – status is not supported.

- **SelinuxError** – need to reboot host.
- **SeCmdError** – execute setenforce failed.
- **SelinuxError** – cmd setenforce exit normally, but status of selinux is not set to expected.

`virttest.utils_selinux.transmogrify_sub_dirs` (*pathregex*)

Append `'(/.*)?'` regex to end of *pathregex* to optionally match all subdirs

`virttest.utils_selinux.transmogrify_usr_local` (*pathregex*)

Replace `usr/local/something` with `usr/(local/)?something`

`virttest.utils_selinux.verify_defcon` (*pathname*, *dirdesc=False*, *readonly=True*,
forcedesc=False)

Verify contexts of *paths*pec (and/or below, if *dirdesc*) match default

Parameters

- **pathname** – Absolute path to file, directory, or symlink
- **dirdesc** – True to descend into sub-directories
- **readonly** – True to passive check and don't change any file labels
- **forcedesc** – True to force a replacement of the entire context

Returns True if all components match default contexts

Note By default DOES NOT follow symlinks

virttest.utils_spice module

Common spice test utility functions.

exception `virttest.utils_spice.RVConnectError`

Bases: `exceptions.Exception`

Exception raised in case that remote-viewer fails to connect

`virttest.utils_spice.clear_interface` (*vm*, *login_timeout=360*, *timeout=5*)

Clears user interface of a *vm* without reboot

Parameters *vm* – VM where cleaning is required

`virttest.utils_spice.clear_interface_linux` (*vm*, *login_timeout*, *timeout*)

Clears user interface of a *vm* without reboot

Parameters *vm* – VM where cleaning is required

`virttest.utils_spice.deploy_epel_repo` (*guest_session*, *params*)

Deploy epel repository to RHEL VM If It's RHEL6 or 5.

:param *guest_session* - ssh session to guest VM :param *params*

`virttest.utils_spice.gen_rv_file` (*params*, *guest_vm*, *host_subj=None*, *cacert=None*)

Generates *vv* file for remote-viewer

Parameters

- **params** – all parameters of the test
- **guest_vm** – object of a guest VM
- **host_subj** – subject of the host
- **cacert** – location of certificate of host

`virttest.utils_spice.get_vdagent_status` (*vm_session*, *test_timeout*)

Return the status of vdagent :param *vm_session*: ssh session of the VM :param *test_timeout*: timeout time for the cmd

`virttest.utils_spice.install_rv_win` (*client*, *host_path*, *client_path*='C:\virt-viewer.msi')

Install remote-viewer on a windows client

Parameters

- **client** – VM object
- **host_path** – Location of installer on host
- **client_path** – Location of installer after copying

`virttest.utils_spice.install_usbclerk_win` (*client*, *host_path*, *client_path*='C:\usbclerk.msi')

Install remote-viewer on a windows client

Parameters

- **client** – VM object
- **host_path** – Location of installer on host
- **client_path** – Location of installer after copying

`virttest.utils_spice.kill_app` (*vm_name*, *app_name*, *params*, *env*)

Kill selected app on selected VM

:params *vm_name* - VM name in parameters :params *app_name* - name of application

`virttest.utils_spice.restart_vdagent` (*guest_session*, *test_timeout*)

Sending commands to restart the spice-vdagentd service

Parameters

- **guest_session** – ssh session of the VM
- **test_timeout** – timeout time for the cmds

`virttest.utils_spice.start_vdagent` (*guest_session*, *test_timeout*)

Sending commands to start the spice-vdagentd service

Parameters

- **guest_session** – ssh session of the VM
- **test_timeout** – timeout time for the cmds

`virttest.utils_spice.stop_vdagent` (*guest_session*, *test_timeout*)

Sending commands to stop the spice-vdagentd service

Parameters

- **guest_session** – ssh session of the VM
- **test_timeout** – timeout time for the cmds

`virttest.utils_spice.verify_established` (*client_vm*, *host*, *port*, *rv_binary*, *tls_port*=None, *secure_channels*=None)

Parses netstat output for established connection on host:port :param *client_session* - *vm.wait_for_login()* :param *host* - host ip addr :param *port* - port for client to connect :param *rv_binary* - remote-viewer binary

`virttest.utils_spice.verify_vdagent` (*guest_session*, *test_timeout*)

Verifying vdagent is installed on a VM

Parameters

- **guest_session** – ssh session of the VM
- **test_timeout** – timeout time for the cmds

`virttest.utils_spice.verify_virtio (guest_session, test_timeout)`
Verify Virtio linux driver is properly loaded.

Parameters

- **guest_session** – ssh session of the VM
- **test_timeout** – timeout time for the cmds

`virttest.utils_spice.wait_timeout (timeout=10)`
time.sleep(timeout) + logging.debug(timeout)
:param timeout=10

virttest.utils_v2v module

Virt-v2v test utility functions.

copyright 2008-2012 Red Hat Inc.

class `virttest.utils_v2v.LinuxVMCheck (test, params, env)`
Bases: `virttest.utils_v2v.VMCheck`

This class handles all basic linux VM check operations.

get_grub_device (*dev_map='/boot/grub2/device.map'*)
Check whether vd[a-z] device is in device map.

get_vm_kernel ()
Get vm kernel info.

get_vm_modprobe_conf ()
Get /etc/modprobe.conf content.

get_vm_modules ()
Get vm modules list.

get_vm_os_info ()
Get vm os info.

get_vm_os_vendor ()
Get vm os vendor.

get_vm_parted ()
Get vm parted info.

get_vm_pci_list ()
Get vm pci list.

get_vm_rc_local ()
Get vm /etc/rc.local output.

get_vm_tty ()
Get vm tty config.

get_vm_video ()
Get vm video config.

has_vmware_tools ()
Check vmware tools.

is_disk_virtio (*disk='/dev/vda'*)

Check whether disk is virtio.

is_net_virtio ()

Check whether vm's interface is virtio

class `virttest.utils_v2v.Target` (*target, uri*)

Bases: `object`

This class is used for generating command options.

get_cmd_options (*params*)

Target dispatcher.

class `virttest.utils_v2v.Uri` (*hypervisor*)

Bases: `object`

This class is used for generating uri.

get_uri (*hostname, vpx_dc=None, esx_ip=None*)

Uri dispatcher.

Parameters **hostname** – String with host name.

class `virttest.utils_v2v.VMCheck` (*test, params, env*)

Bases: `object`

This is VM check class dispatcher.

cleanup ()

Cleanup VM and remove all of storage files about guest

create_session (*timeout=480*)

storage_cleanup ()

Cleanup storage pool and volume

class `virttest.utils_v2v.WindowsVMCheck` (*test, params, env*)

Bases: `virttest.utils_v2v.VMCheck`

This class handles all basic Windows VM check operations.

click_install_driver ()

Move mouse and click button to install driver for new device(Ethernet controller)

click_left_button ()

Click left button of VM mouse.

click_tab_enter ()

Send TAB and ENTER to VM.

copy_windows_file ()

Copy a windows file

delete_windows_file ()

Delete a windows file

get_driver_info (*signed=True*)

Get windows signed driver info.

get_network_restart ()

Get windows network restart.

get_screenshot ()

Do virsh screenshot of the vm and fetch the image if the VM in remote host.

get_viostor_info()

Get viostor info.

get_windows_event_info()

Get windows event log info about WSH.

move_mouse(*coordinate*)

Move VM mouse.

reboot_windows()

Reboot Windows immediately

send_win32_key(*keycode*)

Send key to Windows VM

wait_for_match(*images, similar_degree=0.98, timeout=300*)

Compare VM screenshot with given images, if any image in the list matched, then return the image index, or return -1.

virttest.utils_v2v.import_vm_to_ovirt(*params, address_cache, timeout=600*)

Import VM from export domain to oVirt Data Center

virttest.utils_v2v.v2v_cmd(*params*)

Append 'virt-v2v' and execute it.

Parameters **params** – A dictionary includes all of required parameters such as 'target', 'hypervisor' and 'hostname', etc.

Returns A CmdResult object

virttest.utils_virtio_port module

class virttest.utils_virtio_port.**VirtioPortTest**(*test, env, params*)

Bases: **object**

static cleanup(**args, **kwargs*)

Cleanup function.

Parameters

- **vm** – VM whose ports should be cleaned
- **guest_worker** – guest_worker which should be cleaned/exited

get_virtio_ports(**args, **kwargs*)

Returns separated virtconsoles and virtserialports

Parameters **vm** – VM object

Returns tuple (all virtconsoles, all virtserialports)

get_vm_with_ports(**args, **kwargs*)

Checks whether existing 'main_vm' fits the requirements, modifies it if needed and returns the VM object.

Parameters

- **no_console** – Number of desired virtconsoles.
- **no_serialport** – Number of desired virtserialports.
- **spread** – Spread consoles across multiple virtio-serial-pcis.
- **quiet** – Notify user about VM recreation.
- **strict** – Whether no_consoles have to match or just exceed.

Returns vm object matching the requirements.

get_vm_with_single_port (*args, **kwargs)

Wrapper which returns vm, guest_worker and virtio_ports with at least one port of the type specified by fction parameter.

Parameters **port_type** – type of the desired virtio port.

Returns tuple (vm object with at least 1 port of the port_type, initialized GuestWorker of the vm, list of virtio_ports of the port_type type)

get_vm_with_worker (*args, **kwargs)

Checks whether existing ‘main_vm’ fits the requirements, modifies it if needed and returns the VM object and guest_worker.

Parameters

- **no_console** – Number of desired virtconsoles.
- **no_serialport** – Number of desired virtserialports.
- **spread** – Spread consoles across multiple virtio-serial-pcis.
- **quiet** – Notify user about VM recreation.
- **strict** – Whether no_consoles have to match or just exceed.

Returns tuple (vm object matching the requirements, initialized GuestWorker of the vm)

virttest.version module

Based on work from Douglas Creager <dcreager@dcreager.net>

Gets the current version number. If possible, this is the output of “git describe”, modified to conform to the versioning scheme that setuputils uses. If “git describe” returns an error (most likely because we’re in an unpacked copy of a release tarball, rather than in a git working copy), then we fall back on reading the contents of the RELEASE-VERSION file.

```
virttest.version.get_git_version(abbrev=4)
```

```
virttest.version.get_version(abbrev=4)
```

```
virttest.version.get_top_commit()
```

```
virttest.version.get_current_branch()
```

```
virttest.version.get_pretty_version_info()
```

virttest.versionable_class module

```
class virttest.versionable_class.Manager(name, wrapper=None)
```

Bases: `object`

factory (_class, *args, **kwargs)

Create new class with right version of subclasses.

Goes through class structure and search subclasses with right version.

Parameters

- **_class** (*class.*) – Class which should be prepared.
- **args** – Params for `_is_right_ver` function.

Params kargs Params for `_is_right_ver` function.

getcls (*cls*, *orig_class*)

Return class correspond class and original class.

Parameters

- **cls** (*class*) – class for which should be found derived alternative.
- **orig_class** (*class*) – Original class

Returns Derived alternative class

Return type class

class `virttest.versionable_class.ModuleWrapper` (*wrapped*)

Bases: `object`

Wrapper around module.

Necessary for pickling of dynamic class.

class `virttest.versionable_class.VersionableClass`

Bases: `object`

Class used for marking of mutable class.

`virttest.versionable_class.factory` (*orig_cls*, **args*, ***kargs*)

Create class with specific version.

Parameters

- **orig_class** – Class from which should be derived good version.
- **args** – list of parameters for `_ir_right_ver`

Params kargs dict of named parameters for `_ir_right_ver`

Returns params specific class.

Return type class

`virttest.versionable_class.isclass` (*obj*)

Parameters **obj** – Object for inspection if obj is class.

Returns true if the object is a class.

virttest.versionable_class_unittest module

Wrapper around module.

Necessary for pickling of dynamic class.

class `virttest.versionable_class_unittest.AA`

Bases: `virttest.versionable_class_unittest.Sys_Container`,
`virttest.versionable_class_unittest.BB`, `virttest.versionable_class_unittest.System_Cont`

class `virttest.versionable_class_unittest.BB`

Bases: `virttest.versionable_class_unittest.VM_container`

func1 ()

func2 ()

test_class_bb = None

```
class virttest.versionable_class_unittest.Q(*args, **kwargs)
    Bases: object

class virttest.versionable_class_unittest.Q1(*args, **kwargs)
    Bases: virttest.versionable_class_unittest.Q

class virttest.versionable_class_unittest.Q_Container
    Bases: virttest.versionable_class.VersionableClass

class virttest.versionable_class_unittest.Sys(*args, **kwargs)
    Bases: virttest.versionable_class_unittest.Q_Container

class virttest.versionable_class_unittest.Sys1(*args, **kwargs)
    Bases: virttest.versionable_class_unittest.Sys

class virttest.versionable_class_unittest.Sys_Container
    Bases: virttest.versionable_class.VersionableClass

class virttest.versionable_class_unittest.System(*args, **kwargs)
    Bases: object

class virttest.versionable_class_unittest.System1(*args, **kwargs)
    Bases: virttest.versionable_class_unittest.System

class virttest.versionable_class_unittest.System_Container
    Bases: virttest.versionable_class.VersionableClass

class virttest.versionable_class_unittest.TestVersionableClass(methodName='runTest')
    Bases: unittest.case.TestCase

    setUp()

    tearDown()

    test_complicated_multiple_create_params()

    test_complicated_versioning()

    test_pickleing()
        Test pickling for example save vm env.

    test_sharing_data_in_same_version()

    test_simple_create_by_params_v0()

    test_simple_create_by_params_v1()

    test_simple_versioning()

class virttest.versionable_class_unittest.VM(*args, **kwargs)
    Bases: object

    func1()

    func3()

    test_class_vm1 = None

class virttest.versionable_class_unittest.VM1(*args, **kwargs)
    Bases: virttest.versionable_class_unittest.VM

    func1()

    func2()

    func3()
```

```
class virttest.versionable_class_unittest.VM_container
    Bases: virttest.versionable_class.VersionableClass
virttest.versionable_class_unittest.qemu_verison()
virttest.versionable_class_unittest.system_version()
```

virttest.video_maker module

Video Maker transforms screenshots taken during a test into a HTML 5 compatible video, so that one can watch the screen activity of the whole test from inside your own browser.

This relies on generally available multimedia libraries, frameworks and tools.

```
class virttest.video_maker.GstPythonVideoMaker(verbose=False)
    Bases: object
    Makes a movie out of screendump images using gstreamer-python
    CONTAINER_ENCODER_MAPPING = {'ogg': 'theora', 'webm': 'vp8'}
    CONTAINER_MAPPING = {'ogg': 'oggmux', 'webm': 'webmmux'}
    ENCODER_MAPPING = {'theora': 'theoraenc', 'vp8': 'vp8enc'}
    get_container_name()
        Gets the video container available that is the best based on preference
    get_element(name)
        Makes and returns an element from the gst factory interface
    get_encoder_name()
        Gets the video encoder available that is the best based on preference
    get_most_common_image_size(input_dir)
        Find the most common image size
    has_element(kind)
        Returns True if a gstreamer element is available
    normalize_images(input_dir)
        GStreamer requires all images to be the same size, so we do it here
    start(input_dir, output_file)
        Process the input files and output the video file
virttest.video_maker.video_maker(input_dir, output_file)
    Instantiates and runs a video maker
```

virttest.virsh module

Utility classes and functions to handle connection to a libvirt host system

The entire contents of callables in this module (minus the names defined in NOCLOSE below), will become methods of the Virsh and VirshPersistent classes. A Closure class is used to wrap the module functions, lambda does not properly store instance state in this implementation.

Because none of the methods have a 'self' parameter defined, the classes are defined to be dict-like, and get passed in to the methods as a the special ****dargs** parameter. All virsh module functions **_MUST_** include a special ****dargs** (variable keyword arguments) to accept non-default keyword arguments.

The standard set of keyword arguments to all functions/modules is declared in the `VirshBase` class. Only the 'virsh_exec' key is guaranteed to always be present, the remainder may or may not be provided. Therefore, virsh functions/methods should use the `dict.get()` method to retrieve with a default for non-existent keys.

copyright 2012 Red Hat Inc.

```
class virttest.virsh.Virsh(*args, **dargs)
```

Bases: `virttest.virsh.VirshBase`

Execute libvirt operations, using a new virsh shell each time.

```
class virttest.virsh.VirshBase(*args, **dargs)
```

Bases: `virttest.propcan.PropCanBase`

Base Class storing libvirt Connection & state to a host

debug

get_uri()

Accessor method for 'uri' property that must exist

ignore_status

readonly

uri

virsh_exec

```
class virttest.virsh.VirshClosure(reference_function, dict_like_instance)
```

Bases: `object`

Callable with weak ref. to override `**dargs` when calling `reference_function`

```
class virttest.virsh.VirshConnectBack(*args, **dargs)
```

Bases: `virttest.virsh.VirshPersistent`

Persistent virsh session connected back from a remote host

static kosher_args (*remote_ip*, *uri*)

Convenience static method to help validate argument sanity before use

Parameters

- **remote_ip** – ip/hostname of remote libvirt helper-system
- **uri** – fully qualified libvirt uri of local system, from remote.

Returns True/False if checks pass or not

new_session()

Open new remote session, closing any existing

remote_ip

```
class virttest.virsh.VirshPersistent(*args, **dargs)
```

Bases: `virttest.virsh.Virsh`

Execute libvirt operations using persistent virsh session.

COUNTERS = {}

close_session()

If a persistent session exists, close it down.

counter_decrease()

Method to decrease the counter to self.a_id in COUNTERS. If the counter is less than 1, it means there is no more VirshSession instance referring to the session. So close this session, and return True. Else, decrease the counter in COUNTERS and return False.

counter_increase()

Method to increase the counter to self.a_id in COUNTERS.

new_session()

Open new session, closing any existing

readonly

remote_ip

remote_pwd

remote_user

session_id

set_uri(uri)

Accessor method for 'uri' property, create new session on change

ssh_remote_auth

unprivileged_user

uri

```
class virttest.virsh.VirshSession(virsh_exec=None, uri=None, a_id=None,
                                  prompt='virsh\s*[\#\>\s*]', remote_ip=None, re-
                                  mote_user=None, remote_pwd=None, ssh_remote_auth=False,
                                  readonly=False, unprivileged_user=None, auto_close=False,
                                  check_libvirt=True)
```

Bases: *virttest.aexpect.ShellSession*

A virsh shell session, used with Virsh instances.

ERROR_REGEX_LIST = ['error:\s*.*\$', '.*failed.*']

cmd_result(cmd, ignore_status=False, debug=False, timeout=60)

Mimic utils.run()

cmd_status_output(cmd, timeout=60, internal_timeout=None, print_func=None)

Send a virsh command and return its exit status and output.

Parameters

- **cmd** – virsh command to send (must not contain newline characters)
- **timeout** – The duration (in seconds) to wait for the prompt to return
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)

Returns A tuple (status, output) where status is the exit status and output is the output of cmd

Raises

- **ShellTimeoutError** – Raised if timeout expires
- **ShellProcessTerminatedError** – Raised if the shell process terminates while waiting for output

- **ShellStatusError** – Raised if the exit status cannot be obtained
- **ShellError** – Raised if an unknown error occurs

read_until_output_matches (*patterns*, *filter_func*=<function <lambda>>, *timeout*=60, *internal_timeout*=None, *print_func*=None, *match_func*=None)

Read from child using read_nonblocking until a pattern matches.

Read using read_nonblocking until a match is found using match_patterns, or until timeout expires. Before attempting to search for a match, the data is filtered using the filter_func function provided.

Parameters

- **patterns** – List of strings (regular expression patterns)
- **filter_func** – Function to apply to the data read from the child before attempting to match it against the patterns (should take and return a string)
- **timeout** – The duration (in seconds) to wait until a match is found
- **internal_timeout** – The timeout to pass to read_nonblocking
- **print_func** – A function to be used to print the data being read (should take a string parameter)
- **match_func** – Function to compare the output and patterns.

Returns Tuple containing the match index and the data read so far

Raises

- **ExpectTimeoutError** – Raised if timeout expires
- **ExpectProcessTerminatedError** – Raised if the child process terminates while waiting for output
- **ExpectError** – Raised if an unknown error occurs

virttest.virsh.attach_device (*domainarg*=None, *filearg*=None, *domain_opt*=None, *file_opt*=None, *flagstr*=None, ***dargs*)

Attach a device using full parameter/argument set.

Parameters

- **domainarg** – Domain name (first pos. parameter)
- **filearg** – File name (second pos. parameter)
- **domain_opt** – Option to –domain parameter
- **file_opt** – Option to –file parameter
- **flagstr** – string of “–force, –persistent, etc.”
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

virttest.virsh.attach_disk (*name*, *source*, *target*, *extra*='', ***dargs*)

Attach a disk to VM.

Parameters

- **name** – name of guest
- **source** – source of disk device
- **target** – target of disk device

- **extra** – additional arguments to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.attach_interface(name, option='', **dargs)`

Attach a NIC to VM.

Parameters

- **name** – name of guest
- **option** – options to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.autostart(name, options, **dargs)`

Autostart a domain

Returns cmdresult object.

`virttest.virsh.blkdeviotune(name, device=None, options=None, total_bytes_sec=None, read_bytes_sec=None, write_bytes_sec=None, total_iops_sec=None, read_iops_sec=None, write_iops_sec=None, **dargs)`

Set or get a domain's blkio parameters :param name: name of domain :param options: options may be live, config and current :param dargs: standardized virsh function API keywords :return: CmdResult instance

`virttest.virsh.blkiotune(name, weight=None, device_weights=None, options=None, **dargs)`

Set or get a domain's blkio parameters :param name: name of domain :param options: options may be live, config and current :param dargs: standardized virsh function API keywords :return: CmdResult instance

`virttest.virsh.blockcommit(name, path, options='', **dargs)`

Start a block commit operation.

Parameters

- **name** – name of domain
- **options** – options of blockcommit
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.blockcopy(name, path, dest, options='', **dargs)`

Start a block copy operation.

Parameters

- **name** – name of domain.
- **path** – fully-qualified path or target of disk.
- **dest** – path of the copy to create.
- **options** – options of blockcopy.
- **dargs** – standardized virsh function API keywords.

Returns CmdResult instance.

`virttest.virsh.blockjob(name, path, options='', **dargs)`

Manage active block operations.

Parameters

- **name** – name of domain.
- **path** – fully-qualified path or target of disk.
- **options** – options of blockjob.
- **dargs** – standardized virsh function API keywords.

Returns CmdResult instance.

`virttest.virsh.blockpull (name, path, options='', **dargs)`
Start a block pull operation.

Parameters

- **name** – name of domain
- **options** – options of blockpull
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.blockresize (name, path, size, **dargs)`
Resize block device of domain.

Parameters

- **name** – name of domain
- **path** – path of block device
- **dargs** – standardized virsh function API keywords

Size new size of the block device

Returns CmdResult instance

`virttest.virsh.canonical_uri (option='', **dargs)`
Return the hypervisor canonical URI.

Parameters

- **option** – additional option string to pass
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.capabilities (option='', to_file=None, **dargs)`
Return output from virsh capabilities command

Parameters

- **option** – additional options (takes none)
- **dargs** – standardized virsh function API keywords

`virttest.virsh.cd (dir_path, options='', **dargs)`
Run cd command in virsh interactive session.

Parameters

- **dir_path** – dir path string
- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.change_media` (*name, device, options, **dargs*)
Change media of CD or floppy drive.

Parameters

- **name** – VM’s name.
- **path** – Fully-qualified path or target of disk device
- **options** – command change_media options.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.click_button` (*name, left_button=True, **dargs*)
Click left/right button of VM mouse.

Parameters

- **name** – domain name
- **left_button** – Click left or right button

`virttest.virsh.command` (*cmd, **dargs*)
Interface to cmd function as ‘cmd’ symbol is polluted.

Parameters

- **cmd** – Command line to append to virsh command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

Raise CmdError if non-zero exit status and ignore_status=False

`virttest.virsh.connect` (*connect_uri='', options='', **dargs*)
Run a connect command to the uri.

Parameters

- **connect_uri** – target uri connect to.
- **options** – options to pass to connect command

Returns CmdResult object.

`virttest.virsh.cpu_baseline` (*xml_file, **dargs*)
Compute baseline CPU for a set of given CPUs.

Parameters

- **xml_file** – file containing an XML CPU description.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.cpu_compare` (*xml_file, **dargs*)
Compare host CPU with a CPU described by an XML file

Parameters

- **xml_file** – file containing an XML CPU description.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.cpu_models (arch, options='', **dargs)`
Get the CPU models for an arch.

Parameters

- **arch** – Architecture
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.cpu_stats (name, options, **dargs)`
Display per-CPU and total statistics about domain's CPUs

Parameters

- **name** – name of domain
- **options** – options of cpu_stats
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.create (xmlfile, options='', **dargs)`
Create guest from xml

Parameters

- **xmlfile** – domain xml file
- **options** – --paused

Returns CmdResult object

`virttest.virsh.define (xml_path, **dargs)`
Return cmd result of domain define.

Parameters

- **xml_path** – XML file path
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.desc (name, options, desc_str, **dargs)`
Show or modify description or title of a domain.

Parameters

- **name** – name of domain.
- **options** – options for desc command.
- **desc_str** – new desc message.
- **dargs** – standardized virsh function API keywords.

Returns CmdResult object.

`virttest.virsh.destroy (name, options='', **dargs)`
True on successful domain destruction

Parameters

- **name** – VM name
- **options** – options for virsh destroy
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.detach_device (domainarg=None, filearg=None, domain_opt=None,  
                             file_opt=None, flagstr=None, **dargs)
```

Detach a device using full parameter/argument set.

Parameters

- **domainarg** – Domain name (first pos. parameter)
- **filearg** – File name (second pos. parameter)
- **domain_opt** – Option to –domain parameter
- **file_opt** – Option to –file parameter
- **flagstr** – string of “–force, –persistent, etc.”
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

```
virttest.virsh.detach_disk (name, target, extra='', **dargs)
```

Detach a disk from VM.

Parameters

- **name** – name of guest
- **target** – target of disk device
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.detach_interface (name, option='', **dargs)
```

Detach a NIC to VM.

Parameters

- **name** – name of guest
- **option** – options to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.dom_list (options='', **dargs)
```

Return the list of domains.

Parameters **options** – options to pass to list command

Returns CmdResult object

```
virttest.virsh.domain_exists (name, **dargs)
```

Return True if a domain exists.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns True operation was successful

`virttest.virsh.domblkerror` (*name*, ***dargs*)
Show errors on block devices

Parameters *name* – name of domain

Returns CmdResult object

`virttest.virsh.domblkinfo` (*name*, *device*, ***dargs*)
Get block device size info for a domain.

Parameters

- **name** – VM's name or id,uuid.
- **device** – device of VM.
- **dargs** – standardized virsh function API keywords.

Returns CmdResult object.

`virttest.virsh.domblklist` (*name*, *options=None*, ***dargs*)
Get domain devices.

Parameters

- **name** – name of domain
- **options** – options of domblklist.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domblkstat` (*name*, *device*, *option*, ***dargs*)
Store state of VM into named file.

Parameters

- **name** – VM's name.
- **device** – VM's device.
- **option** – command domblkstat option.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domcapabilities` (*virttype=None*, *emulatorbin=None*, *arch=None*, *machine=None*,
options='', ***dargs*)
Capabilities of emulator with respect to host and libvirt

Parameters

- **virttype** – Virtualization type (/domain/@type)
- **emulatorbin** – Path to emulator binary (/domain/devices/emulator)
- **arch** – Domain architecture (/domain/os/type/@arch)
- **machine** – machine type (/domain/os/type/@machine)
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domcontrol (name, options='', **dargs)`

Return domain control interface state.

Parameters

- **name** – name of domain
- **options** – extra options

Returns CmdResult object

`virttest.virsh.domdisplay (name, options='', **dargs)`

Get domain display connection URI

Parameters

- **name** – name of domain
- **options** – options of domdisplay

Returns CmdResult object

`virttest.virsh.domfsfreeze (name, mountpoint=None, options='', **dargs)`

Freeze domain's mounted filesystems

Parameters

- **name** – name of domain
- **mountpoint** – specific mountpoints to be frozen
- **options** – extra options to domfsfreeze cmd.

Returns CmdResult object

`virttest.virsh.domfsthaw (name, mountpoint=None, options='', **dargs)`

Thaw domain's mounted filesystems

Parameters

- **name** – name of domain
- **mountpoint** – specific mountpoints to be thawed
- **options** – extra options to domfsfreeze cmd.

Returns CmdResult object

`virttest.virsh.domfstrim (name, minimum=None, mountpoint=None, options='', **dargs)`

Do fstrim on domain's mounted filesystems

Parameters

- **name** – name of domain
- **options** – options maybe `–minimum <number>`, `–mountpoint <string>`

Returns CmdResult object

`virttest.virsh.domid (name_or_uuid, **dargs)`

Return VM's ID.

Parameters

- **name_or_uuid** – VM name or uuid
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domif_getlink (name, interface, options=None, **dargs)`

Get network interface stats for a running domain.

Parameters

- **name** – Name of domain
- **interface** – interface device
- **options** – command options.
- **dargs** – standardized virsh function API keywords

Returns domif state

`virttest.virsh.domif_setlink (name, interface, state, options=None, **dargs)`

Set network interface stats for a running domain.

Parameters

- **name** – Name of domain
- **interface** – interface device
- **state** – new state of the device up or down
- **options** – command options.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.domiflist (name, options='', extra='', **dargs)`

Get the domain network devices

Parameters

- **name** – name of domain
- **options** – options of domiflist
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domifstat (name, interface, **dargs)`

Get network interface stats for a running domain.

Parameters

- **name** – Name of domain
- **interface** – interface device

Returns CmdResult object

`virttest.virsh.domiftune (name, interface, options=None, inbound=None, outbound=None, **dargs)`

Set/get parameters of a virtual interface.

Parameters

- **name** – name of domain.
- **interface** – interface device (MAC Address).
- **inbound** – control domain's incoming traffics.
- **outbound** – control domain's outgoing traffics.

- **options** – options may be live, config and current.
- **dargs** – standardized virsh function API keywords.

Returns CmdResult instance.

`virttest.virsh.dominfo(name, **dargs)`

Return the VM information.

Parameters

- **name** – VM's name or id,uuid.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domjobabort(name, **dargs)`

Aborts the currently running domain job.

Parameters

- **name** – VM's name, id or uuid.
- **dargs** – standardized virsh function API keywords

Returns result from command

`virttest.virsh.domjobinfo(name, **dargs)`

Get domain job information.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.dommemstat(name, extra='', **dargs)`

Store state of VM into named file.

Parameters

- **name** – VM name
- **extra** – extra options to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domname(dom_id_or_uuid, **dargs)`

Convert a domain id or UUID to domain name

Parameters

- **dom_id_or_uuid** – a domain id or UUID.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.dompmsuspend(name, target, duration=0, **dargs)`

Suspends a running domain using guest OS's power management.

Parameters

- **name** – VM name

- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.dompmwakeup(name, **dargs)`

Wakeup a domain that was previously suspended by power management.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.domstate(name, extra='', **dargs)`

Return the state about a running domain.

Parameters

- **name** – VM name
- **extra** – command options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.domstats(domains='', options='', **dargs)`

Get statistics about one or multiple domains

Parameters

- **domains** – List of domains
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domtime(name, now=False, pretty=False, sync=False, time=None, options='', **dargs)`

Get/Set domain's time

Parameters

- **name** – name of domain
- **now** – set to the time of the host running virsh
- **pretty** – print domain's time in human readable form
- **sync** – instead of setting given time, synchronize from domain's RTC
- **time** – integer time to set

Returns CmdResult object

`virttest.virsh.domuuid(name_or_id, **dargs)`

Return the Converted domain name or id to the domain UUID.

Parameters

- **name_or_id** – VM name or id
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.domxml_from_native (info_format, native_file, options=None, **dargs)`

Convert native guest configuration format to domain XML format.

:param info_format:The command's options. For exmple:qemu-argv. :param native_file:Native information file. :param options:extra param. :param dargs: standardized virsh function API keywords. :return: result from command

`virttest.virsh.domxml_to_native (info_format, xml_file, options, **dargs)`

Convert domain XML config to a native guest configuration format.

:param info_format:The command's options. For exmple:qemu-argv. :param xml_file:XML config file. :param options:extra param. :param dargs: standardized virsh function API keywords :return: result from command

`virttest.virsh.driver (**dargs)`

Return the driver by asking libvirt

Parameters **dargs** – standardized virsh function API keywords

Returns VM driver name

`virttest.virsh.dump (name, path, option='', **dargs)`

Dump the core of a domain to a file for analysis.

Parameters

- **name** – VM name
- **path** – absolute path to state file
- **option** – command's option.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.dumpxml (name, extra='', to_file='', **dargs)`

Return the domain information as an XML dump.

Parameters

- **name** – VM name
- **to_file** – optional file to write XML output to
- **dargs** – standardized virsh function API keywords

Returns CmdResult object.

`virttest.virsh.echo (echo_str, options='', **dargs)`

Run echo command in virsh session.

Parameters

- **echo_str** – the echo string
- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.edit (options, **dargs)`

Edit the XML configuration for a domain.

Parameters

- **options** – virsh edit options string.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.emulatorpin(name, cpulist=None, options=None, **dargs)`

Control or query domain emulator affinity :param name: name of domain :param cpulist: a list of physical CPU numbers :param options: options may be live, config and current :param dargs: standardized virsh function API keywords :return: CmdResult instance

`virttest.virsh.event(domain=None, event=None, event_timeout=None, options='', **dargs)`

List event types, or wait for domain events to occur

Parameters

- **domain** – Domain name, id or UUID
- **event** – Event type name
- **event_timeout** – Timeout seconds
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.exit(**dargs)`

Run exit command in virsh session.

Parameters **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.find_storage_pool_sources(source_type, srcSpec, **dargs)`

Find potential storage pool sources

Parameters

- **source_type** – type of storage pool sources to find
- **srcSpec** – file of source xml to query for pools
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.find_storage_pool_sources_as(source_type, options='', **dargs)`

Find potential storage pool sources

Parameters

- **source_type** – type of storage pool sources to find
- **options** – cmd options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

`virttest.virsh.freecell(cellno=None, options='', **dargs)`

Prints the available amount of memory on the machine or within a NUMA cell.

Parameters

- **cellno** – number of cell to show.
- **options** – extra argument string to pass to command

- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.freepages (cellno=None, pagesize=None, options='', **dargs)`

Display available free pages for the NUMA cell

Parameters

- **cellno** – NUMA cell number
- **pagesize** – Page size (in kibibytes)
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.has_command_help_match (virsh_cmd, regex, **dargs)`

Regex search on subcommand help output

Parameters

- **virsh_cmd** – Name of virsh command or group to match help output
- **regex** – regular expression string to match
- **dargs** – standardized virsh function API keywords

Returns re match object

`virttest.virsh.has_help_command (virsh_cmd, options='', **dargs)`

String match on virsh command in help output command list

Parameters

- **virsh_cmd** – Name of virsh command or group to look for
- **options** – Additional options to send to help command
- **dargs** – standardized virsh function API keywords

Returns True/False

`virttest.virsh.help (virsh_cmd='', **dargs)`

Prints global help, command specific help, or help for a group of related commands

Parameters

- **virsh_cmd** – Name of virsh command or group
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.help_command (options='', cache=False, **dargs)`

Return list of commands and groups in help command output

Parameters

- **options** – additional options to pass to help command
- **cache** – Return cached result if True, or refreshed cache if False
- **dargs** – standardized virsh function API keywords

Returns List of command and group names

`virttest.virsh.help_command_group(options='', cache=False, **dargs)`

Return list of groups in help command output

Parameters

- **options** – additional options to pass to help command
- **cache** – Return cached result if True, or refreshed cache if False
- **dargs** – standardized virsh function API keywords

Returns List of group names

`virttest.virsh.help_command_only(options='', cache=False, **dargs)`

Return list of commands in help command output

Parameters

- **options** – additional options to pass to help command
- **cache** – Return cached result if True, or refreshed cache if False
- **dargs** – standardized virsh function API keywords

Returns List of command names

`virttest.virsh.hostname(option='', **dargs)`

Return the hypervisor hostname.

Parameters

- **option** – additional option string to pass
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.iface_begin(**dargs)`

Create a snapshot of current interfaces settings

Param dargs: standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.iface_bridge(iface, bridge, extra='', **dargs)`

Create a bridge device and attach an existing network device to it.

Parameters

- **iface** – Interface name or MAC address
- **bridge** – New bridge device name
- **extra** – Free-form string of options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_commit(**dargs)`

Commit changes made since iface-begin and free restore point

Param dargs: standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.iface_define(xml_path, **dargs)`

Define (but don't start) a physical host interface from an XML file.

Parameters

- **xml_path** – XML file path
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_destroy (iface, **dargs)`
Destroy a physical host interface.

Parameters

- **iface** – Interface name or MAC address
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_dumpxml (iface, extra='', to_file='', **dargs)`
Interface information in XML.

Parameters

- **iface** – Interface name or MAC address
- **extra** – Free-form string of options
- **to_file** – Optional file to write xml
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.iface_edit (iface, **dargs)`
Edit XML configuration for a physical host interface.

Parameters

- **iface** – Interface name or MAC address
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_list (extra='', **dargs)`
List physical host interfaces.

Parameters

- **extra** – Free-form string of options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_mac (name, **dargs)`
Convert an interface name to interface MAC address.

Parameters

- **name** – Interface name
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.iface_name (mac, **dargs)`
Convert an interface MAC address to interface name.

Parameters

- **mac** – Interface MAC address
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object`virttest.virsh.iface_rollback (**dargs)`

Rollback to previous saved configuration created via iface-begin

Param dargs: standardized virsh function API keywords**Returns** CmdResult instance`virttest.virsh.iface_start (iface, **dargs)`

Start a physical host interface.

Parameters

- **iface** – Interface name or MAC address
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object`virttest.virsh.iface_unbridge (bridge, extra='', **dargs)`

Undefine a bridge device after detaching its slave device.

Parameters

- **bridge** – Current bridge device name
- **extra** – Free-form string of options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object`virttest.virsh.iface_undefine (iface, **dargs)`

Undefine a physical host interface (remove it from configuration).

Parameters

- **iface** – Interface name or MAC address
- **dargs** – Standardized virsh function API keywords

Returns CmdResult object`virttest.virsh.inject_nmi (name, options='', **dargs)`

Inject NMI to the guest

Parameters

- **name** – domain name
- **options** – extra options

`virttest.virsh.is_alive (name, **dargs)`

Return True if the domain is started/alive.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns True operation was successful

`virttest.virsh.is_dead(name, **dargs)`

Return True if the domain is undefined or not started/dead.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns True operation was successful

`virttest.virsh.managedsave(name, options='', **dargs)`

Managed save of a domain state.

Parameters

- **name** – Name of domain to save
- **options** – options: options to pass to list command

Returns CmdResult object

`virttest.virsh.managedsave_remove(name, **dargs)`

Remove managed save of a domain

Parameters **name** – name of managed-saved domain to remove

Returns CmdResult object

`virttest.virsh.maxvcpus(option='', **dargs)`

Return the connection vcpu maximum number.

Param option: additional option string to pass

Param dargs: standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.memtune_get(name, key)`

Get the specific memory controller value

Parameters

- **domname** – VM Name
- **key** – memory controller limit for which the value needed

Returns the memory value of a key in Kbs

`virttest.virsh.memtune_list(name, **dargs)`

List the memory controller value of a given domain

Parameters **domname** – VM Name

`virttest.virsh.memtune_set(name, options, **dargs)`

Set the memory controller parameters

Parameters

- **domname** – VM Name
- **options** – contains the values limit, state and value

`virttest.virsh.metadata(name, uri, options='', key=None, new_metadata=None, **dargs)`

Show or set domain's custom XML Metadata

Parameters

- **name** – Domain name, id or uuid

- **uri** – URI of the namespace
- **options** – options may be live, config and current
- **key** – Key to be used as a namespace identifier
- **new_metadata** – new metadata to set
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.migrate(name='', dest_uri='', option='', extra='', **dargs)`
Migrate a guest to another host.

Parameters

- **name** – name of guest on uri.
- **dest_uri** – libvirt uri to send guest to
- **option** – Free-form string of options to virsh migrate
- **extra** – Free-form string of options to follow <domain> <desturi>
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.migrate_compcache(domain, size=None, **dargs)`
Get/set compression cache size for migration.

Parameters

- **domain** – name/uuid/id of guest
- **size** – compression cache size to be set.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.migrate_getspeed(domain, **dargs)`
Get the maximum migration bandwidth (in MiB/s) for a domain.

Parameters

- **domain** – name/uuid/id of guest
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.migrate_setmaxdowntime(domain, downtime, extra=None, **dargs)`
Set maximum tolerable downtime of a domain which is being live-migrated to another host.

Parameters

- **domain** – name/uuid/id of guest
- **downtime** – downtime number of live migration

`virttest.virsh.migrate_setspeed(domain, bandwidth, extra=None, **dargs)`
Set the maximum migration bandwidth (in MiB/s) for a domain which is being migrated to another host.

Parameters

- **domain** – name/uuid/id of guest
- **bandwidth** – migration bandwidth limit in MiB/s

- **dargs** – standardized virsh function API keywords

`virttest.virsh.move_mouse` (*name*, *coordinate*, ***dargs*)
Move VM mouse.

Parameters

- **name** – domain name
- **coordinate** – Mouse coordinate

`virttest.virsh.net_autostart` (*network*, *extra*='', ***dargs*)
Set/unset a network to autostart on host boot

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra parameters to pass to command (e.g. `--disable`)
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_create` (*xml_file*, *extra*='', ***dargs*)
Create `_transient_` network from a XML file.

Parameters

- **xml_file** – xml defining network
- **extra** – extra parameters to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_define` (*xml_file*, *extra*='', ***dargs*)
Define network from a XML file, do not start

Parameters

- **xml_file** – xml defining network
- **extra** – extra parameters to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_destroy` (*network*, *extra*='', ***dargs*)
Destroy (stop) an activated network on host.

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra string to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_dhcp_leases` (*network*, *mac*=None, *options*='', ***dargs*)
Print lease info for a given network

Parameters

- **network** – Network name or uuid

- **mac** – Mac address
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.net_dumpxml (name, extra='', to_file='', **dargs)`
Dump XML from network named param name.

Parameters

- **name** – Name of a network
- **extra** – Extra parameters to pass to command
- **to_file** – Send result to a file
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_event (network=None, event=None, event_timeout=None, options='', **dargs)`
List event types, or wait for network events to occur

Parameters

- **network** – Network name or uuid
- **event** – Event type to wait for
- **event_timeout** – Timeout seconds
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.net_info (network, extra='', **dargs)`
Get network information

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra parameters to pass to command.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.net_list (options, extra='', **dargs)`
List networks on host.

Parameters

- **options** – options to pass to command
- **extra** – extra parameters to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_name (uuid, extra='', **dargs)`
Get network name on host.

Parameters

- **uuid** – network UUID.
- **extra** – extra parameters to pass to command.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_start(network, extra='', **dargs)`
Start network on host.

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra parameters to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_state_dict(only_names=False, virsh_instance=None, **dargs)`
Return network name to state/autostart/persistent mapping

Parameters

- **only_names** – When true, return network names as keys and None values
- **virsh_instance** – Call `net_list()` on this instance instead of module
- **dargs** – standardized virsh function API keywords

Returns dictionary

`virttest.virsh.net_undefine(network, extra='', **dargs)`
Undefine a defined network on host.

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra string to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.net_update(network, update_cmd, section, xml, extra='', **dargs)`
Update parts of an existing network's configuration

Parameters

- **network** – network name or uuid
- **update_cmd** – type of update (add-first, add-last, delete, or modify)
- **section** – which section of network configuration to update
- **xml** – name of file containing xml
- **extra** – extra parameters to pass to command.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.net_uuid(network, extra='', **dargs)`
Get network UUID on host.

Parameters

- **network** – name/parameter for network option/argument
- **extra** – extra parameters to pass to command.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.node_memtune (shm_pages_to_scan=None, shm_sleep_millisecs=None,  
                             shm_merge_across_nodes=None, options=None, **dargs)
```

Get or set node memory parameters.

Parameters

- **options** – Extra options to virsh.
- **shm-pages-to-scan** – Pages to scan.
- **shm-sleep-millisecs** – Sleep time (ms).
- **shm-merge-across-nodes** – Merge across nodes.
- **dargs** – Standardized virsh function API keywords.

Returns CmdResult instance

```
virttest.virsh.nodecpumap (extra='', **dargs)
```

Displays the node's total number of CPUs, the number of online CPUs and the list of online CPUs.

Parameters

- **extra** – extra argument string to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.nodecpustats (option='', **dargs)
```

Returns basic information about the node CPU statistics

Parameters

- **option** – additional options (takes none)
- **dargs** – standardized virsh function API keywords

```
virttest.virsh.nodedev_create (xml_file, options=None, **dargs)
```

Return cmd result of the device to be created by an XML file

Parameters

- **xml_file** – device XML file
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.nodedev_destroy (dev_name, options=None, **dargs)
```

Return cmd result of the device to be destroyed

Parameters

- **dev_name** – name of the device
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

```
virttest.virsh.nodedev_detach (name, options='', **dargs)
```

Detach node device from host.

Returns cmdresult object.

`virttest.virsh.nodedev_dettach (name, options='', **dargs)`
Detach node device from host.

Returns nodedev_detach(name).

`virttest.virsh.nodedev_dumpxml (name, options='', to_file=None, **dargs)`
Do dumpxml for node device.

Parameters

- **name** – the name of device.
- **options** – extra options to nodedev-dumpxml cmd.
- **to_file** – optional file to write XML output to.

Returns Cmdobject of virsh nodedev-dumpxml.

`virttest.virsh.nodedev_list (tree=False, cap='', options='', **dargs)`
List the node devices.

Parameters

- **tree** – list devices in a tree
- **cap** – capability names, separated by comma
- **options** – extra command options.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object.

`virttest.virsh.nodedev_reattach (name, options='', **dargs)`
If node device is detached, this action will reattach it to its device driver.

Returns cmdresult object.

`virttest.virsh.nodedev_reset (name, options='', **dargs)`
Trigger a device reset for device node.

Parameters

- **name** – device node name to be reset.
- **options** – additional options passed to virsh command
- **dargs** – standardized virsh function API keywords

Returns cmdresult object.

`virttest.virsh.nodeinfo (extra='', **dargs)`
Returns basic information about the node, like number and type of CPU, and size of the physical memory.

Parameters

- **extra** – extra argument string to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.nodememstats (option='', **dargs)`
Returns basic information about the node Memory statistics

Parameters

- **option** – additional options (takes none)
- **dargs** – standardized virsh function API keywords

`virttest.virsh.nodesuspend(target, duration, extra='', **dargs)`

Suspend the host node for a given time duration.

Parameters

- **target** – Suspend target mem/disk/hybrid. mem(Suspend-to-RAM) disk(Suspend-to-Disk) hybrid(Hybrid-Suspend)
- **duration** – Suspend duration in seconds, at least 60.
- **extra** – extra argument string to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.numatune(name, mode=None, nodeset=None, options=None, **dargs)`

Set or get a domain's numa parameters :param name: name of domain :param options: options may be live, config and current :param dargs: standardized virsh function API keywords :return: CmdResult instance

`virttest.virsh.nwfilter_define(xml_file, options='', **dargs)`

Return cmd result of network filter define.

Parameters

- **xml_file** – network filter XML file
- **options** – extra options to nwfilter-define cmd.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.nwfilter_dumpxml(name, options='', to_file=None, **dargs)`

Do dumpxml for network filter.

Parameters

- **name** – the name or uuid of filter.
- **options** – extra options to nwfilter-dumpxml cmd.
- **to_file** – optional file to write XML output to.
- **dargs** – standardized virsh function API keywords

Returns Cmdobject of virsh nwfilter-dumpxml.

`virttest.virsh.nwfilter_edit(name, options='', **dargs)`

Edit the XML configuration for a network filter.

Parameters

- **name** – network filter name or uuid.
- **options** – extra options to nwfilter-edit cmd.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.nwfilter_list(options='', **dargs)`

Get list of network filters.

Parameters

- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns list of network filters

`virttest.virsh.nwfilter_undefine (name, options='', **dargs)`

Return cmd result of network filter undefine.

Parameters

- **name** – network filter name or uuid
- **options** – extra options to nwfilter-undefine cmd.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_autostart (name, extra='', **dargs)`

Mark for autostart of a pool

Parameters

- **name** – Name of the pool to be mark for autostart
- **extra** – Free-form string of options
- **dargs** – standardized virsh function API keywords

Returns True if pool autostart command was successful

`virttest.virsh.pool_build (name, options='', **dargs)`

Build pool.

Parameters

- **name** – Name of the pool to be built
- **options** – options for pool-build

`virttest.virsh.pool_create (xml_file, extra='', **dargs)`

Create a pool from an xml file.

Parameters

- **xml_file** – file containing an XML pool description
- **extra** – extra parameters to pass to command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_create_as (name, pool_type, target, extra='', **dargs)`

Create a pool from a set of args.

Parameters

- **name** – name of pool
- **pool_type** – storage pool type such as 'dir'
- **target** – libvirt uri to send guest to
- **extra** – Free-form string of options
- **dargs** – standardized virsh function API keywords

Returns True if pool creation command was successful

`virttest.virsh.pool_define(xml_path, **dargs)`

To create the pool from xml file.

Parameters

- **xml_path** – XML file path
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_define_as(name, pool_type, target='', extra='', **dargs)`

Define the pool from the arguments

Parameters

- **name** – Name of the pool to be defined
- **pool_type** – Type of the pool to be defined
 - dir** file system directory
 - disk** Physical Disk Device
 - fs** Pre-formatted Block Device
 - netfs** Network Exported Directory
 - iscsi** iSCSI Target
 - logical** LVM Volume Group
 - mpath** Multipath Device Enumerator
 - scsi** SCSI Host Adapter
 - rbd** Rados Block Device
- **target** – libvirt uri to send guest to
- **extra** – Free-form string of options
- **dargs** – standardized virsh function API keywords

Returns True if pool define command was successful

`virttest.virsh.pool_delete(name, **dargs)`

Delete the resources used by a given pool object

Parameters

- **name** – Name of the pool
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_destroy(name, **dargs)`

Forcefully stop a given pool.

Parameters

- **name** – name of pool
- **dargs** – standardized virsh function API keywords

`virttest.virsh.pool_dumpxml(name, extra='', to_file='', **dargs)`

Return the pool information as an XML dump.

Parameters

- **name** – pool_name name
- **to_file** – optional file to write XML output to
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.pool_edit (name, **dargs)`
Edit XML configuration for a storage pool.

Parameters

- **name** – pool name or uuid
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_info (name, **dargs)`
Returns basic information about the storage pool.

Parameters

- **name** – name of pool
- **dargs** – standardized virsh function API keywords

`virttest.virsh.pool_list (option='', extra='', **dargs)`
Prints the pool information of Host.

Parameters **option** – options given to command

all gives all pool details, including inactive

inactive gives only inactive pool details

details Gives the complete details about the pools

Parameters **extra** – to provide extra options(to enter invalid options)

`virttest.virsh.pool_name (uuid, **dargs)`
Convert a pool UUID to pool name

Parameters

- **name** – UUID of the pool
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_refresh (name, **dargs)`
Refresh a pool

Parameters

- **name** – Name of the pool
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pool_start (name, extra='', **dargs)`
Start the defined pool

Parameters

- **name** – Name of the pool to be started
- **extra** – Free-form string of options
- **dargs** – standardized virsh function API keywords

Returns True if pool start command was successful

`virttest.virsh.pool_state_dict (only_names=False, **dargs)`

Return pool name to state/autostart mapping

Parameters

- **only_names** – When true, return pool names as keys and None values
- **dargs** – standardized virsh function API keywords

Returns dictionary

`virttest.virsh.pool_undefine (name, extra='', **dargs)`

Undefine the given pool

Parameters

- **name** – Name of the pool to be undefined
- **extra** – Free-form string of options
- **dargs** – standardized virsh function API keywords

Returns True if pool undefine command was successful

`virttest.virsh.pool_uuid (name, **dargs)`

Convert a pool name to pool UUID

Parameters

- **name** – Name of the pool
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.pwd (options='', **dargs)`

Run pwd command in virsh session.

Parameters

- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.qemu_agent_command (name, cmd, options='', **dargs)`

This helps to execute the qemu agent command through virsh command.

Parameters

- **name** – Name of monitor domain
- **cmd** – agent command to execute
- **options** – extra options
- **dargs** – standardized virsh function API keywords

`virttest.virsh.qemu_attach (pid, extra='', **dargs)`

This helps to execute the qemu-attach command through virsh command.

Parameters

- **pid** – pid of qemu process
- **extra** – extra options
- **dargs** – standardized virsh function API keywords

`virttest.virsh.gemu_monitor_command(name, cmd, options='', **dargs)`

This helps to execute the qemu monitor command through virsh command.

Parameters

- **name** – Name of monitor domain
- **cmd** – monitor command to execute
- **options** – extra options
- **dargs** – standardized virsh function API keywords

`virttest.virsh.gemu_monitor_event(domain=None, event=None, event_timeout=None, options='', **dargs)`

Listen for QEMU Monitor Events

Parameters

- **domain** – Domain name, id or UUID
- **event** – Event type name
- **event_timeout** – Timeout seconds
- **options** – Extra options
- **dargs** – Standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.quit(**dargs)`

Run quit command in virsh session.

Parameters **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.reboot(name, options='', **dargs)`

Run a reboot command in the target domain.

Parameters

- **name** – Name of domain.
- **options** – options: options to pass to reboot command

Returns CmdResult object

`virttest.virsh.remove_domain(name, options=None, **dargs)`

Return True after forcefully removing a domain if it exists.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns True operation was successful

`virttest.virsh.reset(name, **dargs)`

Reset a domain

Parameters **name** – name of domain

Returns CmdResult object

`virttest.virsh.restore(path, options='', **dargs)`

Load state of VM from named file and remove file.

Parameters

- **path** – absolute path to state file.
- **options** – options for virsh restore.
- **dargs** – standardized virsh function API keywords

`virttest.virsh.resume(name, **dargs)`

True on successful moving domain out of suspend

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.save(name, path, options='', **dargs)`

Store state of VM into named file.

Parameters

- **name** – VM name, id or uuid.
- **path** – absolute path to state file
- **options** – command's options.
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.save_image_define(state_file, xmlfile, options='', **dargs)`

Redefine the XML for a domain's saved state file

Parameters

- **state_file** – saved state file to modify
- **xmlfile** – filename containing updated XML for the target
- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.save_image_dumpxml(state_file, options='', to_file='', **dargs)`

Dump xml from saved state file

Parameters

- **state_file** – saved state file to read
- **options** – extra options
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.schedinfo` (*domain*, *options*='', ***dargs*)
Show/Set scheduler parameters.

Parameters

- **domain** – vm's name id or uuid.
- **options** – additional options.
- **dargs** – standardized virsh function API keywords

`virttest.virsh.screenshot` (*name*, *filename*, ***dargs*)
Capture a screenshot of VM's console and store it in file on host

Parameters

- **name** – VM name
- **filename** – name of host file
- **dargs** – standardized virsh function API keywords

Returns filename

`virttest.virsh.screenshot_test` (*name*, *filename*='', *options*='', ***dargs*)
Capture a screenshot of VM's console and store it in file on host

Parameters

- **name** – VM name or id
- **filename** – name of host file
- **options** – command options
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.secret_define` (*xml_file*, *options*=None, ***dargs*)
Return cmd result of secret define.

Parameters

- **xml_file** – secret XML file
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.secret_dumpxml` (*uuid*, *to_file*='', *options*=None, ***dargs*)
Return the secret information as an XML dump.

Parameters

- **uuid** – secret UUID
- **to_file** – optional file to write XML output to
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.secret_get_value` (*uuid*, *options*=None, ***dargs*)
Get a secret value

Parameters **uuid** – secret UUID

Returns CmdResult object.

`virttest.virsh.secret_list (options='', **dargs)`

Get list of secret.

Parameters

- **options** – the option may be ‘–ephemeral’
- **dargs** – standardized virsh function API keywords

Returns list of secret

`virttest.virsh.secret_set_value (uuid, base64, options=None, **dargs)`

Set a secret value

Parameters

- **uuid** – secret UUID
- **base64** – base64-encoded secret value

Returns CmdResult object.

`virttest.virsh.secret_undefine (uuid, options=None, **dargs)`

Return cmd result of secret undefine.

Parameters

- **uuid** – secret UUID
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.sendkey (name, options='', **dargs)`

Send keycodes to the guest :param name: name of domain :param codeset: the codeset of keycodes :param keycode: the key code :return: CmdResult object

`virttest.virsh.setmaxmem (domainarg=None, sizearg=None, domain=None, size=None, use_kilobytes=False, flagstr='', **dargs)`

Change the maximum memory allocation for the guest domain.

Parameters

- **domainarg** – Domain name (first pos. parameter)
- **sizearg** – Memory size in KiB (second. pos. parameter)
- **domain** – Option to –domain parameter
- **size** – Option to –size or –kilobytes parameter
- **use_kilobytes** – True for –kilobytes, False for –size
- **flagstr** – string of “–config, –live, –current, etc.”

Returns CmdResult instance

Raise error.CmdError: if libvirtd is not running.

`virttest.virsh.setmem (domainarg=None, sizearg=None, domain=None, size=None, use_kilobytes=False, flagstr='', **dargs)`

Change the current memory allocation in the guest domain.

Parameters

- **domainarg** – Domain name (first pos. parameter)
- **sizearg** – Memory size in KiB (second. pos. parameter)

- **domain** – Option to `--domain` parameter
- **size** – Option to `--size` or `--kilobytes` parameter
- **use_kilobytes** – True for `--kilobytes`, False for `--size`
- **dargs** – standardized virsh function API keywords
- **flagstr** – string of “`--config`, `--live`, `--current`, etc.”

Returns CmdResult instance

Raise error.CmdError: if libvirtd is not running

`virttest.virsh.setvcpus(name, count, extra='', **dargs)`

Change the number of virtual CPUs in the guest domain.

Parameters

- **name** – name of vm to affect
- **count** – value for `vcpu` parameter
- **options** – any extra command options.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object from command

`virttest.virsh.shutdown(name, options='', **dargs)`

True on successful domain shutdown.

Parameters

- **name** – VM name
- **options** – options for virsh shutdown.
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.snapshot_create(name, options='', **dargs)`

Create snapshot of domain.

Parameters

- **name** – name of domain
- **dargs** – standardized virsh function API keywords

Returns name of snapshot

`virttest.virsh.snapshot_create_as(name, options='', **dargs)`

Create snapshot of domain with options.

Parameters

- **name** – name of domain
- **options** – options of `snapshot-create-as`
- **dargs** – standardized virsh function API keywords

Returns name of snapshot

`virttest.virsh.snapshot_current(name, options='--name', **dargs)`

Get name or xml of current snapshot.

Parameters

- **name** – name of domain
- **options** – options of snapshot-current, default is `-name`
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.snapshot_delete(name, snapshot, options='', **dargs)`
Remove domain snapshot

Parameters

- **name** – name of domain
- **dargs** – standardized virsh function API keywords
- **snapshot** – snapshot to delete

Returns CmdResult instance

`virttest.virsh.snapshot_dumpxml(name, snapshot, options=None, to_file=None, **dargs)`
Get dumpxml of snapshot

Parameters

- **name** – name of domain
- **snapshot** – name of snapshot
- **options** – options of snapshot_list
- **to_file** – optional file to write XML output to
- **dargs** – standardized virsh function API keywords

Returns standard output from command

`virttest.virsh.snapshot_edit(name, options='', **dargs)`
Edit snapshot xml

Parameters

- **name** – name of domain
- **options** – options of snapshot-edit command
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.snapshot_info(name, snapshot, **dargs)`
Check snapshot information.

Parameters

- **name** – name of domain
- **snapshot** – name os snapshot to verify
- **dargs** – standardized virsh function API keywords

Returns snapshot information dictionary

`virttest.virsh.snapshot_list(name, options=None, **dargs)`
Get list of snapshots of domain.

Parameters

- **name** – name of domain

- **options** – options of snapshot_list
- **dargs** – standardized virsh function API keywords

Returns list of snapshot names

`virttest.virsh.snapshot_parent` (*name*, *options*, ***dargs*)
Get name of snapshot parent

Parameters

- **name** – name of domain
- **options** – options of snapshot-parent
- **dargs** – standardized virsh function API keywords

Returns name of snapshot

`virttest.virsh.snapshot_revert` (*name*, *snapshot*, *options=''*, ***dargs*)
Revert domain state to saved snapshot.

Parameters

- **name** – name of domain
- **dargs** – standardized virsh function API keywords
- **snapshot** – snapshot to revert to

Returns CmdResult instance

`virttest.virsh.start` (*name*, *options=''*, ***dargs*)
True on successful start of (previously defined) inactive domain.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult object.

`virttest.virsh.suspend` (*name*, ***dargs*)
True on successful suspend of VM - kept in memory and not scheduled.

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.sysinfo` (*options=''*, ***dargs*)
Return the hypervisor sysinfo xml.

Parameters **options** – extra options

Returns CmdResult object

`virttest.virsh.ttyconsole` (*name*, ***dargs*)
Print tty console device.

Parameters **name** – name, uuid or id of domain

Returns CmdResult instance

`virttest.virsh.undefine` (*name*, *options=None*, ***dargs*)

Return cmd result of domain undefine (after shutdown/destroy).

Parameters

- **name** – VM name
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.update_device` (*domainarg=None*, *filearg=None*, *domain_opt=None*,
file_opt=None, *flagstr=''*, ***dargs*)

Update device from an XML <file>.

Parameters

- **domainarg** – Domain name (first pos. parameter)
- **filearg** – File name (second pos. parameter)
- **domain_opt** – Option to `--domain` parameter
- **file_opt** – Option to `--file` parameter
- **flagstr** – string of “`--force`, `--persistent`, etc.”
- **dargs** – standardized virsh function API keywords

Returns CmdResult instance

`virttest.virsh.vcpucount` (*name*, *options=''*, ***dargs*)

Get the vcpu count of guest.

Parameters

- **name** – name of domain.
- **options** – options for `vcpuoutn` command.

Returns CmdResult object.

`virttest.virsh.vcpuinfo` (*name*, ***dargs*)

Parameters

- **name** – name of domain
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.vcpupin` (*name*, *vcpu=''*, *cpu_list=''*, *options=''*, ***dargs*)

Changes the cpu affinity for respective vcpu.

Parameters

- **name** – name of domain
- **vcpu** – virtual CPU to modify
- **cpu_list** – physical CPU specification (string)
- **dargs** – standardized virsh function API keywords
- **options** – `--live`, `--current` or `--config`.

Returns CmdResult object.

`virttest.virsh.version(option='', **dargs)`

Return the major version info about what this built from.

Parameters

- **option** – additional option string to pass
- **dargs** – standardized virsh function API keywords

Returns CmdResult object

`virttest.virsh.vncdisplay(name, **dargs)`

Output the IP address and port number for the VNC display.

Parameters

- **name** – VM's name or id,uuid.
- **dargs** – standardized virsh function API keywords.

Returns result from command

`virttest.virsh.vol_clone(volume_name, new_name, pool_name='', extra='', **dargs)`

Clone an existing volume.

Parameters

- **volume_name** – Name of the original volume
- **new_name** – Clone name
- **pool_name** – Name of the pool
- **extra** – Free-form string options
- **dargs** – Standardized virsh function API keywords

Returns Returns the output of the command

`virttest.virsh.vol_create(pool_name, xml_file, extra='', **dargs)`

To create the volumes from xml file.

Parameters

- **pool_name** – Name of the pool to be used
- **xml_file** – file containing an XML vol description
- **extra** – string of extra options

Returns CmdResult object

`virttest.virsh.vol_create_as(volume_name, pool_name, capacity, allocation, frmt, extra='', **dargs)`

To create the volumes on different available pool

Parameters

- **name** – Name of the volume to be created
- **pool_name** – Name of the pool to be used
- **capacity** – Size of the volume
- **allocation** – Size of the volume to be pre-allocated
- **frmt** – volume formats(e.g. raw, qed, qcow2)
- **extra** – Free-form string of options

- **dargs** – standardized virsh function API keywords

Returns True if pool undefine command was successful

`virttest.virsh.vol_create_from(pool_name, vol_file, input_vol, input_pool, extra='', **dargs)`

Create a vol, using another volume as input

Param pool_name: Name of the pool to create the volume in

Param vol_file: XML <file> with the volume definition

Param input_vol: Name of the source volume

Param input_pool: Name of the pool the source volume is in

Param extra: Free-form string of options

Returns True if volume create successfully

`virttest.virsh.vol_delete(volume_name, pool_name, extra='', **dargs)`

Delete a given volume

Parameters

- **volume_name** – Name of the volume
- **pool_name** – Name of the pool
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

`virttest.virsh.vol_download(name, dfile, options='', **dargs)`

Download volume contents to a file

Parameters

- **name** – name of volume
- **dfile** – file path that will download to
- **options** – pool name, offset and length

Returns CmdResult object

`virttest.virsh.vol_dumpxml(volume_name, pool_name, to_file=None, options='', **dargs)`

Dumps volume details in xml

Parameters

- **volume_name** – Name of the volume
- **pool_name** – Name of the pool
- **to_file** – path of the file to store the output
- **options** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

`virttest.virsh.vol_info(volume_name, pool_name, extra='', **dargs)`

Prints the given volume info

Parameters

- **volume_name** – Name of the volume

- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

```
virttest.virsh.vol_key(volume_name, pool_name, extra='', **dargs)
```

Prints the key of the given volume name

Parameters

- **volume_name** – Name of the volume
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

```
virttest.virsh.vol_list(pool_name, extra='', **dargs)
```

List the volumes for a given pool

Parameters

- **pool_name** – Name of the pool
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

```
virttest.virsh.vol_name(volume_key, extra='', **dargs)
```

Prints the given volume name

Parameters

- **volume_name** – Name of the volume
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

```
virttest.virsh.vol_path(volume_name, pool_name, extra='', **dargs)
```

Prints the give volume path

Parameters

- **volume_name** – Name of the volume
- **pool_name** – Name of the pool
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

```
virttest.virsh.vol_pool(volume_name, extra='', **dargs)
```

Returns pool name for a given vol-key

Parameters

- **volume_name** – Name of the volume
- **extra** – Free-form string options
- **dargs** – standardized virsh function API keywords

Returns returns the output of the command

`virttest.virsh.vol_resize(volume_name, capacity, pool_name='', extra='', **dargs)`
Resizes a storage volume.

Parameters

- **volume_name** – Name of the volume
- **capacity** – New capacity for the volume (default bytes)
- **pool_name** – Name of the pool
- **extra** – Free-form string options
- **dargs** – Standardized virsh function API keywords

Returns Returns the output of the command

`virttest.virsh.vol_upload(name, dfile, options='', **dargs)`
Upload file contents to a volume

Parameters

- **name** – name of volume
- **dfile** – file path that will upload from
- **options** – pool name, offset and length

Returns CmdResult object

`virttest.virsh.vol_wipe(volume_name, pool_name='', alg='', **dargs)`
Ensure data previously on a volume is not accessible to future reads.

Parameters

- **volume_name** – Name of the volume
- **pool_name** – Name of the pool
- **alg** – Perform selected wiping algorithm
- **dargs** – Standardized virsh function API keywords

Returns Returns the output of the command

virttest.virsh_unittest module

```
class virttest.virsh_unittest.ConstantsTest (methodName='runTest')
    Bases: virttest.virsh_unittest.ModuleLoad
    test_ModuleLoad()

class virttest.virsh_unittest.ConstructorsTest (methodName='runTest')
    Bases: virttest.virsh_unittest.ModuleLoad
    TestVirshClosure()
    test_Virsh()
    test_VirshBase()
    test_VirshPersistent()
```

`virttest.virsh_unittest.FakeVirshFactory` (*preserve=None*)

Return `Virsh()` instance with methods to raise `bogusVirshFailureException`.

Users of this class should override methods under test on instance. :param `preserve`: List of symbol names NOT to modify, `None` for all

class `virttest.virsh_unittest.ModuleLoad` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

virsh = <module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest.virsh'>

class `virttest.virsh_unittest.ModuleLoadCheckVirsh` (*methodName='runTest'*)

Bases: `unittest.case.TestCase`

run (**args, **dargs*)

virsh = <module 'virttest.virsh' from '/home/docs/checkouts/readthedocs.org/user_builds/virt-test/checkouts/latest/virttest.virsh'>

class `virttest.virsh_unittest.SessionManagerTest` (*methodName='runTest'*)

Bases: `virttest.virsh_unittest.ModuleLoadCheckVirsh`

test_VirshPersistent ()

Unittest for session manager of `VirshPersistent`.

test_VirshSession ()

Unittest for `VirshSession`.

This test use `VirshSession` over `VirshPersistent` with `auto_close=True`.

test_del_VirshPersistent ()

Unittest for `__del__` of `VirshPersistent`.

This test makes sure the `__del__` method of `VirshPersistent` works well in *del vp_instance*.

class `virttest.virsh_unittest.TestVirshClosure` (*methodName='runTest'*)

Bases: `virttest.virsh_unittest.ModuleLoad`

class `SomeClass`

Bases: `dict`

somemethod ()

static `TestVirshClosure.somefunc` (**args, **dargs*)

`TestVirshClosure.test_args` ()

`TestVirshClosure.test_args_and_dargs` ()

`TestVirshClosure.test_args_dargs_subclass` ()

`TestVirshClosure.test_dargs` ()

`TestVirshClosure.test_fake_virsh` ()

`TestVirshClosure.test_init` ()

`TestVirshClosure.test_multi_inst` ()

`TestVirshClosure.test_update_args_dargs_subclass` ()

class `virttest.virsh_unittest.VirshClassHasHelpCommandTest` (*methodName='runTest'*)

Bases: `virttest.virsh_unittest.VirshHasHelpCommandTest`

setUp ()

class `virttest.virsh_unittest.VirshHasHelpCommandTest` (*methodName='runTest'*)

Bases: `virttest.virsh_unittest.ModuleLoadCheckVirsh`

```

    setUp()
    test_false_command()
    test_groups_in_commands()
    test_no_cache()
    test_subcommand_help()
    test_true_command()
class virttest.virsh_unittest.VirshHelpCommandTest (methodName='runTest')
    Bases: virttest.virsh_unittest.ModuleLoadCheckVirsh
    test_cache_command()
class virttest.virsh_unittest.VirshPersistentClassHasHelpCommandTest (methodName='runTest')
    Bases: virttest.virsh_unittest.VirshHasHelpCommandTest
    setUp()
    tearDown()
    test_recycle_session()
exception virttest.virsh_unittest.bogusVirshFailureException (*args, **dargs)
    Bases: exceptions.AssertionError

```

virttest.virt_vm module

```

class virttest.virt_vm.BaseVM(name, params)
    Bases: object

```

Base class for all hypervisor specific VM subclasses.

This class should not be used directly, that is, do not attempt to instantiate and use this class. Instead, one should implement a subclass that implements, at the very least, all methods defined right after the the comment blocks that are marked with:

“Public API - *must* be reimplemented with virt specific code”

and

“Protected API - *must* be reimplemented with virt specific classes”

The current proposal regarding methods naming convention is:

- Public API methods: named in the usual way, consumed by tests
- Protected API methods: name begins with a single underline, to be consumed only by BaseVM and sub-classes
- Private API methods: name begins with double underline, to be consumed only by the VM subclass itself (usually implements virt specific functionality: example: `__make_qemu_command()`)

So called “protected” methods are intended to be used only by VM classes, and not be consumed by tests. Theses should respect a naming convention and always be preceded by a single underline.

Currently most (if not all) methods are public and appears to be consumed by tests. It is a ongoing task to determine whether methods should be “public” or “protected”.

```
COPY_FILES_TIMEOUT = 600
```

```
LOGIN_TIMEOUT = 10
```

LOGIN_WAIT_TIMEOUT = 240

MIGRATE_TIMEOUT = 3600

MIGRATION_PROTOS = ['tcp']

REBOOT_TIMEOUT = 240

activate_nic (*nic_index_or_name*)

Activate an inactive network device

Parameters **nic_index_or_name** – name or index number for existing NIC

add_nic (***params*)

Add new or setup existing NIC with optional model type and mac address

Parameters **params** – Dict with additional NIC parameters to set.

Returns Dict with new NIC's info.

cleanup_serial_console ()

Close serial console and associated log file

clone (*name, **params*)

Return a clone of the VM object with optionally modified parameters.

This method should be implemented by

commander (**args, **kwargs*)

Log into the guest via SSH/Telnet/Netcat. If timeout expires while waiting for output from the guest (e.g. a password prompt or a shell prompt) – fail.

Parameters

- **nic_index** – The index of the NIC to connect to.
- **timeout** – Time (seconds) before giving up logging into the guest.
- **commander_path** – Path where will be commander placed.

Returns A ShellSession object.

copy_files_from (**args, **kwargs*)

Transfer files from the guest.

Parameters

- **host_path** – Guest path
- **guest_path** – Host path
- **nic_index** – The index of the NIC to connect to.
- **limit** – Speed limit of file transfer.
- **verbose** – If True, log some stats using logging.debug (RSS only)
- **timeout** – Time (seconds) before giving up on doing the remote copy.

copy_files_to (**args, **kwargs*)

Transfer files to the remote host(guest).

Parameters

- **host_path** – Host path
- **guest_path** – Guest path
- **nic_index** – The index of the NIC to connect to.

- **limit** – Speed limit of file transfer.
- **verbose** – If True, log some stats using logging.debug (RSS only)
- **timeout** – Time (seconds) before giving up on doing the remote copy.

create_serial_console()

Establish a session with the serial console.

Let's consider the first serial port as serial console. Note: requires a version of netcat that supports -U

create_virtio_console()

Establish a session with the virtio console.

deactivate_nic(*nic_index_or_name*)

Deactivate an active network device

Parameters *nic_index_or_name* – name or index number for existing NIC

del_nic(*nic_index_or_name*)

Remove the nic specified by name, or index number

destroy(*gracefully=True, free_mac_addresses=True*)

Destroy the VM.

If gracefully is True, first attempt to shutdown the VM with a shell command. Then, attempt to destroy the VM via the monitor with a 'quit' command. If that fails, send SIGKILL to the qemu process.

Parameters

- **gracefully** – If True, an attempt will be made to end the VM using a shell command before trying to end the qemu process with a 'quit' or a kill signal.
- **free_mac_addresses** – If True, the MAC addresses used by the VM will be freed.

fill_addr(*addrs*)

Fill VM's nic address to the virtnet structure based on VM's address structure addrs.

Parameters *addrs* – Dict of interfaces and address

```
{ "if_name": { "mac": [ 'addrs', ],
               "ipv4": [ 'addrs', ],
               "ipv6": [ 'addrs', ] },
  ... }
```

free_mac_address(*nic_index_or_name=0*)

Free a NIC's MAC address.

Parameters *nic_index* – Index of the NIC

get_address(*index=0*)

Return the IP address of a NIC or guest (in host space).

Parameters *index* – Name or index of the NIC whose address is requested.

Returns 'localhost': Port redirection is in use

Returns IP address of NIC if valid in arp cache.

Raises

- **VMMACAddressMissingError** – If no MAC address is defined for the requested NIC
- **VMIPAddressMissingError** – If no IP address is found for the the NIC's MAC address

- **VMAddressVerificationError** – If the MAC-IP address mapping cannot be verified (using arping)

get_cpu_count ()

Get the cpu count of the VM.

get_current_memory_size ()

Get current memory size of the VM, rather than bootup memory.

get_mac_address (*nic_index=0*)

Return the MAC address of a NIC.

Parameters *nic_index* – Index of the NIC

Raises **VMMACAddressMissingError** – If no MAC address is defined for the requested NIC

get_memory_size (*cmd=None, timeout=60, re_str=None*)

Get bootup memory size of the VM.

Parameters

- **cmd** – Command used to check memory. If not provided, `self.params.get("mem_chk_cmd")` will be used.
- **timeout** – timeout for cmd
- **re_str** – pattern to get memory size from the command output. If not provided, `self.params.get("mem_chk_re_str")` will be used.

get_params ()

Return the VM's params dict. Most modified params take effect only upon `VM.create()`.

get_port (*port, nic_index=0*)

Return the port in host space corresponding to port in guest space.

Parameters

- **port** – Port number in host space.
- **nic_index** – Index of the NIC.

Returns If port redirection is used, return the host port redirected to guest port `port`. Otherwise return `port`.

Raises **VMPortNotRedirectedError** – If an unredirected port is requested in user mode

get_testlog_filename ()

Return the testlog filename.

get_uuid ()

Catch UUID of the VM.

Returns None, if not specified in config file

get_virtio_port_filename (*port_name*)

Return the filename corresponding to a given monitor name.

get_virtio_port_filenames ()

Return a list of all virtio port filenames (as specified in the VM's params).

is_alive ()

Return True if the VM is alive and the management interface is responsive.

is_dead ()

Return True if the VM is dead.

is_paused()

Return True if the VM is paused

loadvm(*tag_name*)

Load the virtual machine tagged 'tag_name'.

Parameters **tag_name** – tag of the virtual machine that saved

login(**args, **kwargs*)

Log into the guest via SSH/Telnet/Netcat. If timeout expires while waiting for output from the guest (e.g. a password prompt or a shell prompt) – fail.

Parameters

- **nic_index** – The index of the NIC to connect to.
- **timeout** – Time (seconds) before giving up logging into the guest.

Returns A ShellSession object.

static lookup_vm_class(*vm_type, target*)

migrate(*timeout=3600, protocol='tcp', cancel_delay=None, offline=False, stable_check=False, clean=True, save_path='/tmp', dest_host='localhost', remote_port=None*)

Migrate the VM.

If the migration is local, the VM object's state is switched with that of the destination VM. Otherwise, the state is switched with that of a dead VM (returned by self.clone()).

Parameters

- **timeout** – Time to wait for migration to complete.
- **protocol** – Migration protocol ('tcp', 'unix' or 'exec').
- **cancel_delay** – If provided, specifies a time duration after which migration will be canceled. Used for testing migrate_cancel.
- **offline** – If True, pause the source VM before migration.
- **stable_check** – If True, compare the VM's state after migration to its state before migration and raise an exception if they differ.
- **clean** – If True, delete the saved state files (relevant only if stable_check is also True).
- **save_path** – The path for state files.
- **dest_host** – Destination host (defaults to 'localhost').
- **remote_port** – Port to use for remote migration.

needs_restart(*name, params, basedir*)

Verifies whether the current virt_install commandline matches the requested one, based on the test parameters.

pause()

Stop the VM operation.

reboot(*session=None, method='shell', nic_index=0, timeout=240, serial=False*)

Reboot the VM and wait for it to come back up by trying to log in until timeout expires.

Parameters

- **session** – A shell session object or None.
- **method** – Reboot method. Can be "shell" (send a shell reboot command) or "system_reset" (send a system_reset monitor command).

- **nic_index** – Index of NIC to access in the VM, when logging in after rebooting.
- **timeout** – Time to wait for login to succeed (after rebooting).
- **serial** – Serial port login or not (default is False).

Returns A new shell session object.

remote_commander (*nic_index=0, timeout=10, username=None, password=None*)
Alias for commander() for backward compatibility.

remote_login (*nic_index=0, timeout=10, username=None, password=None*)
Alias for login() for backward compatibility.

restore_from_file (*path*)
A shutdown or paused VM is resumed from path, & possibly set running
Throws a VMStatusError if before/after restore state is incorrect

Parameters path – path to file vm state was saved to

resume ()
Resume the VM operation in case it's stopped.

save_to_file (*path*)
State of paused VM recorded to path and VM shutdown on success
Throws a VMStatusError if before/after state is incorrect.

Parameters path – file where VM state recorded

savevm (*tag_name*)
Save the virtual machine as the tag 'tag_name'

Parameters tag_name – tag of the virtual machine that saved

send_key (*keystr*)
Send a key event to the VM.

Parameters keystr – A key event string (e.g. "ctrl-alt-delete")

send_string (*sr*)
Send a string to the VM.

Parameters sr – String, that must consist of alphanumeric characters only. Capital letters are allowed.

serial_login (**args, **kwargs*)
Log into the guest via the serial console. If timeout expires while waiting for output from the guest (e.g. a password prompt or a shell prompt) – fail.

Parameters

- **timeout** – Time (seconds) before giving up logging into the guest.
- **virtio** – is a console virtio console.

Returns ShellSession object on success and None on failure.

update_vm_id ()
Update vm identifier, we need do that when force reboot vm, since vm virnet params may be changed.

verify_alive ()
Make sure the VM is alive and that the main monitor is responsive.
Can be subclassed to provide better information on why the VM is not alive (reason, detail)

Raises VMDeadError – If the VM is dead

Raise Various monitor exceptions if the monitor is unresponsive

verify_bsod (*scrdump_file*)

verify_illegal_instruction ()

Find illegal instruction code on VM serial console output.

Raise VMInvalidInstructionCode, in case a wrong instruction code.

verify_kernel_crash ()

Find kernel crash message on the VM serial console.

Raise VMDeadKernelCrashError, in case a kernel crash message was found.

verify_userspace_crash ()

Verify if the userspace component of the virtualization backend crashed.

wait_for_get_address (*args, **kwargs)

Wait for a nic to acquire an IP address, then return it. For ipv6 linklocal address, we can generate it by nic mac, so we can ignore this case

wait_for_login (*nic_index=0, timeout=240, internal_timeout=10, serial=False, restart_network=False, username=None, password=None*)

Make multiple attempts to log into the guest via SSH/Telnet/Netcat.

Parameters

- **nic_index** – The index of the NIC to connect to.
- **timeout** – Time (seconds) to keep trying to log in.
- **internal_timeout** – Timeout to pass to login().
- **serial** – Whether to use a serial connection when remote login (ssh, rss) failed.
- **restart_network** – Whether to try to restart guest's network when remote login (ssh, rss) failed.

Returns A ShellSession object.

wait_for_serial_login (*timeout=240, internal_timeout=10, restart_network=False, username=None, password=None, virtio=False*)

Make multiple attempts to log into the guest via serial console.

Parameters

- **timeout** – Time (seconds) to keep trying to log in.
- **internal_timeout** – Timeout to pass to serial_login().
- **restart_network** – Whether try to restart guest's network.

Params virtio is a virtio console.

Returns A ShellSession object.

class virttest.virt_vm.**CpuInfo** (*model=None, vendor=None, flags=None, family=None, smp=0, maxcpus=0, sockets=0, cores=0, threads=0*)

Bases: `object`

A class for VM's cpu information.

exception virttest.virt_vm.**VMAddNetDevError** (*args)

Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMAddNicError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMAddressError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMAddressVerificationError (mac, ip)`
Bases: `virttest.virt_vm.VMAddressError`

exception `virttest.virt_vm.VMBadPATypeError (pa_type)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMConfigMissingError (name, config)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMCreateError (cmd, status, output)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDeadError (reason='', detail='')`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDeadKernelCrashError (kernel_crash)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDelNetDevError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDelNicError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDeviceError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMDeviceNotSupportedError (name, device)`
Bases: `virttest.virt_vm.VMDeviceError`

exception `virttest.virt_vm.VMError (*args)`
Bases: `exceptions.Exception`

exception `virttest.virt_vm.VMHashMismatchError (actual, expected)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMHugePageError (cmd, output)`
Bases: `virttest.virt_vm.VMPostCreateError`

exception `virttest.virt_vm.VMIPAddressMissingError (mac, ip_version='ipv4')`
Bases: `virttest.virt_vm.VMAddressError`

exception `virttest.virt_vm.VMImageCheckError (filename)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMImageMissingError (filename)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMInterfaceIndexError (*args)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMInvalidInstructionCode (invalid_code)`
Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMKVMInitError (cmd, output)`
Bases: `virttest.virt_vm.VMPostCreateError`

```

exception virttest.virt_vm.VMMACAddressMissingError (nic_index)
    Bases: virttest.virt_vm.VMAddressError

exception virttest.virt_vm.VMMigrateCancelError (*args)
    Bases: virttest.virt_vm.VMMigrateError

exception virttest.virt_vm.VMMigrateError (*args)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMMigrateFailedError (*args)
    Bases: virttest.virt_vm.VMMigrateError

exception virttest.virt_vm.VMMigrateProtoUnknownError (protocol)
    Bases: autotest.client.shared.error.TestNAError

exception virttest.virt_vm.VMMigrateStateMismatchError
    Bases: virttest.virt_vm.VMMigrateError

exception virttest.virt_vm.VMMigrateTimeoutError (*args)
    Bases: virttest.virt_vm.VMMigrateError

exception virttest.virt_vm.VMPAError (pa_type)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMPCIDeviceError (*args)
    Bases: virttest.virt_vm.VMDeviceError

exception virttest.virt_vm.VMPCIOOutOfRangeError (name, max_dev_num)
    Bases: virttest.virt_vm.VMPCIDeviceError

exception virttest.virt_vm.VMPCISlotInUseError (name, slot)
    Bases: virttest.virt_vm.VMPCIDeviceError

exception virttest.virt_vm.VMPortNotRedirectedError (port, virtnet_nic=None)
    Bases: virttest.virt_vm.VMAddressError

exception virttest.virt_vm.VMPostCreateError (cmd, output)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMRebootError (*args)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMRemoveError (*args)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMScreenInactiveError (vm, inactive_time)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMStartError (name, reason=None)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMStatusError (*args)
    Bases: virttest.virt_vm.VMError

exception virttest.virt_vm.VMUSBControllerError (*args)
    Bases: virttest.virt_vm.VMUSBError

exception virttest.virt_vm.VMUSBControllerMissingError (name, controller_type)
    Bases: virttest.virt_vm.VMUSBControllerError

exception virttest.virt_vm.VMUSBControllerPortFullError (name, usb_dev_dict)
    Bases: virttest.virt_vm.VMUSBControllerError

```

exception `virttest.virt_vm.VMUSBError(*args)`

Bases: `virttest.virt_vm.VMError`

exception `virttest.virt_vm.VMUSBPortInUseError(vm_name, controller, port)`

Bases: `virttest.virt_vm.VMUSBError`

exception `virttest.virt_vm.VMUnknownNetTypeError(vmname, nicname, nettype)`

Bases: `virttest.virt_vm.VMError`

virttest.xml_utils module

Utility module standardized on ElementTree 2.6 to minimize dependencies in python 2.4 systems.

Often operations on XML files suggest making a backup copy first is a prudent measure. However, it's easy to lose track of these temporary files and they can quickly leave a mess behind. The TempXMLFile class helps by trying to clean up the temporary file whenever the instance is deleted, goes out of scope, or an exception is thrown.

The XMLBackup class extends the TempXMLFile class by basing its file-like instances off of an automatically created TempXMLFile instead of pointing at the source. Methods are provided for overwriting the backup copy from the source, or restoring the source from the backup. Similar to TempXMLFile, the temporary backup files are automatically removed. Access to the original source is provided by the sourcefilename attribute.

An interface for querying and manipulating XML data is provided by the XMLTreeFile class. Instances of this class are BOTH file-like and ElementTree-like objects. Whether or not they are constructed from a file or a string, the file-like instance always represents a temporary backup copy. Access to the source (even when itself is temporary) is provided by the sourcefilename attribute, and a (closed) file object attribute sourcebackupfile. See the ElementTree documentation for methods provided by that class.

Finally, the TemplateXML class represents XML templates that support dynamic keyword substitution based on a dictionary. Substitution keys in the XML template (string or file) follow the 'bash' variable reference style (\$foo or \${bar}). Extension of the parser is possible by subclassing TemplateXML and overriding the ParserClass class attribute. The parser class should be an ElementTree.TreeBuilder class or subclass. Instances of XMLTreeFile are returned by the parse method, which are themselves temporary backups of the parsed content. See the xml_utils_unittest module for examples.

class `virttest.xml_utils.Sub(**mapping)`

Bases: `object`

String substituter using string.Template

substitute (*text*)

Use string.safe_substitute on text and return the result

Parameters *text* – string to substitute

class `virttest.xml_utils.TempXMLFile(suffix='.xml', prefix='xml_utils_temp_', mode='wb+',
buffsz=1)`

Bases: `file`

Temporary XML file auto-removed on instance del / module exit.

unlink ()

Unconditionally delete file, ignoring related exceptions

class `virttest.xml_utils.TemplateXML(xml, **mapping)`

Bases: `virttest.xml_utils.XMLTreeFile`

Template-sourced XML ElementTree backed by temporary file.

ParserClass

alias of `TemplateXMLTreeBuilder`

parse (*source*, *parser=None*)

Parse source XML file or filename using TemplateXMLTreeBuilder

Parameters

- **source** – XML file or filename
- **parser** – ignored

restore ()

Raise an IOError to protect the original template source.

class `virttest.xml_utils.TemplateXMLTreeBuilder` (***mapping*)

Bases: `virttest.element_tree.XMLTreeBuilder`, `virttest.xml_utils.Sub`

Resolve XML templates into temporary file-backed ElementTrees

BuilderClass

alias of TreeBuilder

feed (*data*)

class `virttest.xml_utils.XMLBackup` (*sourcefilename*)

Bases: `virttest.xml_utils.TempXMLFile`

Backup file copy of XML data, automatically removed on instance destruction.

backup ()

Overwrite temporary backup with contents of original source.

restore ()

Overwrite original source with contents of temporary backup

sourcefilename = `None`

class `virttest.xml_utils.XMLTreeFile` (*xml*)

Bases: `virttest.element_tree.ElementTree`, `virttest.xml_utils.XMLBackup`

Combination of ElementTree root and auto-cleaned XML backup file.

backup ()

Overwrite original source from current tree

backup_copy ()

Return a copy of instance, including copies of files

create_by_xpath (*xpath*)

Creates all elements in simplistic xpath from root if not exist

get_element_string (*xpath*)

Returns the string for the element on xpath.

get_parent (*element*, *relative_root=None*)

Return the parent node of an element or None

param: element: Element to retrieve parent of param: relative_root: Search only below this element

get_parent_map (*element=None*)

Return a child to parent mapping dictionary

param: element: Search only below this element

get_xpath (*element*)

Return the XPath string formed from first-match tag names

read (*xml*)

remove (*element*)

Removes a matching subelement.

Parameters **element** – element to be removed.

remove_by_xpath (*xpath*, *remove_all=False*)

Remove an element found by xpath

Parameters **xpath** – element name or path to remove

reroot (*xpath*)

Return a copy of instance, re-rooted onto xpath

restore ()

Overwrite and reparse current tree from original source

sourcebackupfile = None

write (*filename=None*, *encoding='UTF-8'*)

Write current XML tree to filename, or self.name if None.

virttest.xml_utils_unittest module

class virttest.xml_utils_unittest.**test_ElementTree** (*methodName='runTest'*)

Bases: *virttest.xml_utils_unittest.xml_test_data*

test_bundled_elementtree ()

class virttest.xml_utils_unittest.**test_TempXMLFile** (*methodName='runTest'*)

Bases: *virttest.xml_utils_unittest.xml_test_data*

test_TempXMLFile_explicit ()

test_TempXMLFile_implicit ()

test_prefix_sufix ()

test_test_TempXMLFile_canread ()

class virttest.xml_utils_unittest.**test_XMLBackup** (*methodName='runTest'*)

Bases: *virttest.xml_utils_unittest.xml_test_data*

class_to_test

alias of XMLBackup

test_TempXMLBackup_exception_exit ()

test_TempXMLBackup_implicit ()

test_TempXMLBackup_unexception_exit ()

test_backup_file ()

test_backup_filename ()

test_rebackup_file ()

test_remove_backup_file ()

test_restore_file ()

class virttest.xml_utils_unittest.**test_XMLTreeFile** (*methodName='runTest'*)

Bases: *virttest.xml_utils_unittest.xml_test_data*

class_to_test

alias of XMLTreeFile


```
get_xpath_elements(target_path_string)
test_backup_backup_and_remove()
test_create_by_xpath()
test_get_xpath()
test_init_str()
test_init_xml()
test_read_other_changed()
test_restore_from_file()
test_restore_from_string()
test_sourcebackupfile_closed_file()
test_sourcebackupfile_closed_string()
test_stringify()
test_write_default()
test_write_other()
test_write_other_changed()

class virttest.xml_utils_unittest.test_templatized_xml(methodName='runTest')
    Bases: virttest.xml_utils_unittest.xml_test_data
        setUp()
        test_MappingTreeBuilder_standalone()
        test_TemplateXML()
        test_TemplateXMLTreeBuilder_nosub()
        test_restore_fails()
        test_sub()

class virttest.xml_utils_unittest.xml_test_data(methodName='runTest')
    Bases: unittest.case.TestCase
        canonicalize_test_xml()
        get_tmp_files(prefix, suffix)
        is_same_contents(filename, other=None)
            Compare filename contents with XMLSTR, or contents of other
        setUp()
        tearDown()
```

virttest.yumrepo module

This module implements classes that allow a user to create, enable and disable YUM repositories on the system.

```
class virttest.yumrepo.YumRepo(name, baseurl, path=None)
    Bases: object
    Represents a YUM repository
```

The goal of this class is not to give access to all features of a YUM Repository, but to provide a simple way to configure a valid one during a test run.

Sample usage:

```
>>> mainrepo = YumRepo("main", "http://download.project.org/repo",
                        "/etc/yum.repos.d/main.repo")
```

Or to use a default path:

```
>>> mainrepo = YumRepo("main", 'http://download.project.org/repo')
```

And then:

```
>>> mainrepo.save()
```

When it comes to the repo URL, currently there's no support for setting a mirrorlist, only a baseurl.

remove()

Removes the repo file

render()

Renders the repo file

Yes, we could use ConfigParser for this, but it produces files with spaces between keys and values, which look akward by YUM defaults.

save()

Saves the repo file

Module contents

Indices and tables

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

V

virttest, 566
virttest.aexpect, 280
virttest.arch, 289
virttest.asset, 289
virttest.base_installer, 291
virttest.bootstrap, 293
virttest.build_helper, 294
virttest.cartesian_config, 298
virttest.cartesian_config_unittest, 307
virttest.ceph, 308
virttest.common, 308
virttest.data_dir, 308
virttest.defaults, 309
virttest.element_path, 309
virttest.element_tree, 309
virttest.env_process, 310
virttest.funcatexit, 312
virttest.gluster, 313
virttest.guest_agent, 313
virttest.http_server, 316
virttest.installer, 317
virttest.installer_unittest, 318
virttest.iscsi, 318
virttest.iscsi_unittest, 320
virttest.libvirt_network_unittest, 320
virttest.libvirt_storage, 321
virttest.libvirt_storage_unittest, 323
virttest.libvirt_vm, 324
virttest.libvirt_xml, 212
virttest.libvirt_xml.accessors, 169
virttest.libvirt_xml.base, 175
virttest.libvirt_xml.capability_xml, 177
virttest.libvirt_xml.devices, 138
virttest.libvirt_xml.devices.address, 121
virttest.libvirt_xml.devices.base, 122
virttest.libvirt_xml.devices.channel, 122
virttest.libvirt_xml.devices.character, 123
virttest.libvirt_xml.devices.console, 124
virttest.libvirt_xml.devices.controller, 124
virttest.libvirt_xml.devices.disk, 125
virttest.libvirt_xml.devices.emulator, 128
virttest.libvirt_xml.devices.filesystem, 128
virttest.libvirt_xml.devices.graphics, 128
virttest.libvirt_xml.devices.hostdev, 130
virttest.libvirt_xml.devices.hub, 130
virttest.libvirt_xml.devices.input, 131
virttest.libvirt_xml.devices.interface, 131
virttest.libvirt_xml.devices.lease, 133
virttest.libvirt_xml.devices.librarian, 133
virttest.libvirt_xml.devices.memballoon, 134
virttest.libvirt_xml.devices.memory, 134
virttest.libvirt_xml.devices.panic, 135
virttest.libvirt_xml.devices.parallel, 135
virttest.libvirt_xml.devices.redirdev, 135
virttest.libvirt_xml.devices.rng, 136
virttest.libvirt_xml.devices.seclabel, 136
virttest.libvirt_xml.devices.serial, 137
virttest.libvirt_xml.devices.smartcard, 137
virttest.libvirt_xml.devices.sound, 138
virttest.libvirt_xml.devices.video, 138
virttest.libvirt_xml.devices.watchdog, 138
virttest.libvirt_xml.network_xml, 179
virttest.libvirt_xml.nodedev_xml, 185

virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.nwfilter_xml, 188
169
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.pool_xml, 191
139
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.secret_xml, 194
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.snapshot_xml, 195
140
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.sysinfo_xml, 197
141
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.vm_xml, 197
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.vol_xml, 210
142
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml.xcepts, 212
virttest.libvirt_xml.nwfilter_protocols, virttest.libvirt_xml_unittest, 332
143
virttest.libvirt_xml.nwfilter_protocols, virttest.lvm, 334
virttest.libvirt_xml.nwfilter_protocols, virttest.lvusb, 338
144
virttest.libvirt_xml.nwfilter_protocols, virttest.lvusb_base, 339
virttest.libvirt_xml.nwfilter_protocols, virttest.lvusb, 341
145
virttest.libvirt_xml.nwfilter_protocols, virttest.nfs, 342
virttest.libvirt_xml.nwfilter_protocols, virttest.nfs_unittest, 343
145
virttest.libvirt_xml.nwfilter_protocols, virttest.openvswitch, 344
virttest.libvirt_xml.nwfilter_protocols, virttest.ovirt, 346
147
virttest.libvirt_xml.nwfilter_protocols, virttest.ovs_utils, 350
virttest.libvirt_xml.nwfilter_protocols, virttest.passfd_setup, 351
148
virttest.libvirt_xml.nwfilter_protocols, virttest.postprocess_iozone, 351
virttest.libvirt_xml.nwfilter_protocols, virttest.ppm_utils, 353
149
virttest.libvirt_xml.nwfilter_protocols, virttest.propcan, 356
virttest.libvirt_xml.nwfilter_protocols, virttest.propcan_unittest, 358
150
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices, 224
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices.qbuses, 213
151
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices.qcontainer, 215
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices.qdevices, 219
153
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices.utils, 223
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_devices_unittest, 358
154
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_installer, 359
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_io, 360
154
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_monitor, 361
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_monitor_unittest, 373
155
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_qtree, 374
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_qtree_unittest, 376
156
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_storage, 377
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_virtio_port, 379
157
virttest.libvirt_xml.nwfilter_protocols, virttest.qemu_vm, 381
virttest.libvirt_xml.nwfilter_protocols, virttest.remote, 391
158
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_build, 397
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_commander, 231
160
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_commander.messenger, 224
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_commander.remote_interface, 226
161
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_commander.remote_master, 227
163
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_commander.remote_runner, 229
164
virttest.libvirt_xml.nwfilter_protocols, virttest.remote_unittest, 397
165
virttest.libvirt_xml.nwfilter_protocols, virttest.RFBDes, 279
virttest.libvirt_xml.nwfilter_protocols, virttest.rfbdes_client, 398
166
virttest.libvirt_xml.nwfilter_protocols, virttest.scheduler, 400
virttest.libvirt_xml.nwfilter_protocols, virttest.service_unittest, 400
168
virttest.staging, 257

```

virttest.staging.backports, 243
virttest.staging.backports.collections, 233
virttest.staging.backports.collections.defaultdict, 505
virttest.staging.backports.collections.namedtuple, 506
virttest.staging.backports.collections.OrderedDict, 508
virttest.staging.backports.simplejson, 236
virttest.staging.backports.simplejson.decoder, 234
virttest.staging.backports.simplejson.ordered_dict, 235
virttest.staging.backports.simplejson.scanner, 236
virttest.staging.lv_utils, 243
virttest.staging.service, 244
virttest.staging.utils_cgroup, 247
virttest.staging.utils_koji, 251
virttest.staging.utils_memory, 256
virttest.standalone_test, 401
virttest.storage, 403
virttest.syslog_server, 405
virttest.test_setup, 407
virttest.tests, 258
virttest.tests.unattended_install, 257
virttest.utils_cgroup_unittest, 412
virttest.utils_config, 412
virttest.utils_config_unittest, 414
virttest.utils_conn, 415
virttest.utils_disk, 420
virttest.utils_env, 422
virttest.utils_env_unittest, 424
virttest.utils_gdb, 426
virttest.utils_libguestfs, 427
virttest.utils_libguestfs_unittest, 462
virttest.utils_libvirtd, 462
virttest.utils_misc, 463
virttest.utils_misc_unittest, 479
virttest.utils_net, 479
virttest.utils_net_unittest, 492
virttest.utils_netperf, 494
virttest.utils_params, 496
virttest.utils_params_unittest, 496
virttest.utils_sasl, 496
virttest.utils_selinux, 497
virttest.utils_spice, 500
virttest.utils_test, 274
virttest.utils_test.libguestfs, 259
virttest.utils_test.libvirt, 261
virttest.utils_test.qemu, 269
virttest.utils_v2v, 502
virttest.utils_virtio_port, 504
virttest.versionable_class, 505
virttest.versionable_class_unittest, 506
virttest.video_maker, 508
virttest.virsh, 508
virttest.virsh_unittest, 551
virttest.virt_vm, 553
virttest.xml_utils, 562
virttest.xml_utils_unittest, 564
virttest.yumrepo, 565

```


A

- AA (class in virttest.versionable_class_unittest), 506
- acceleration (virttest.libvirt_xml.devices.video.Video attribute), 138
- access_drivers (virttest.utils_conn.UNIXConnection attribute), 419
- accessor_name() (virttest.libvirt_xml.accessors.AccessorGeneratorBase method), 169
- AccessorBase (class in virttest.libvirt_xml.accessors), 169
- AccessorGeneratorBase (class in virttest.libvirt_xml.accessors), 169
- AccessorsTest (class in virttest.libvirt_xml_unittest), 332
- ACQUIRE_LOCK_TIMEOUT (virttest.qemu_monitor.Monitor attribute), 365
- action (virttest.libvirt_xml.devices.watchdog.Watchdog attribute), 138
- action_after_suspend() (virttest.utils_test.qemu.GuestSuspend method), 269
- action_before_suspend() (virttest.utils_test.qemu.GuestSuspend method), 269
- action_during_suspend() (virttest.utils_test.qemu.GuestSuspend method), 269
- activate_netdev() (virttest.qemu_vm.VM method), 381
- activate_nic() (virttest.libvirt_vm.VM method), 324
- activate_nic() (virttest.qemu_vm.VM method), 381
- activate_nic() (virttest.virt_vm.BaseVM method), 554
- active (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 183
- add() (virttest.ovirt.ClusterManager method), 346
- add() (virttest.ovirt.DataCenterManager method), 346
- add() (virttest.ovirt.HostManager method), 347
- add() (virttest.ovirt.VMManager method), 347
- add() (virttest.remote.RemoteFile method), 391
- add_br() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 344
- add_br() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 344
- add_bridge() (virttest.utils_net.Bridge method), 479
- add_channel() (virttest.libvirt_xml.devices.graphics.Graphics method), 128
- add_chap_account() (virttest.iscsi.IscsiTGT method), 319
- add_child() (virttest.qemu_qtree.QtreeBus method), 374
- add_child() (virttest.qemu_qtree.QtreeDev method), 374
- add_child() (virttest.qemu_qtree.QtreeNode method), 375
- add_child_bus() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- add_device() (virttest.libvirt_xml.vm_xml.VMXML method), 202
- add_device() (virttest.test_setup.PciAssignable method), 409
- add_device_to_iommu_group() (virttest.utils_misc.VFIOController method), 466
- add_domain() (virttest.utils_libguestfs.GuestfishPersistent method), 427
- add_downstream_port() (virttest.qemu_devices.qbuses.QPCISwitchBus method), 214
- add_drive() (virttest.utils_libguestfs.GuestfishPersistent method), 427
- add_drive_opts() (virttest.utils_libguestfs.GuestfishPersistent method), 427
- add_drive_ro() (virttest.utils_libguestfs.GuestfishPersistent method), 427
- add_drive_ro_with_if() (virttest.utils_libguestfs.GuestfishPersistent method), 428
- add_drive_scratch() (virttest.utils_libguestfs.GuestfishPersistent method), 428
- add_drive_with_if() (virttest.utils_libguestfs.GuestfishPersistent method), 428
- add_fake_br() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 344
- add_feature() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 177
- add_feature() (virttest.libvirt_xml.vm_xml.VMCPUXML method), 198
- add_feature() (virttest.libvirt_xml.vm_xml.VMFeaturesXML method), 200
- add_flag() (virttest.lvsb_base.SandboxCommandBase

- method), 340
- add_function() (virttest.remote_commander.remote_runner.CommanderSlaveControlLibvirtXml.devices.panic.Panic attribute), 135
- method), 230
- add_graphic() (virttest.libvirt_xml.devices.graphics.Graphics attribute), 135
- static method), 128
- add_hostdev() (virttest.libvirt_xml.vm_xml.VMXML attribute), 135
- method), 202
- add_identities_into_ssh_agent() (in module virttest.libvirt_xml.devices.address), 121
- virttest.utils_misc), 467
- add_ker_cmd() (in module virttest.utils_misc), 467
- add_listens() (virttest.libvirt_xml.devices.graphics.Graphics attribute), 123
- method), 129
- add_mm() (virttest.lvsb_base.SandboxCommandBase attribute), 124
- method), 340
- add_netdev() (virttest.qemu_vm.VM method), 381
- add_nic() (virttest.qemu_vm.VM method), 382
- add_nic() (virttest.virt_vm.BaseVM method), 554
- add_optarg() (virttest.lvsb_base.SandboxCommandBase attribute), 127
- method), 340
- add_ovs_bridge() (in module virttest.utils_net), 486
- add_port() (virttest.openvswitch.OpenVSwitchControl attribute), 131
- method), 344
- add_port() (virttest.openvswitch.OpenVSwitchControlCli attribute), 133
- method), 344
- add_port() (virttest.utils_net.Bridge method), 479
- add_port_tag() (virttest.openvswitch.OpenVSwitchControl attribute), 135
- method), 344
- add_port_tag() (virttest.openvswitch.OpenVSwitchControlCli attribute), 137
- method), 345
- add_port_trunk() (virttest.openvswitch.OpenVSwitchControl attribute), 138
- method), 344
- add_port_trunk() (virttest.openvswitch.OpenVSwitchControlCli attribute), 138
- method), 345
- add_pos() (virttest.lvsb_base.SandboxCommandBase attribute), 180
- method), 340
- add_rpc_insecure() (in module virttest.gluster), 313
- add_rule() (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 195
- method), 190
- add_security_info() (virttest.libvirt_xml.vm_xml.VMXML attribute), 137
- static method), 203
- add_source() (virttest.libvirt_xml.devices.character.CharacterBase attribute), 137
- method), 123
- add_target() (virttest.libvirt_xml.devices.character.CharacterBase attribute), 137
- method), 123
- add_to_bridge() (in module virttest.utils_net), 486
- add_to_slots() (in module virttest.libvirt_xml.accessors), 175
- add_vlan_iface() (virttest.ovs_utils.Machine method), 350
- add_vm_from_template() (virttest.ovirt.VMManager attribute), 199
- method), 348
- addr_bus (virttest.libvirt_xml.devices.panic.Panic attribute), 193
- addr_controller (virttest.libvirt_xml.devices.panic.Panic attribute), 193
- addr_port (virttest.libvirt_xml.devices.panic.Panic attribute), 193
- addr_type (virttest.libvirt_xml.devices.panic.Panic attribute), 193
- Address (class in virttest.libvirt_xml.devices.address), 121
- address (virttest.libvirt_xml.devices.channel.Channel attribute), 123
- address (virttest.libvirt_xml.devices.controller.Controller attribute), 124
- address (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- address (virttest.libvirt_xml.devices.hub.Hub attribute), 131
- address (virttest.libvirt_xml.devices.input.Input attribute), 131
- address (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- address (virttest.libvirt_xml.devices.memory.Memory attribute), 135
- address (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- address (virttest.libvirt_xml.devices.sound.Sound attribute), 138
- address (virttest.libvirt_xml.devices.video.Video attribute), 138
- address (virttest.libvirt_xml.devices.watchdog.Watchdog attribute), 138
- address (virttest.libvirt_xml.network_xml.IPXML attribute), 180
- address (virttest.libvirt_xml.nodedev_xml.NetXML attribute), 185
- address (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- address_controller (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- address_slot (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- address_string() (virttest.http_server.HTTPRequestHandler method), 316
- address_type (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- adjustment (virttest.libvirt_xml.vm_xml.VMClockXML attribute), 199
- adp_name (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- adp_parent (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- adp_type (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- adp_wwnn (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193

attribute), 193
 adp_wwpn (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
 AexpectIOWrapperOut (class in virttest.remote), 391
 Ah (class in virttest.libvirt_xml.nwfilter_protocols.ah), 139
 Ah.Attr (class in virttest.libvirt_xml.nwfilter_protocols.ah), 139
 Ah_ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.ah_ipv6), 140
 Ah_ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.ah_ipv6), 140
 alias (virttest.libvirt_xml.devices.channel.Channel attribute), 123
 aliases (virttest.utils_misc.Flag attribute), 463
 All (class in virttest.libvirt_xml.nwfilter_protocols.all), 141
 all() (in module virttest.staging.backports), 243
 All.Attr (class in virttest.libvirt_xml.nwfilter_protocols.all), 141
 all_cgroup_delete() (in module virttest.staging.utils_cgroup), 250
 All_ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.all_ipv6), 142
 All_ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.all_ipv6), 142
 AllForbidden (class in virttest.libvirt_xml.accessors), 169
 alloc() (virttest.utils_libguestfs.GuestfishPersistent method), 428
 allocation (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
 allocation (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 211
 alter_boot_order() (in module virttest.utils_test.libvirt), 262
 analyse_release() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 analyze() (virttest.postprocess_iozone.IOzoneAnalyzer method), 352
 AnalyzerLoggingConfig (class in virttest.postprocess_iozone), 351
 answer_kickstart() (virttest.tests.unattended_install.UnattendedInstallConfig method), 257
 answer_suse_xml() (virttest.tests.unattended_install.UnattendedInstallConfig method), 257
 answer_windows_ini() (virttest.tests.unattended_install.UnattendedInstallConfig method), 257
 answer_windows_xml() (virttest.tests.unattended_install.UnattendedInstallConfig method), 257
 any() (in module virttest.staging.backports), 243
 app_running() (virttest.utils_test.HostStress method), 275
 app_running() (virttest.utils_test.VMStress method), 276
 append() (virttest.libvirt_xml.nwfilter_xml.NwfilterRulesProtocol method), 188
 append() (virttest.libvirt_xml.vm_xml.VMXMLDevices method), 210
 append() (virttest.utils_net.VMNet method), 484
 append_lv() (virttest.lvm.VolumeGroup method), 337
 append_to_element() (virttest.libvirt_xml.network_xml.RangeList method), 185
 append_to_shortcode() (virttest.cartesian_config.Node attribute), 305
 apply_defcon() (in module virttest.utils_selinux), 498
 apply_predict() (in module virttest.cartesian_config), 307
 apply_to_dict() (virttest.cartesian_config.BlockFilter method), 300
 apply_to_dict() (virttest.cartesian_config.LAppend method), 301
 apply_to_dict() (virttest.cartesian_config.LApplyPreDict method), 301
 apply_to_dict() (virttest.cartesian_config.LDel method), 301
 apply_to_dict() (virttest.cartesian_config.LPrepend method), 303
 apply_to_dict() (virttest.cartesian_config.LRegExpAppend method), 303
 apply_to_dict() (virttest.cartesian_config.LRegExpPrepend method), 303
 apply_to_dict() (virttest.cartesian_config.LRegExpSet method), 303
 apply_to_dict() (virttest.cartesian_config.LSet method), 303
 apply_to_dict() (virttest.cartesian_config.LUpdateFileMap method), 304
 arch (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 177
 arch (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 201
 archive_as_tarball() (in module virttest.utils_misc), 467
 are_failed() (virttest.lvsb_base.TestSandboxes method), 341
 are_running() (virttest.lvsb_base.TestSandboxes method), 341
 args (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
 arp (virttest.libvirt_xml.nwfilter_protocols.arp), 143
 Arp (class in virttest.libvirt_xml.nwfilter_protocols.arp), 144
 Arp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.arp), 144
 Arp.Host (class in virttest.libvirt_xml.nwfilter_protocols.arp), 155
 arpdstmacaddr (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144
 arpdstmacaddr (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 144

attribute), 155

arpsrcipaddr (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144

arpsrcipaddr (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155

arpsrcmacaddr (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144

arpsrcmacaddr (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155

assign() (virttest.cartesian_config.Parser method), 306

assign_callable() (virttest.libvirt_xml.accessors.AccessorGenerator method), 169

aton() (in module virttest.utils_misc), 467

attach_additional_device() (in module virttest.utils_test.libvirt), 262

attach_additional_disk() (in module virttest.utils_test.libguestfs), 260

attach_device() (in module virttest.virsh), 511

attach_disk() (in module virttest.virsh), 511

attach_disk() (virttest.libvirt_vm.VM method), 324

attach_disks() (in module virttest.utils_test.libvirt), 263

attach_interface() (in module virttest.virsh), 512

attach_interface() (virttest.libvirt_vm.VM method), 324

attach_iso_export_domain_into_datacenter() (virttest.ovirt.StorageDomainManager method), 347

attribute (virttest.libvirt_xml.accessors.XMLAttribute.Deleter attribute), 170

attribute (virttest.libvirt_xml.accessors.XMLAttribute.Getter attribute), 170

attribute (virttest.libvirt_xml.accessors.XMLAttribute.Setter attribute), 170

attrs (virttest.libvirt_xml.devices.address.Address attribute), 121

attrs (virttest.libvirt_xml.devices.controller.Controller.Address attribute), 124

attrs (virttest.libvirt_xml.devices.disk.Disk.Address attribute), 125

attrs (virttest.libvirt_xml.devices.disk.Disk.DiskSource attribute), 126

attrs (virttest.libvirt_xml.devices.hub.Hub.Address attribute), 131

attrs (virttest.libvirt_xml.devices.input.Input.Address attribute), 131

attrs (virttest.libvirt_xml.devices.interface.Interface.Address attribute), 132

attrs (virttest.libvirt_xml.devices.memory.Memory.Address attribute), 134

attrs (virttest.libvirt_xml.nwfilter_protocols.ah.Ah attribute), 139

attrs (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6 attribute), 141

attrs (virttest.libvirt_xml.nwfilter_protocols.all.All attribute), 142

attrs (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6 attribute), 143

attrs (virttest.libvirt_xml.nwfilter_protocols.arp.Arp attribute), 144

attrs (virttest.libvirt_xml.nwfilter_protocols.esp.Esp attribute), 146

attrs (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6 attribute), 147

attrs (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp attribute), 149

attrs (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6 attribute), 150

attrs (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp attribute), 151

attrs (virttest.libvirt_xml.nwfilter_protocols.ip.Ip attribute), 152

attrs (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6 attribute), 154

attrs (virttest.libvirt_xml.nwfilter_protocols.mac.Mac attribute), 155

attrs (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp attribute), 156

attrs (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp attribute), 157

attrs (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6 attribute), 158

attrs (virttest.libvirt_xml.nwfilter_protocols.stp.Stp attribute), 160

attrs (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp attribute), 161

attrs (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6 attribute), 162

attrs (virttest.libvirt_xml.nwfilter_protocols.udp.Udp attribute), 164

attrs (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6 attribute), 165

attrs (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite attribute), 166

attrs (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6 attribute), 167

attrs (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan attribute), 168

aug_clear() (virttest.utils_libguestfs.GuestfishPersistent method), 428

aug_close() (virttest.utils_libguestfs.GuestfishPersistent method), 428

aug_defnode() (virttest.utils_libguestfs.GuestfishPersistent method), 428

aug_defvar() (virttest.utils_libguestfs.GuestfishPersistent method), 428

aug_get() (virttest.utils_libguestfs.GuestfishPersistent method), 428

aug_init() (virttest.utils_libguestfs.GuestfishPersistent method), 428

- `aug_insert()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_label()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_load()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_ls()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_match()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_mv()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_rm()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_save()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_set()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `aug_setm()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `auth` (`virttest.libvirt_xml.devices.disk.Disk` attribute), 127
 - `auth` (`virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML` attribute), 195
 - `auth_tcp` (`virttest.utils_conn.TCPConnection` attribute), 417
 - `auth_tls` (`virttest.utils_conn.TLSConnection` attribute), 418
 - `auth_type` (`virttest.libvirt_xml.pool_xml.SourceXML` attribute), 193
 - `auth_type` (`virttest.libvirt_xml.secret_xml.SecretXMLBase` attribute), 195
 - `auth_unix_ro` (`virttest.utils_conn.UNIXConnection` attribute), 419
 - `auth_unix_rw` (`virttest.utils_conn.UNIXConnection` attribute), 419
 - `auth_user` (`virttest.libvirt_xml.devices.disk.Disk.Auth` attribute), 126
 - `auth_username` (`virttest.libvirt_xml.pool_xml.SourceXML` attribute), 193
 - `auth_username` (`virttest.libvirt_xml.secret_xml.SecretXMLBase` attribute), 195
 - `auto_clean()` (`virttest.lvsb_base.SandboxBase` method), 339
 - `auto_clean()` (`virttest.lvsb_base.SandboxSession` method), 340
 - `auto_recover` (`virttest.utils_conn.ConnectionBase` attribute), 416
 - `auto_recover` (`virttest.utils_net.IPv6Manager` attribute), 481
 - `auto_recover` (`virttest.utils_sasl.SASL` attribute), 497
 - `autoport` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129
 - `autostart` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` attribute), 183
 - `autostart()` (in module `virttest.virsh`), 512
 - `available` (`virttest.libvirt_xml.pool_xml.PoolXMLBase` attribute), 193
 - `available()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 429
 - `available_all_groups()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 430
 - `average_performance()` (`virttest.postprocess_iozone.IOzoneAnalyzer` method), 352
- ## B
- `back_trace()` (`virttest.utils_gdb.GDB` method), 426
 - `back_trace()` (`virttest.utils_libvirtd.LibvirtdSession` method), 462
 - `backend` (`virttest.libvirt_xml.devices.interface.Interface` attribute), 133
 - `backend` (`virttest.libvirt_xml.devices.rng.Rng` attribute), 136
 - `backend_dev` (`virttest.libvirt_xml.devices.rng.Rng.Backend` attribute), 136
 - `backend_model` (`virttest.libvirt_xml.devices.rng.Rng.Backend` attribute), 136
 - `backend_protocol` (`virttest.libvirt_xml.devices.rng.Rng.Backend` attribute), 136
 - `backend_type` (`virttest.libvirt_xml.devices.rng.Rng.Backend` attribute), 136
 - `BackgroundTest` (class in `virttest.utils_test`), 274
 - `backup()` (`virttest.xml_utils.XMLBackup` method), 563
 - `backup()` (`virttest.xml_utils.XMLTreeFile` method), 563
 - `backup_copy()` (`virttest.xml_utils.XMLTreeFile` method), 563
 - `backup_image()` (`virttest.storage.QemuImg` method), 403
 - `backup_rule()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules` method), 191
 - `backup_xml()` (`virttest.libvirt_vm.VM` method), 324
 - `backup_xml()` (`virttest.libvirt_xml.pool_xml.PoolXML` static method), 191
 - `bandwidth` (`virttest.libvirt_xml.devices.interface.Interface` attribute), 133
 - `bandwidth_inbound` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` attribute), 183
 - `bandwidth_inbound` (`virttest.libvirt_xml.network_xml.PortgroupXML` attribute), 185
 - `bandwidth_outbound` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` attribute), 183
 - `bandwidth_outbound` (`virttest.libvirt_xml.network_xml.PortgroupXML` attribute), 185
 - `Bar` (class in `virttest.libvirt_xml_unittest`), 332
 - `BaseCmd` (class in `virttest.remote_commander.remote_interface`), 226
 - `basecmd` (`virttest.remote_commander.remote_master.CmdMaster` attribute), 227
 - `BaseInstaller` (class in `virttest.base_installer`), 291

- baselabel (virttest.libvirt_xml.devices.seclabel.Seclabel attribute), 136
- BaseLocalSourceInstaller (class in virttest.base_installer), 291
- BaseVM (class in virttest.virt_vm), 553
- Baz (class in virttest.libvirt_xml_unittest), 332
- baz (virttest.libvirt_xml_unittest.Bar attribute), 332
- BB (class in virttest.versionable_class_unittest), 506
- Bcolors (class in virttest.standalone_test), 401
- before_migration() (virttest.utils_test.qemu.MultihostMigration method), 271
- bg_start() (virttest.utils_netperf.NetperfClient method), 494
- bin() (in module virttest.staging.backports), 243
- BINARY_PATH_PARAM (virttest.lvsb_base.SandboxCommandBase attribute), 340
- bind_device_driver() (in module virttest.utils_misc), 468
- bind_device_to_iommu_group() (virttest.utils_misc.VFIOController method), 467
- bios_reboot_timeout (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 201
- bios_useserial (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- bitlist_to_string() (in module virttest.utils_misc), 468
- blkdeviotune() (in module virttest.virsh), 512
- blkid() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blkiotune() (in module virttest.virsh), 512
- block (virttest.libvirt_xml.nodedev_xml.StorageXML attribute), 188
- block_mirror() (virttest.qemu_monitor.HumanMonitor method), 361
- block_mirror() (virttest.qemu_monitor.QMPMonitor method), 367
- block_mirror() (virttest.qemu_vm.VM method), 382
- block_reopen() (virttest.qemu_monitor.HumanMonitor method), 361
- block_reopen() (virttest.qemu_monitor.QMPMonitor method), 367
- block_reopen() (virttest.qemu_vm.VM method), 382
- block_resize() (virttest.qemu_monitor.HumanMonitor method), 362
- block_resize() (virttest.qemu_monitor.QMPMonitor method), 367
- block_stream() (virttest.qemu_monitor.HumanMonitor method), 362
- block_stream() (virttest.qemu_monitor.QMPMonitor method), 368
- block_stream() (virttest.qemu_vm.VM method), 382
- blockcommit() (in module virttest.virsh), 512
- blockcopy() (in module virttest.virsh), 512
- blockdev_flushbufs() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_getbsz() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_getro() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_getsize64() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_getss() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_getsz() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_rereadpt() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_setbsz() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_setro() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blockdev_setrw() (virttest.utils_libguestfs.GuestfishPersistent method), 430
- blocked (virttest.cartesian_config.BlockFilter attribute), 300
- BlockFilter (class in virttest.cartesian_config), 300
- blockio (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- blockio (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- blockjob() (in module virttest.virsh), 512
- blockpull() (in module virttest.virsh), 513
- blockresize() (in module virttest.virsh), 513
- bogusVirshFailureException, 553
- boot (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- boot (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- boot_order (virttest.libvirt_xml.devices.hostdev.Hostdev attribute), 130
- boot_order (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- bootloader (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- bootloader_args (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- bootmenu_enable (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- boots (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- bootstrap() (in module virttest.bootstrap), 293
- bootstrap_tests() (in module virttest.standalone_test), 401
- br_exist() (virttest.openvswitch.OpenVSwitchControl method), 344
- br_exist() (virttest.openvswitch.OpenVSwitchControlCli method), 345
- BRAddIfError, 479
- BRDelIfError, 479
- Bridge (class in virttest.utils_net), 479

- bridge (virttest.libvirt_xml.network_xml.NetworkXMLBase.canonicalize_test_xml() (virttest.xml_utils_unittest.xml_test_data attribute), 183
 - bring_down_ifname() (in module virttest.utils_net), 486
 - bring_iface_down() (virttest.ovs_utils.Machine method), 350
 - bring_iface_up() (virttest.ovs_utils.Machine method), 350
 - bring_up_ifname() (in module virttest.utils_net), 486
 - BRIPError, 479
 - BRNotExistError, 479
 - build() (virttest.remote_build.Builder method), 397
 - build_CA() (in module virttest.utils_conn), 420
 - build_client_key() (in module virttest.utils_conn), 420
 - build_pool() (virttest.libvirt_storage.StoragePool method), 322
 - build_server_key() (in module virttest.utils_conn), 420
 - Builder (class in virttest.remote_build), 397
 - BuilderClass (virttest.xml_utils.TemplateXMLTreeBuilder attribute), 563
 - BuildError, 397
 - bus (virttest.libvirt_xml.devices.hostdev.Hostdev.SourceAddress attribute), 130
 - bus (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
 - bus (virttest.libvirt_xml.nodedev_xml.PCIXML.Address attribute), 187
 - bus (virttest.libvirt_xml.nodedev_xml.StorageXML attribute), 188
 - Buses (class in virttest.qemu_devices_unittest), 358
 - by_device_tag() (virttest.libvirt_xml.nwfilter_xml.NwfilterRuleset attribute), 188
 - by_device_tag() (virttest.libvirt_xml.vm_xml.VMXMLElement method), 210
- ## C
- ca_cakey_path (virttest.utils_conn.TLSConnection attribute), 418
 - ca_cn (virttest.utils_conn.TLSConnection attribute), 418
 - cal_hamming_distance() (in module virttest.ppm_utils), 353
 - callable_name() (virttest.libvirt_xml.accessors.AccessorGeneratorBase static method), 169
 - cancel_block_job() (virttest.qemu_monitor.HumanMonitor method), 362
 - cancel_block_job() (virttest.qemu_monitor.QMPMonitor method), 368
 - cancel_block_job() (virttest.qemu_vm.VM method), 382
 - canonical_device_name() (virttest.utils_libguestfs.GuestfishPersistent method), 431
 - canonical_uri() (in module virttest.virsh), 513
 - canonicalize() (virttest.utils_net.IPAddress method), 481
 - canonicalize_disk_address() (in module virttest.utils_test), 276
 - cap (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
 - cap_type (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
 - capabilities() (in module virttest.virsh), 513
 - CapabilityXML (class in virttest.libvirt_xml.capability_xml), 177
 - capacity (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
 - capacity (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 211
 - capacity_unit (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 211
 - CAPXML (class in virttest.libvirt_xml.nodedev_xml), 185
 - CartesianConfigTest (class in virttest.cartesian_config_unittest), 307
 - case_sensitive_path() (virttest.utils_libguestfs.GuestfishPersistent method), 431
 - cat() (virttest.utils_libguestfs.GuestfishPersistent method), 431
 - cat() (virttest.utils_test.libguestfs.VirtTools method), 259
 - catch_monitor (virttest.qemu_vm.VM attribute), 383
 - catchup_limit (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199
 - catchup_slew (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199
 - catchup_threshold (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199
 - cd (in module virttest.virsh), 513
 - CdromDisk (class in virttest.utils_disk), 420
 - CdromInstallDisk (class in virttest.utils_disk), 420
 - cdroms_define_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 215
 - cell (virttest.libvirt_xml.capability_xml.TopologyXML attribute), 179
 - cell_id (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
 - cells_topology (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 177
 - CellXML (class in virttest.libvirt_xml.capability_xml), 178
 - CephError, 308
 - cert_recover() (virttest.utils_conn.TLSConnection method), 418
 - CERTTOOL (virttest.utils_conn.TLSConnection attribute), 418
 - cgclassify_cgroup() (virttest.staging.utils_cgroup.Cgroup method), 248
 - cgconfig_condrestart() (virttest.staging.utils_cgroup.CgconfigService method), 247

`cgconfig_is_running()` (`virttest.staging.utils_cgroup.CgconfigService` method), 248

`cgconfig_restart()` (`virttest.staging.utils_cgroup.CgconfigService` method), 248

`cgconfig_start()` (`virttest.staging.utils_cgroup.CgconfigService` method), 248

`cgconfig_stop()` (`virttest.staging.utils_cgroup.CgconfigService` method), 248

`CgconfigService` (class in `virttest.staging.utils_cgroup`), 247

`cgdelete_all_cgroups()` (`virttest.staging.utils_cgroup.Cgroup` method), 248

`cgdelete_cgroup()` (`virttest.staging.utils_cgroup.Cgroup` method), 248

`cgexec()` (`virttest.staging.utils_cgroup.Cgroup` method), 248

`Cgroup` (class in `virttest.staging.utils_cgroup`), 248

`CgroupModules` (class in `virttest.staging.utils_cgroup`), 250

`CgroupTest` (class in `virttest.utils_cgroup_unittest`), 412

`cgset_property()` (`virttest.staging.utils_cgroup.Cgroup` method), 248

`change_graphic_type_passwd()` (`virttest.libvirt_xml.devices.graphics.Graphics` static method), 129

`change_iface_bridge()` (in module `virttest.utils_net`), 486

`change_media()` (in module `virttest.virsh`), 514

`change_media()` (`virttest.qemu_monitor.HumanMonitor` method), 362

`change_media()` (`virttest.qemu_monitor.QMPMonitor` method), 368

`change_media()` (`virttest.qemu_vm.VM` method), 383

`Channel` (class in `virttest.libvirt_xml.devices.channel`), 122

`channel` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129

`CharacterBase` (class in `virttest.libvirt_xml.devices.character`), 123

`check()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`check_activated_pool()` (in module `virttest.utils_test.libvirt`), 263

`check_add_dnsmasq_to_br()` (in module `virttest.utils_net`), 486

`check_bg_program()` (`virttest.utils_test.qemu.GuestSuspend` method), 269

`check_block_locked()` (`virttest.qemu_vm.VM` method), 383

`check_blockjob()` (in module `virttest.utils_test.libvirt`), 263

`check_connectivity()` (`virttest.utils_net.IPv6Manager` static method), 481

`check_cpu_mode()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 203

`check_daemon()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`check_db_file()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`check_db_socket()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`check_dest_vm_network()` (in module `virttest.utils_test`), 276

`check_device_driver()` (in module `virttest.utils_misc`), 468

`check_disk_exist()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 203

`check_disk_params()` (`virttest.qemu_qtree.QtreeDisksContainer` method), 375

`check_disk_type()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 203

`check_exit_status()` (in module `virttest.utils_test.libvirt`), 263

`check_feature_name()` (`virttest.libvirt_xml.capability_xml.CapabilityXML` method), 177

`check_feature_name()` (`virttest.libvirt_xml.vm_xml.VMCPXML` static method), 198

`check_guests_proc_scsi()` (`virttest.qemu_qtree.QtreeDisksContainer` method), 375

`check_if_vm_vcpu_match()` (in module `virttest.utils_misc`), 468

`check_iface()` (in module `virttest.utils_test.libvirt`), 263

`check_image()` (`virttest.libvirt_storage.QemuImg` method), 321

`check_image()` (`virttest.qemu_storage.QemuImg` method), 377

`check_iommu()` (`virttest.utils_misc.VFIOController` method), 467

`check_ipv6_connectivity` (`virttest.utils_net.IPv6Manager` attribute), 481

`check_listening_port_by_service()` (in module `virttest.utils_net`), 486

`check_listening_port_remote_by_service()` (in module `virttest.utils_net`), 486

`check_module()` (in module `virttest.utils_misc`), 468

`check_option()` (`virttest.storage.QemuImg` method), 404

`check_port_in_br()` (`virttest.openvswitch.OpenVSwitchControl` method), 344

`check_result()` (in module `virttest.utils_test.libvirt`), 263

`check_switch_daemon()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`check_token()` (`virttest.cartesian_config.Lexer` method), 304

`check_vfio_id()` (`virttest.utils_misc.VFIOController` method), 467

`check_vfs_count()` (`virttest.test_setup.PciAssignable` method), 409

`check_vms_dst()` (`virttest.utils_test.qemu.MultihostMigration`

- method), 271
- check_vms_src() (virttest.utils_test.qemu.MultihostMigration method), 271
- checkAlpha() (virttest.cartesian_config.LIdentifier method), 302
- checkChar() (virttest.cartesian_config.LIdentifier method), 302
- checkCharAlpha() (virttest.cartesian_config.LIdentifier method), 302
- checkCharAlphaNum() (virttest.cartesian_config.LIdentifier method), 302
- checkCharNumeric() (virttest.cartesian_config.LIdentifier method), 302
- checkNumbers() (virttest.cartesian_config.LIdentifier method), 302
- checksum() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- checksum_device() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- checksums_out() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- children (virttest.cartesian_config.Node attribute), 305
- chmod() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- chown() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- class_to_test (virttest.xml_utils_unittest.test_XMLBackup attribute), 564
- class_to_test (virttest.xml_utils_unittest.test_XMLTreeFile attribute), 564
- classproperty (class in virttest.propan), 357
- clean() (virttest.openvswitch.OpenVSwitch method), 344
- clean() (virttest.openvswitch.OpenVSwitchSystem method), 345
- clean_objects() (virttest.utils_env.Env method), 422
- clean_tmp_dir() (in module virttest.remote_commander.remote_runner), 230
- clean_tmp_files() (in module virttest.aexpect), 288
- clean_tmp_files() (in module virttest.data_dir), 308
- clean_tmp_files() (in module virttest.qemu_vm), 390
- clean_tmp_files() (in module virttest.utils_net), 486
- clean_up_snapshots() (in module virttest.utils_test.libvirt), 264
- cleanup() (virttest.lvm.EmulatedLVM method), 334
- cleanup() (virttest.lvm.LVM method), 335
- cleanup() (virttest.nfs.Nfs method), 343
- cleanup() (virttest.nfs.NFSCClient method), 342
- cleanup() (virttest.qemu_storage.Iscsidev method), 377
- cleanup() (virttest.qemu_storage.LVMdev method), 377
- cleanup() (virttest.qemu_virtio_port.GuestWorker method), 379
- cleanup() (virttest.test_setup.EGDCConfig method), 407
- cleanup() (virttest.test_setup.HugePageConfig method), 407
- cleanup() (virttest.test_setup.KSMConfig method), 408
- cleanup() (virttest.test_setup.LibvirtPolkitConfig method), 408
- cleanup() (virttest.test_setup.PrivateBridgeConfig method), 411
- cleanup() (virttest.test_setup.TransparentHugePageConfig method), 412
- cleanup() (virttest.utils_misc.SELinuxBoolean method), 466
- cleanup() (virttest.utils_net.IPv6Manager method), 481
- cleanup() (virttest.utils_sasl.SASL method), 497
- cleanup() (virttest.utils_test.libvirt.LibvirtNetwork method), 261
- cleanup() (virttest.utils_test.qemu.MultihostMigration method), 271
- cleanup() (virttest.utils_v2v.VMCheck method), 503
- cleanup() (virttest.utils_virtio_port.VirtioPortTest static method), 504
- cleanup_dest_vm() (virttest.utils_test.libvirt.MigrationTest method), 261
- cleanup_env() (in module virttest.standalone_test), 402
- cleanup_pool() (virttest.utils_test.libvirt.PoolVolumeTest method), 262
- cleanup_ports() (virttest.qemu_virtio_port.GuestWorker method), 379
- cleanup_serial_console() (virttest.libvirt_vm.VM method), 324
- cleanup_serial_console() (virttest.qemu_vm.VM method), 383
- cleanup_serial_console() (virttest.virt_vm.BaseVM method), 554
- cleanup_swap() (virttest.libvirt_vm.VM method), 324
- cleanup_vm() (in module virttest.utils_test.libguestfs), 260
- clear() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- clear() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 235
- clear() (virttest.staging.backports.simplejson.OrderedDict method), 242
- clear_event() (virttest.qemu_monitor.QMPMonitor method), 368
- clear_events() (virttest.qemu_monitor.QMPMonitor method), 368
- clear_interface() (in module virttest.utils_spice), 500
- clear_interface_linux() (in module virttest.utils_spice), 500
- clear_win_driver_verifier() (in module virttest.utils_test.qemu), 273
- click_button() (in module virttest.virsh), 514
- click_install_driver() (virttest.utils_v2v.WindowsVMCheck method), 503
- click_left_button() (virttest.utils_v2v.WindowsVMCheck

- method), 503
- click_tab_enter() (virttest.utils_v2v.WindowsVMCheck method), 503
- client (virttest.utils_net.IPv6Manager attribute), 481
- client (virttest.utils_sasl.SASL attribute), 497
- client_cn (virttest.utils_conn.TLSConnection attribute), 418
- client_hosts (virttest.utils_conn.TLSConnection attribute), 418
- client_ifname (virttest.utils_net.IPv6Manager attribute), 481
- client_ip (virttest.utils_conn.ConnectionBase attribute), 416
- client_ip (virttest.utils_conn.UNIXConnection attribute), 419
- client_ipv6_addr (virttest.utils_net.IPv6Manager attribute), 481
- client_libvirtdconf (virttest.utils_conn.UNIXConnection attribute), 419
- client_pwd (virttest.utils_conn.ConnectionBase attribute), 416
- client_pwd (virttest.utils_conn.UNIXConnection attribute), 419
- client_session (virttest.utils_conn.ConnectionBase attribute), 416
- client_setup() (virttest.utils_conn.TLSConnection method), 418
- client_user (virttest.utils_conn.ConnectionBase attribute), 416
- client_user (virttest.utils_conn.UNIXConnection attribute), 419
- clone() (virttest.libvirt_vm.VM method), 324
- clone() (virttest.qemu_vm.VM method), 383
- clone() (virttest.virt_vm.BaseVM method), 554
- clone_image() (virttest.storage.QemuImg static method), 404
- clone_vm_filesystem() (virttest.utils_test.libguestfs.VirtTools method), 260
- clone_volume() (virttest.libvirt_storage.PoolVolume method), 321
- close() (virttest.aexpect.Spawn method), 286
- close() (virttest.element_tree.TreeBuilder method), 310
- close() (virttest.element_tree.XMLTreeBuilder method), 310
- close() (virttest.qemu_io.QemuIO method), 360
- close() (virttest.qemu_io.QemuIOShellSession method), 360
- close() (virttest.qemu_io.QemuIOSystem method), 361
- close() (virttest.qemu_monitor.Monitor method), 365
- close() (virttest.remote.AexpectIOWrapperOut method), 391
- close() (virttest.remote_commander.messenger.IOWrapper method), 225
- close() (virttest.remote_commander.messenger.Messenger method), 225
- close() (virttest.remote_commander.remote_master.CommanderMaster method), 228
- close() (virttest.rss_client.FileTransferClient method), 398
- close() (virttest.tests.unattended_install.RemoteInstall method), 257
- close() (virttest.utils_disk.CdromDisk method), 420
- close() (virttest.utils_disk.CdromInstallDisk method), 420
- close() (virttest.utils_disk.Disk method), 420
- close() (virttest.utils_disk.FloppyDisk method), 421
- close() (virttest.utils_env_unittest.FakeSyncListenServer method), 424
- close_log_file() (in module virttest.utils_misc), 468
- close_pipes() (virttest.remote_commander.remote_runner.CmdSlave method), 229
- close_session() (virttest.lvsb_base.SandboxSession method), 340
- close_session() (virttest.utils_conn.ConnectionBase method), 416
- close_session() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- close_session() (virttest.utils_net.IPv6Manager method), 481
- close_session() (virttest.utils_sasl.SASL method), 497
- close_session() (virttest.virsh.VirshPersistent method), 509
- CLOSE_SESSION_TIMEOUT (virttest.qemu_vm.VM attribute), 381
- close_unused_fds() (in module virttest.remote_commander.remote_runner), 230
- ClusterManager (class in virttest.ovirt), 346
- cmd() (virttest.aexpect.ShellSession method), 283
- cmd() (virttest.guest_agent.QemuAgent method), 314
- cmd() (virttest.ovs_utils.Machine method), 350
- cmd() (virttest.qemu_monitor.HumanMonitor method), 362
- cmd() (virttest.qemu_monitor.QMPMonitor method), 368
- cmd() (virttest.qemu_virtio_port.GuestWorker method), 379
- cmd() (virttest.remote_commander.remote_master.CommanderMaster method), 228
- cmd() (virttest.utils_gdb.GDB method), 426
- cmd() (virttest.utils_libguestfs.GuestfishRemote method), 457
- cmd_hash (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
- cmd_id (virttest.remote_commander.remote_interface.CmdMessage attribute), 227
- cmd_in_src() (virttest.ovs_utils.Machine method), 350

- CMD_LOOKUP_ORDER
(virttest.staging.utils_koji.KojiClient attribute), 251
- cmd_loop() (virttest.remote_commander.remote_runner.CommanderSlave method), 229
- cmd_obj() (virttest.guest_agent.QemuAgent method), 314
- cmd_obj() (virttest.qemu_monitor.QMPMonitor method), 369
- cmd_output() (in module virttest.lvm), 338
- cmd_output() (virttest.aexpect.ShellSession method), 283
- cmd_output() (virttest.qemu_io.QemuIO method), 360
- cmd_output() (virttest.qemu_io.QemuIOShellSession method), 360
- cmd_output() (virttest.qemu_io.QemuIOSystem method), 361
- cmd_output_safe() (virttest.aexpect.ShellSession method), 284
- cmd_qmp() (virttest.qemu_monitor.QMPMonitor method), 369
- cmd_raw() (virttest.guest_agent.QemuAgent method), 315
- cmd_raw() (virttest.qemu_monitor.QMPMonitor method), 369
- cmd_result() (virttest.utils_libguestfs.GuestfishRemote method), 457
- cmd_result() (virttest.utils_libguestfs.GuestfishSession method), 457
- cmd_result() (virttest.virsh.VirshSession method), 510
- cmd_state() (virttest.ova_utils.Machine method), 350
- cmd_status() (virttest.aexpect.ShellSession method), 284
- cmd_status_output() (virttest.aexpect.ShellSession method), 285
- cmd_status_output() (virttest.utils_libguestfs.GuestfishRemote method), 457
- cmd_status_output() (virttest.utils_libguestfs.GuestfishSession method), 457
- cmd_status_output() (virttest.virsh.VirshSession method), 510
- CMD_TIMEOUT (virttest.guest_agent.QemuAgent attribute), 313
- CMD_TIMEOUT (virttest.qemu_monitor.HumanMonitor attribute), 361
- CMD_TIMEOUT (virttest.qemu_monitor.QMPMonitor attribute), 367
- cmd_tokens() (in module virttest.cartesian_config), 307
- CmdEncapsulation (class in virttest.remote_commander.remote_master), 227
- CmdFinish (class in virttest.remote_commander.remote_runner), 229
- cmdline (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- cmdline() (virttest.qemu_devices.qcontainer.DevContainer method), 215
- cmdline() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- cmdline() (virttest.qemu_devices.qdevices.QCustomDevice method), 221
- cmdline() (virttest.qemu_devices.qdevices.QGlobal method), 222
- cmdline() (virttest.qemu_devices.qdevices.QStringDevice method), 223
- cmdline_nd() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- cmdline_nd() (virttest.qemu_devices.qdevices.QCustomDevice method), 221
- cmdline_nd() (virttest.qemu_devices.qdevices.QStringDevice method), 223
- CmdMaster (class in virttest.remote_commander.remote_master), 227
- CmdMessage (class in virttest.remote_commander.remote_interface), 226
- CmdSlave (class in virttest.remote_commander.remote_runner), 229
- CmdTimeout, 228
- CmdTraceBack, 227
- code (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
- code (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 149
- codec_type (virttest.libvirt_xml.devices.sound.Sound attribute), 138
- combine() (in module virttest.qemu_qtree_unittest), 376
- command() (in module virttest.virsh), 514
- command() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- command_lines() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- command_suffixes() (virttest.lvsb.TestBaseSandboxes method), 338
- Commander (class in virttest.remote_commander.remote_master), 228
- commander() (virttest.virt_vm.BaseVM method), 554
- CommanderError, 227
- CommanderMaster (class in virttest.remote_commander.remote_master), 228
- CommanderSlave (class in virttest.remote_commander.remote_runner), 229
- CommanderSlaveCmds (class in virttest.remote_commander.remote_runner), 229
- comment (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
- comment (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr

- attribute), 140
- comment (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 141
- comment (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
- comment (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144
- comment (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
- comment (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147
- comment (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
- comment (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 150
- comment (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
- comment (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
- comment (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
- comment (virttest.libvirt_xml.nwfilter_protocols.mac.Mac.Attr attribute), 154
- comment (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155
- comment (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 156
- comment (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
- comment (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- comment (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
- comment (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
- comment (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163
- comment (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 164
- comment (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 166
- comment (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 167
- comment (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan.Attr attribute), 168
- Comment() (in module virttest.element_tree), 309
- commit() (virttest.libvirt_storage.QemuImg method), 321
- commit() (virttest.qemu_storage.QemuImg method), 377
- compare_images() (virttest.qemu_storage.QemuImg method), 377
- compare_matrices() (in module virttest.postprocess_iozone), 353
- compare_string() (in module virttest.cartesian_config), 307
- compat (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 211
- compile_autotools_app_tar() (virttest.ovs_utils.Machine method), 350
- complete_cmd() (virttest.utils_libguestfs.Guestfish method), 427
- complete_mac_address() (virttest.utils_net.VirtIface class method), 485
- complete_mac_address() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 492
- complete_uri() (in module virttest.libvirt_vm), 331
- compress_device_out() (virttest.utils_libguestfs.GuestfishPersistent method), 431
- compress_out() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- Condition (class in virttest.cartesian_config), 300
- conf_path (virttest.utils_config.LibvirtConfigCommon attribute), 413
- conf_path (virttest.utils_config.LibvirtConfig attribute), 413
- conf_path (virttest.utils_config.LibvirtSysConfig attribute), 413
- conf_path (virttest.utils_config.LibvirtGuestsConfig attribute), 413
- conf_path (virttest.utils_config.LibvirtQemuConfig attribute), 413
- conf_path (virttest.utils_config_unittest.LibvirtConfigCommonTest.NoType attribute), 414
- conf_path (virttest.utils_config_unittest.LibvirtConfigCommonTest.Undefined attribute), 414
- config() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- config_file (virttest.libvirt_xml.devices.disk.Disk.DiskSource attribute), 126
- CONFIG_MAP (virttest.staging.utils_koji.KojiClient attribute), 251
- ConfigError, 412
- ConfigNoOptionError, 412
- clone() (virttest.build_helper.GnuSourceBuildHelper method), 295
- clone() (in module virttest.standalone_test), 402
- configure_file_logging() (in module virttest.standalone_test), 402
- configure_logging() (virttest.postprocess_iozone.AnalyzerLoggingConfig method), 351
- configure_logging() (virttest.utils_misc.VirtLoggingConfig method), 467
- conn_check() (virttest.utils_conn.ConnectionBase method), 416
- conn_check() (virttest.utils_conn.SSHConnection method), 417
- conn_recover() (virttest.utils_conn.ConnectionBase method), 417

- method), 416
- conn_recover() (virttest.utils_conn.SSHConnection method), 417
- conn_recover() (virttest.utils_conn.TCPConnection method), 418
- conn_recover() (virttest.utils_conn.TLSConnection method), 418
- conn_recover() (virttest.utils_conn.UNIXConnection method), 419
- conn_setup() (virttest.utils_conn.ConnectionBase method), 416
- conn_setup() (virttest.utils_conn.SSHConnection method), 417
- conn_setup() (virttest.utils_conn.TCPConnection method), 418
- conn_setup() (virttest.utils_conn.TLSConnection method), 418
- conn_setup() (virttest.utils_conn.UNIXConnection method), 419
- ConnCertError, 415
- ConnCmdClientError, 415
- ConnCopyError, 415
- connect() (in module virttest.ovirt), 349
- connect() (in module virttest.virsh), 514
- connect_libvirtd() (in module virttest.utils_test.libvirt), 264
- CONNECT_TIMEOUT (virttest.qemu_monitor.Monitor attribute), 365
- connected (virttest.lvsb_base.SandboxSession attribute), 340
- ConnectionBase (class in virttest.utils_conn), 416
- ConnectionError, 417
- ConnForbiddenError, 415
- ConnLoginError, 415
- ConnMkdirError, 415
- ConnNotImplementedError, 415
- ConnPrivKeyError, 415
- ConnRmCertError, 415
- ConnSCPErr, 415
- ConnServerRestartError, 416
- ConnToolNotFoundError, 416
- Console (class in virttest.libvirt_xml.devices.console), 124
- ConstantsTest (class in virttest.service_unittest), 400
- ConstantsTest (class in virttest.virsh_unittest), 551
- ConstructorsTest (class in virttest.virsh_unittest), 551
- cont() (virttest.utils_gdb.GDB method), 426
- cont() (virttest.utils_libvirtd.LibvirtdSession method), 462
- Container (class in virttest.qemu_devices_unittest), 359
- CONTAINER_ENCODER_MAPPING (virttest.video_maker.GstPythonVideoMaker attribute), 508
- CONTAINER_MAPPING (virttest.video_maker.GstPythonVideoMaker attribute), 508
- content (virttest.cartesian_config.Condition attribute), 300
- content (virttest.cartesian_config.NegativeCondition attribute), 305
- content (virttest.cartesian_config.Node attribute), 305
- Controller (class in virttest.libvirt_xml.devices.controller), 124
- Controller.Address (class in virttest.libvirt_xml.devices.controller), 124
- convert() (virttest.libvirt_storage.QemuImg method), 321
- convert() (virttest.qemu_storage.QemuImg method), 377
- convert_data_size() (in module virttest.cartesian_config), 307
- convert_ipv4_to_ipv6() (in module virttest.utils_misc), 468
- convert_systemd_target_to_runlevel() (in module virttest.staging.service), 246
- convert_sysv_runlevel() (in module virttest.staging.service), 246
- convert_version_to_int() (virttest.openvswitch.OpenVSwitchControl static method), 344
- copy() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176
- copy() (virttest.propcan.PropCanBase method), 357
- copy() (virttest.staging.backports.collections.defaultdict.defaultdict method), 232
- copy() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- copy() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 235
- copy() (virttest.staging.backports.simplejson.OrderedDict method), 242
- copy_file() (virttest.remote_commander.remote_runner.CommanderSlaveC method), 230
- copy_file_from_nfs() (in module virttest.tests.unattended_install), 258
- copy_files_from() (in module virttest.remote), 392
- copy_files_from() (virttest.virt_vm.BaseVM method), 554
- COPY_FILES_TIMEOUT (virttest.virt_vm.BaseVM attribute), 553
- copy_files_to() (in module virttest.remote), 392
- copy_files_to() (virttest.virt_vm.BaseVM method), 554
- copy_ifcfg_back() (virttest.utils_test.libguestfs.GuestfishTools method), 259
- copy_in() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- copy_in() (virttest.utils_test.libguestfs.VirtTools method), 260
- copy_out() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- copy_out() (virttest.utils_test.libguestfs.VirtTools method), 260

- method), 260
- copy_size() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- copy_to() (virttest.ovs_utils.Machine method), 350
- copy_to() (virttest.utils_disk.Disk method), 420
- copy_to() (virttest.utils_disk.FloppyDisk method), 421
- copy_windows_file() (virttest.utils_v2v.WindowsVMCheck method), 503
- copyfile_range() (virttest.http_server.HTTPRequestHandler method), 316
- copytree() (in module virttest.utils_disk), 422
- correct() (virttest.qemu_monitor.Monitor method), 366
- counter (virttest.utils_net_unittest.TestVmNetSubclasses attribute), 493
- counter_decrease() (virttest.virsh.VirshPersistent method), 509
- counter_increase() (virttest.virsh.VirshPersistent method), 510
- COUNTERS (virttest.virsh.VirshPersistent attribute), 509
- cp() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- cp_a() (virttest.utils_libguestfs.GuestfishPersistent method), 432
- cp_linux_kernel() (virttest.build_helper.LinuxKernelBuildHelper method), 296
- cpu (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
- cpu (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- cpu_allowed_list_by_task() (in module virttest.utils_test.libvirt), 264
- cpu_baseline() (in module virttest.virsh), 514
- cpu_compare() (in module virttest.virsh), 514
- cpu_count (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 177
- cpu_models() (in module virttest.virsh), 515
- cpu_stats() (in module virttest.virsh), 515
- cpu_str_to_list() (in module virttest.utils_misc), 468
- cpu_topology (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 177
- CpuInfo (class in virttest.virt_vm), 559
- cpus_num (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
- cpus_parser() (in module virttest.utils_test.libvirt), 264
- cpus_string_to_affinity_list() (in module virttest.utils_test.libvirt), 264
- cpuset (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- cputune (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- create() (in module virttest.virsh), 515
- create() (virttest.libvirt_storage.QemuImg method), 322
- create() (virttest.libvirt_vm.VM method), 325
- create() (virttest.libvirt_xml.network_xml.NetworkXML method), 181
- create() (virttest.libvirt_xml.vol_xml.VolXML method), 211
- create() (virttest.lvm.LogicalVolume method), 336
- create() (virttest.lvm.PhysicalVolume method), 336
- create() (virttest.lvm.VolumeGroup method), 337
- create() (virttest.lvsbs.SandboxService method), 341
- create() (virttest.qemu_storage.QemuImg method), 378
- create() (virttest.qemu_vm.VM method), 383
- create() (virttest.utils_net.Macvtap method), 482
- create_and_open_macvtap() (in module virttest.utils_net), 486
- create_bridge_xml() (virttest.utils_test.libvirt.LibvirtNetwork method), 261
- create_by_xpath() (virttest.xml_utils.XMLTreeFile method), 563
- create_channel_xml() (in module virttest.utils_test.libvirt), 264
- create_config_files() (in module virttest.bootstrap), 293
- create_config_files() (in module virttest.standalone_test), 402
- create_disk_xml() (in module virttest.utils_test.libvirt), 264
- create_fs() (virttest.utils_test.libguestfs.GuestfishTools method), 259
- create_generic_service() (virttest.staging.service.Factory static method), 245
- create_guest_os_cfg() (in module virttest.bootstrap), 293
- create_hostdev_xml() (in module virttest.utils_test.libvirt), 264
- create_image() (virttest.utils_test.RemoteDiskManager method), 275
- create_iSCSI() (virttest.iscsi.Iscsi static method), 318
- create_Kn() (virttest.RFBDes.Des method), 279
- create_local_disk() (in module virttest.utils_test.libvirt), 264
- create_macvtap() (in module virttest.utils_net), 487
- create_macvtap_xml() (virttest.utils_test.libvirt.LibvirtNetwork method), 261
- create_monitor() (in module virttest.qemu_monitor), 373
- create_msdos_part() (virttest.utils_test.libguestfs.GuestfishTools method), 259
- create_net_xml() (in module virttest.utils_test.libvirt), 264
- create_nwfilter_xml() (in module virttest.utils_test.libvirt), 264
- create_process_cmd() (in module virttest.remote_commander.remote_runner), 230
- create_qdev() (virttest.qemu_devices_unittest.Container method), 359
- create_scsi_disk() (in module virttest.utils_test.libvirt), 264
- create_serial_console() (virttest.libvirt_vm.VM method),

- 325
 create_serial_console() (virttest.qemu_vm.VM method), 384
 create_serial_console() (virttest.virt_vm.BaseVM method), 555
 create_service() (virttest.staging.service.Factory static method), 246
 create_session() (virttest.utils_v2v.VMCheck method), 503
 create_specific_service() (virttest.staging.service.Factory static method), 246
 create_subtests_cfg() (in module virttest.bootstrap), 293
 create_swap_file() (virttest.libvirt_vm.VM method), 325
 create_swap_partition() (virttest.libvirt_vm.VM method), 325
 create_template() (virttest.ovirt.VMManager method), 348
 create_vg() (virttest.utils_test.RemoteDiskManager method), 275
 create_virtio_console() (virttest.qemu_vm.VM method), 384
 create_virtio_console() (virttest.virt_vm.BaseVM method), 555
 create_vm() (virttest.utils_env.Env method), 423
 create_vnet_xml() (virttest.utils_test.libvirt.LibvirtNetwork method), 261
 create_volume() (virttest.libvirt_storage.PoolVolume method), 321
 create_whole_disk_msdos_part() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 create_x509_dir() (in module virttest.utils_misc), 468
 creation_time (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase attribute), 197
 crypt() (virttest.RFBDes.Des method), 279
 current_mem (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
 current_mem_unit (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
 current_vcpu (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
 custom_pki_path (virttest.utils_conn.TLSConnection attribute), 419
- ## D
- daemonize() (in module virttest.remote_commander.remote_runner), 230
 data() (virttest.element_tree.TreeBuilder method), 310
 DATA_AVAILABLE_TIMEOUT (virttest.qemu_monitor.Monitor attribute), 365
 DataCenterManager (class in virttest.ovirt), 346
 DataWrapper (class in virttest.remote_commander.messenger), 224
 DataWrapperBase64 (class in virttest.remote_commander.messenger), 224
 db_entry() (virttest.utils_net.DbNet method), 480
 db_filename (virttest.utils_net_unittest.TestVmNetSubclasses attribute), 493
 db_item_count (virttest.utils_net_unittest.TestVmNetSubclasses attribute), 493
 DbNet (class in virttest.utils_net), 480
 DbNoLockError, 480
 dd() (virttest.utils_libguestfs.GuestfishPersistent method), 432
 deactivate_netdev() (virttest.qemu_vm.VM method), 384
 deactivate_nic() (virttest.libvirt_vm.VM method), 325
 deactivate_nic() (virttest.qemu_vm.VM method), 384
 deactivate_nic() (virttest.virt_vm.BaseVM method), 555
 debug (virttest.utils_libguestfs.LibguestfsBase attribute), 458
 debug (virttest.virsh.VirshBase attribute), 509
 debug() (virttest.utils_libguestfs.GuestfishPersistent method), 432
 debug_xml() (virttest.libvirt_xml.network_xml.NetworkXML method), 181
 debug_xml() (virttest.libvirt_xml.pool_xml.PoolXML method), 192
 decode() (virttest.remote_commander.messenger.DataWrapper method), 224
 decode() (virttest.remote_commander.messenger.DataWrapperBase64 method), 224
 decode() (virttest.staging.backports.simplejson.decoder.JSONDecoder method), 233
 decode() (virttest.staging.backports.simplejson.JSONDecoder method), 240
 decodeFacilityPriority() (virttest.syslog_server.RequestHandler method), 406
 default (virttest.cartesian_config.Node attribute), 305
 default (virttest.libvirt_xml.network_xml.PortgroupXML attribute), 185
 default() (virttest.staging.backports.simplejson.encoder.JSONEncoder method), 234
 default() (virttest.staging.backports.simplejson.JSONEncoder method), 241
 default_data (virttest.remote_unittest.RemoteFileTest attribute), 397
 DEFAULT_VIRT_NAME (virttest.installer.InstallerRegistry attribute), 317
 defaultdict (class in virttest.staging.backports.collections.defaultdict), 232
 defaultMode (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129
 define() (in module virttest.virsh), 515
 define() (virttest.libvirt_vm.VM method), 325

`define()` (`virttest.libvirt_xml.network_xml.NetworkXML` method), 181

`define()` (`virttest.libvirt_xml.vm_xml.VMXML` method), 203

`define_dir_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_disk_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_fs_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_iscsi_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_lvm_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_netfs_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_new_vm()` (in module `virttest.utils_test.libguestfs`), 260

`define_new_vm()` (in module `virttest.utils_test.libvirt`), 264

`define_pool()` (in module `virttest.utils_test.libvirt`), 265

`define_rbd_pool()` (`virttest.libvirt_storage.StoragePool` method), 322

`define_vm_with_newdisk()` (`virttest.utils_test.libguestfs.VirtTools` method), 260

`defined` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` attribute), 183

`del_active()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_autostart()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_br()` (`virttest.openvswitch.OpenVSwitchControl` method), 344

`del_br()` (`virttest.openvswitch.OpenVSwitchControlCli_140` method), 345

`del_bridge()` (`virttest.utils_net.Bridge` method), 480

`del_cap()` (`virttest.libvirt_xml.nodedev_xml.NodedevXMLBase` method), 186

`del_channel()` (`virttest.libvirt_xml.devices.graphics.Graphics` method), 129

`del_client_session()` (`virttest.utils_conn.ConnectionBase` method), 416

`del_controller()` (`virttest.libvirt_xml.vm_xml.VMXMLBase` method), 209

`del_defcon()` (in module `virttest.utils_selinux`), 498

`del_defined()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_device()` (`virttest.libvirt_xml.vm_xml.VMXML` method), 203

`del_devices()` (`virttest.libvirt_xml.vm_xml.VMXMLBase` method), 209

`del_disks()` (`virttest.libvirt_xml.snapshot_xml.SnapshotXML` method), 196

`del_from_bridge()` (in module `virttest.utils_net`), 487

`del_graphic()` (`virttest.libvirt_xml.devices.graphics.Graphics` static method), 129

`del_ip()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_listens()` (`virttest.libvirt_xml.devices.graphics.Graphics` method), 129

`del_memoryBacking_tag()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 203

`del_net_if_ip()` (in module `virttest.utils_net`), 487

`del_netdev()` (`virttest.qemu_vm.VM` method), 384

`del_nic()` (`virttest.qemu_vm.VM` method), 384

`del_nic()` (`virttest.virt_vm.BaseVM` method), 555

`del_ovs_bridge()` (in module `virttest.utils_net`), 487

`del_persistent()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_port()` (`virttest.openvswitch.OpenVSwitchControl` method), 344

`del_port()` (`virttest.openvswitch.OpenVSwitchControlCli_140` method), 345

`del_port()` (`virttest.utils_net.Bridge` method), 480

`del_portgroup()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`del_protocol()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules` method), 191

`del_rule()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase` method), 190

`del_seclabel()` (`virttest.libvirt_xml.vm_xml.VMXMLBase` method), 209

`del_server_session()` (`virttest.utils_conn.ConnectionBase` method), 416

`del_source()` (`virttest.libvirt_xml.pool_xml.PoolXMLBase` method), 193

`del_sources()` (`virttest.libvirt_xml.devices.character.CharacterBase` method), 123

`del_targets()` (`virttest.libvirt_xml.devices.character.CharacterBase` method), 123

`del_validates()` (`virttest.libvirt_xml.base.LibvirtXMLBase` method), 176

`del_vlan_iface()` (`virttest.ovs_utils.Machine` method), 350

`del_xmltreefile()` (`virttest.libvirt_xml.base.LibvirtXMLBase` method), 176

`delete()` (`virttest.ovirt.VMManager` method), 348

`delete()` (`virttest.utils_net.Macvtap` method), 483

`delete_chap_account()` (`virttest.iscsi.IscsiTGT` method), 319

`delete_event()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 432

`delete_from_export_domain()` (`virttest.ovirt.VMManager` method), 348

`delete_local_disk()` (in module `virttest.utils_test.libvirt`), 265

`delete_pid_file_if_exists()` (in module `virttest.utils_misc`),

- 468
- `delete_pool()` (virttest.libvirt_storage.StoragePool method), 322
- `delete_scsi_disk()` (in module virttest.utils_test.libvirt), 265
- `delete_target()` (virttest.iscsi.IscsiLIO method), 318
- `delete_target()` (virttest.iscsi.IscsiTGT method), 319
- `delete_volume()` (virttest.libvirt_storage.PoolVolume method), 321
- `delete_windows_file()` (virttest.utils_v2v.WindowsVMCheck method), 503
- `dellink()` (virttest.utils_net.Interface method), 482
- `DelLinkError`, 480
- `dep` (virttest.cartesian_config.Node attribute), 305
- `deploy_epel_repo()` (in module virttest.utils_spice), 500
- `deprecation_warning()` (in module virttest.utils_libvirt), 463
- `Des` (class in virttest.RFBDes), 279
- `des_crypt()` (virttest.RFBDes.Des method), 279
- `desc()` (in module virttest.virsh), 515
- `describe()` (virttest.staging.utils_koji.KojiPkgSpec method), 254
- `describe_invalid()` (virttest.staging.utils_koji.KojiPkgSpec method), 254
- `description` (virttest.libvirt_xml.secret_xml.SecretXMLBased attribute), 195
- `description` (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBased attribute), 197
- `dest` (virttest.cartesian_config.LUpdateFileMap attribute), 304
- `destroy()` (in module virttest.virsh), 515
- `destroy()` (virttest.libvirt_vm.VM method), 325
- `destroy()` (virttest.lvsbs.SandboxService method), 341
- `destroy()` (virttest.ovirt.VMManager method), 348
- `destroy()` (virttest.qemu_vm.VM method), 384
- `destroy()` (virttest.utils_env.Env method), 423
- `destroy()` (virttest.virt_vm.BaseVM method), 555
- `destroy_pool()` (virttest.libvirt_storage.StoragePool method), 323
- `detach_device()` (in module virttest.virsh), 516
- `detach_disk()` (in module virttest.virsh), 516
- `detach_disk()` (virttest.libvirt_vm.VM method), 326
- `detach_interface()` (in module virttest.virsh), 516
- `detach_interface()` (virttest.libvirt_vm.VM method), 326
- `DevContainer` (class in virttest.qemu_devices.qcontainer), 215
- `device` (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- `device` (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- `device_exists()` (in module virttest.utils_test.libvirt), 265
- `device_id` (virttest.utils_net.Qemuiface attribute), 483
- `device_index()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `device_path` (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- `device_tag` (virttest.libvirt_xml.devices.base.UntypedDeviceBase attribute), 122
- `DeviceError`, 223
- `DeviceHotplugError`, 224
- `DeviceInsertError`, 224
- `DeviceRemoveError`, 224
- `Devices` (class in virttest.qemu_devices_unittest), 359
- `devices` (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- `DeviceUnplugError`, 224
- `df()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `df_h()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `dhcp_bootp` (virttest.libvirt_xml.network_xml.IPXML attribute), 180
- `dhcp_ranges` (virttest.libvirt_xml.network_xml.IPXML attribute), 180
- `diff_defcon()` (in module virttest.utils_selinux), 498
- `dir_path` (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- `disable()` (virttest.standalone_test.Bcolors method), 401
- `disable_windows_guest_network()` (in module virttest.utils_net), 487
- `DISCARD_WARNINGS` (virttest.utils_net.VMNet attribute), 484
- `disconnect()` (in module virttest.ovirt), 349
- `Disk` (class in virttest.libvirt_xml.devices.disk), 125
- `Disk` (class in virttest.utils_disk), 420
- `Disk.Address` (class in virttest.libvirt_xml.devices.disk), 125
- `Disk.Auth` (class in virttest.libvirt_xml.devices.disk), 126
- `Disk.DiskSource` (class in virttest.libvirt_xml.devices.disk), 126
- `Disk.Encryption` (class in virttest.libvirt_xml.devices.disk), 126
- `Disk.IOTune` (class in virttest.libvirt_xml.devices.disk), 127
- `disk_enabled` (virttest.libvirt_xml.vm_xml.VMPMXML attribute), 202
- `disk_format()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `disk_has_backing_file()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `disk_name` (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- `disk_size()` (virttest.utils_libguestfs.GuestfishPersistent method), 433
- `display()` (virttest.lvm.LogicalVolume method), 336
- `display()` (virttest.lvm.PhysicalVolume method), 337
- `display_attributes()` (in module virttest.utils_misc), 469
- `dmesg()` (virttest.utils_libguestfs.GuestfishPersistent

- method), 433
- dns (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 183
- dns_forward (virttest.libvirt_xml.network_xml.DNSXML attribute), 180
- DNSXML (class in virttest.libvirt_xml.network_xml), 179
- DNSXML.HostnameXML (class in virttest.libvirt_xml.network_xml), 180
- DNSXML.HostXML (class in virttest.libvirt_xml.network_xml), 180
- do_GET() (virttest.http_server.HTTPRequestHandler method), 316
- do_migration() (in module virttest.utils_test.libvirt), 265
- do_migration() (virttest.utils_test.libvirt.MigrationTest method), 261
- do_mount() (virttest.utils_libguestfs.GuestfishPersistent method), 433
- doctype() (virttest.element_tree.XMLTreeBuilder method), 310
- dom_list() (in module virttest.virsh), 516
- domain (virttest.libvirt_xml.devices.hostdev.Hostdev.SourceAddressTypeAttribute), 130
- domain (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
- domain (virttest.libvirt_xml.nodedev_xml.PCIXML.Address attribute), 187
- domain_exists() (in module virttest.virsh), 516
- domain_name (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 183
- domblkerror() (in module virttest.virsh), 517
- domblkinfo() (in module virttest.virsh), 517
- domblklist() (in module virttest.virsh), 517
- domblkstat() (in module virttest.virsh), 517
- domcapabilities() (in module virttest.virsh), 517
- domcontrol() (in module virttest.virsh), 517
- domdisplay() (in module virttest.virsh), 518
- domfsfreeze() (in module virttest.virsh), 518
- domfsthaw() (in module virttest.virsh), 518
- domfsttrim() (in module virttest.virsh), 518
- domid() (in module virttest.virsh), 518
- domif_getlink() (in module virttest.virsh), 518
- domif_setlink() (in module virttest.virsh), 519
- domiflist() (in module virttest.virsh), 519
- domifstat() (in module virttest.virsh), 519
- domiftune() (in module virttest.virsh), 519
- dominfo() (in module virttest.virsh), 520
- dominfo() (virttest.libvirt_vm.VM method), 326
- domjobabort() (in module virttest.virsh), 520
- domjobabort() (virttest.libvirt_vm.VM method), 326
- domjobinfo() (in module virttest.virsh), 520
- dommemstat() (in module virttest.virsh), 520
- domname() (in module virttest.virsh), 520
- dompmwsuspend() (in module virttest.virsh), 520
- dompmwakeup() (in module virttest.virsh), 521
- domstate() (in module virttest.virsh), 521
- domstats() (in module virttest.virsh), 521
- domtime() (in module virttest.virsh), 521
- domuuid() (in module virttest.virsh), 521
- domxml_from_native() (in module virttest.virsh), 522
- domxml_to_native() (in module virttest.virsh), 522
- down() (virttest.utils_net.Interface method), 482
- download() (in module virttest.rss_client), 399
- download() (virttest.build_helper.PatchHelper method), 297
- download() (virttest.rss_client.FileDownloadClient method), 398
- download() (virttest.utils_libguestfs.GuestfishPersistent method), 433
- download_all_test_providers() (in module virttest.asset), 289
- download_asset() (in module virttest.asset), 289
- download_file() (in module virttest.asset), 290
- download_offset() (virttest.utils_libguestfs.GuestfishPersistent method), 433
- download_offset() (in module virttest.asset), 290
- download_type() (in module virttest.asset), 290
- driver (virttest.libvirt_xml.devices.controller.Controller attribute), 124
- driver (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- driver (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- driver (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- driver() (in module virttest.virsh), 522
- driver_attr (virttest.libvirt_xml.devices.interface.Interface.Driver attribute), 132
- driver_guest (virttest.libvirt_xml.devices.interface.Interface.Driver attribute), 132
- driver_host (virttest.libvirt_xml.devices.interface.Interface.Driver attribute), 132
- driver_type (virttest.libvirt_xml.nodedev_xml.StorageXML attribute), 188
- drop_caches() (in module virttest.staging.utils_memory), 256
- drop_caches() (virttest.utils_libguestfs.GuestfishPersistent method), 433
- dscp (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
- dscp (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 140
- dscp (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 141
- dscp (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
- dscp (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
- dscp (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 146

attribute), 147
 dscp (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr dstipaddr (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 148
 dscp (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr dstipaddr (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 150
 dscp (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr dstipaddr (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 151
 dscp (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
 dscp (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr dstipaddr (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 153
 dscp (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr dstipaddr (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 156
 dscp (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 158
 dscp (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 161
 dscp (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 162
 dscp (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 163
 dscp (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 164
 dscp (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 166
 dscp (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 167
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 139
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 140
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 142
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 143
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 146
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 147
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 148
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 150
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr dstipfrom (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 151
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr dstipmask (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 152
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr dstipmask (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 153
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr dstipmask (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 156
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr dstipmask (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 158
 dstipaddr (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr dstipmask (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 161

attribute), 146
dstipmask (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147
dstipmask (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
dstipmask (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 150
dstipmask (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
dstipmask (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
dstipmask (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
dstipmask (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 156
dstipmask (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
dstipmask (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
dstipmask (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
dstipmask (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163
dstipmask (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 165
dstipmask (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 166
dstipmask (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 167
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 140
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 142
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.mac.Mac.Attr attribute), 154
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 166
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 167
dstmacaddr (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan.Attr attribute), 168
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 140
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 142
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
dstmacmask (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147

attribute), 147
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.mac.Mac.Attr attribute), 154
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 166
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 167
 dstmacmask (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan.Attr attribute), 168
 dstportend (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
 dstportend (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
 dstportend (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 157
 dstportend (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
 dstportend (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
 dstportend (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
 dstportend (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163
 dstportend (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 165
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 157
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163
 dstportstart (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 165
 dtb (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
 du() (virttest.utils_libguestfs.GuestfishPersistent method), 433
 dump() (in module virttest.element_tree), 309
 dump() (in module virttest.staging.backports.simplejson), 237
 dump() (in module virttest.virsh), 522
 dump() (virttest.cartesian_config.Node method), 305
 dump() (virttest.libvirt_vm.VM method), 326
 dump_core (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
 duvps() (in module virttest.staging.backports.simplejson), 238
 duvpsxml() (in module virttest.virsh), 522
 E
 E (virttest.RFBDes.Des attribute), 279
 eject_cdrom() (virttest.utils_libguestfs.GuestfishPersistent method), 434
 eject_cdrom() (virttest.qemu_monitor.HumanMonitor method), 362
 eject_cdrom() (virttest.qemu_monitor.QMPMonitor method), 370
 eject_cdrom() (virttest.qemu_vm.VM method), 385
 Element() (in module virttest.element_tree), 309
 element_by_parent() (virttest.libvirt_xml.accessors.AccessorBase method), 169
 ElementTree (class in virttest.element_tree), 309
 EmulatedLVM (class in virttest.lvm), 334
 Emulator (class in virttest.libvirt_xml.devices.emulator), 128
 emulator_period (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 197
 emulator_quota (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 197
 emulatorpin (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 197
 emulatorpin() (in module virttest.virsh), 523
 enable_debug_symbols() (virttest.build_helper.GnuSourceBuildHelper method), 295

[enable_windows_guest_network\(\)](#) (in module [event\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#)
[virttest.utils_net](#)), 487 [method](#)), 434
[encap_protocol](#) ([virttest.libvirt_xml.nwfilter_protocols.vlan.VlanAttr](#) attribute), 168 [VlanAttr.edit\(\)](#) (in module [virttest.utils_test.libvirt](#)), 265
[encode\(\)](#) ([virttest.remote_commander.messenger.DataWrapper.execute\(\)](#) ([virttest.build_helper.GitRepoParamHelper](#)
[method](#)), 224 [method](#)), 294
[encode\(\)](#) ([virttest.remote_commander.messenger.DataWrapper.execute_base64\(\)](#) ([virttest.build_helper.GnuSourceBuildHelper](#)
[method](#)), 224 [method](#)), 295
[encode\(\)](#) ([virttest.staging.backports.simplejson.encoder.JSONEncoder.encode\(\)](#) ([virttest.build_helper.LinuxKernelBuildHelper](#)
[method](#)), 234 [method](#)), 296
[encode\(\)](#) ([virttest.staging.backports.simplejson.encoder.JSONEncoder.encode_for_virttest\(\)](#) ([virttest.build_helper.LocalSourceDirHelper](#)
[method](#)), 235 [method](#)), 296
[encode\(\)](#) ([virttest.staging.backports.simplejson.JSONEncoder.execute\(\)](#) ([virttest.build_helper.LocalTarHelper](#) [method](#)),
[method](#)), 241 [method](#)), 297
[encode_basestring\(\)](#) (in module [execute\(\)](#) ([virttest.build_helper.PatchHelper](#) [method](#)), 297
[virttest.staging.backports.simplejson.encoder](#)), [execute\(\)](#) ([virttest.build_helper.RemoteTarHelper](#)
235 [method](#)), 297
[encode_basestring_ascii\(\)](#) (in module [execute_qemu\(\)](#) ([virttest.qemu_devices.qcontainer.DevContainer](#)
[virttest.staging.backports.simplejson.encoder](#)), [method](#)), 216
235
[ENCODER_MAPPING](#) ([virttest.video_maker.GstPythonVideoMaker](#) [ExistPoolTest](#) (class in [virttest.libvirt_storage_unittest](#)),
[attribute](#)), 508 [exists\(\)](#) ([virttest.libvirt_vm.VM](#) [method](#)), 326
[encryption](#) ([virttest.libvirt_xml.devices.disk.Disk](#) [attribute](#)), 127 [exists\(\)](#) ([virttest.libvirt_xml.network_xml.NetworkXML](#)
[attribute](#)), 127 [method](#)), 181
[encryption](#) ([virttest.libvirt_xml.devices.disk.Disk.Encryption](#) [exists\(\)](#) ([virttest.lvm.Volume](#) [method](#)), 337
[attribute](#)), 127 [exists\(\)](#) ([virttest.lvm.VolumeGroup](#) [method](#)), 338
[encryption](#) ([virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase](#) [exists\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#)
[attribute](#)), 195 [method](#)), 434
[encryption](#) ([virttest.libvirt_xml.vol_xml.VolXMLBase](#) [exit\(\)](#) (in module [virttest.virsh](#)), 523
[attribute](#)), 211 [exit\(\)](#) ([virttest.remote_commander.remote_runner.CommanderSlaveCmds](#)
[method](#)), 230
[end\(\)](#) ([virttest.element_tree.TreeBuilder](#) [method](#)), 310
[Env](#) (class in [virttest.utils_env](#)), 422
[env_cleanup\(\)](#) ([virttest.utils_netperf.NetperfPackage](#) [exit\(\)](#) ([virttest.utils_gdb.GDB](#) [method](#)), 426
[method](#)), 495 [exit\(\)](#) ([virttest.utils_libvirtd.LibvirtdSession](#) [method](#)), 462
[env_version](#) ([virttest.standalone_test.Test](#) [attribute](#)), 401
[EnvSaveError](#), 424
[equal\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) [exit_code\(\)](#) ([virttest.lvsb_base.SandboxBase](#) [method](#)),
[method](#)), 434 [method](#)), 339
[ERROR_REGEX_LIST](#) ([virttest.utils_libguestfs.GuestfishRemote](#) [exit_code\(\)](#) ([virttest.lvsb_base.SandboxSession](#) [method](#)),
[attribute](#)), 457 [method](#)), 260
[ERROR_REGEX_LIST](#) ([virttest.utils_libguestfs.GuestfishSession](#) [expand_vm_filesystem\(\)](#) ([virttest.utils_test.libguestfs.VirtTools](#)
[attribute](#)), 457 [method](#)), 282
[ERROR_REGEX_LIST](#) ([virttest.virsh.VirshSession](#) [Expect](#) (class in [virttest.aexpect](#)), 280
[attribute](#)), 510 [ExpectError](#), 282
[Esp](#) (class in [virttest.libvirt_xml.nwfilter_protocols.esp](#)), 145 [ExpectProcessTerminatedError](#), 282
[Esp.Attr](#) (class in [virttest.libvirt_xml.nwfilter_protocols.esp](#)), 145 [ExpectTimeoutError](#), 283
[Esp_ipv6](#) (class in [virttest.libvirt_xml.nwfilter_protocols.esp_ipv6](#)), 147
[Esp_ipv6.Attr](#) (class in [virttest.libvirt_xml.nwfilter_protocols.esp_ipv6](#)), 147
[event\(\)](#) (in module [virttest.virsh](#)), 523 [export\(\)](#) ([virttest.nfs.Exportfs](#) [method](#)), 342
[export_target\(\)](#) ([virttest.iscsi.IscsiLIO](#) [method](#)), 318
[export_target\(\)](#) ([virttest.iscsi.IscsiTGT](#) [method](#)), 319
[Exportfs](#) (class in [virttest.nfs](#)), 342
[extend\(\)](#) ([virttest.libvirt_xml.nwfilter_xml.NwfilterRulesProtocol](#)
[method](#)), 188
[extend\(\)](#) ([virttest.libvirt_xml.vm_xml.VMXMLDevices](#)
[method](#)), 210
[extend_pv\(\)](#) ([virttest.lvm.VolumeGroup](#) [method](#)), 338

- extlinux() (virttest.utils_libguestfs.GuestfishPersistent method), 434
- extract() (virttest.build_helper.LocalTarHelper method), 297
- extract_qemu_cpu_models() (in module virttest.utils_misc), 469
- ## F
- f() (virttest.RFBDes.Des method), 279
- fabric_wwn (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
- FACILITY_NAMES (virttest.syslog_server.RequestHandler attribute), 405
- Factory (class in virttest.staging.service), 244
- factory() (in module virttest.versionable_class), 506
- factory() (virttest.versionable_class.Manager method), 505
- Factory.FactoryHelper (class in virttest.staging.service), 244
- failed_cases (virttest.cartesian_config.Node attribute), 305
- FailedInstaller (class in virttest.base_installer), 292
- FakeCmd (class in virttest.utils_misc_unittest), 479
- FakeService (class in virttest.nfs_unittest), 343
- FakeSyncListenServer (class in virttest.utils_env_unittest), 424
- FakeVirshFactory() (in module virttest.virsh_unittest), 551
- FakeVm (class in virttest.utils_env_unittest), 424
- FakeVm (class in virttest.utils_net_unittest), 492
- fakevm_generator() (virttest.utils_net_unittest.TestVmNetSubclass method), 493
- fallback (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 198
- fallocate() (virttest.utils_libguestfs.GuestfishPersistent method), 434
- fallocate64() (virttest.utils_libguestfs.GuestfishPersistent method), 434
- family (virttest.libvirt_xml.network_xml.IPXML attribute), 181
- fc_type (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
- feature_available() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- feature_list (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 177
- feature_list (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 198
- feature_list (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- features (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- feed() (virttest.element_tree.XMLTreeBuilder method), 310
- feed() (virttest.xml_utils.TemplateXMLTreeBuilder method), 563
- fgrep() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- fgrep() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- file() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- file_architecture() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- file_exists() (in module virttest.gluster), 313
- file_exists() (in module virttest.storage), 404
- file_remove() (in module virttest.storage), 404
- file_replace_append() (virttest.test_setup.LibvirtPolkitConfig method), 409
- FileDownloadClient (class in virttest.rss_client), 398
- filename (virttest.cartesian_config.Node attribute), 306
- fileno() (virttest.remote.AexpectIOWrapperOut method), 391
- fileno() (virttest.remote_commander.messenger.IOWrapper method), 225
- fileno() (virttest.remote_commander.messenger.StdIOWrapper method), 225
- FileReader (class in virttest.cartesian_config), 300
- filesize() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- Filesystem (in module virttest.libvirt_xml.devices.filesystem), 128
- filesystem_available() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- FileTransferClient (class in virttest.rss_client), 398
- FileTransferConnectError, 399
- FileTransferError, 399
- FileTransferNotFoundError, 399
- FileTransferProtocolError, 399
- FileTransferServerError, 399
- FileTransferSocketError, 399
- FileTransferTimeoutError, 399
- FileUploadClient (class in virttest.rss_client), 399
- fill() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- fill_addrs() (virttest.ovs_utils.Machine method), 350
- fill_addrs() (virttest.virt_vm.BaseVM method), 555
- fill_dir() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- fill_pattern() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- Filter (class in virttest.cartesian_config), 300
- filter (virttest.cartesian_config.Filter attribute), 300
- filter_chain (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189
- filter_chain (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 190
- filter_name (virttest.libvirt_xml.nwfilter_xml.NwfilterXML

- attribute), 189
- filter_name (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 190
- filter_priority (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 189
- filter_priority (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 190
- filterref (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- filterrefs (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 189
- filterrefs (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 190
- find() (in module virttest.element_path), 309
- find() (virttest.element_path.Path method), 309
- find() (virttest.element_tree.ElementTree method), 309
- find_bridge_manager() (in module virttest.utils_net), 487
- find_command() (in module virttest.utils_misc), 469
- find_current_bridge() (in module virttest.utils_net), 487
- find_default_qemu_paths() (in module virttest.standalone_test), 402
- find_defcon() (in module virttest.utils_selinux), 498
- find_defcon_idx() (in module virttest.utils_selinux), 498
- find_dnsmasq_listen_address() (in module virttest.utils_net), 487
- find_free_port() (in module virttest.utils_misc), 469
- find_free_ports() (in module virttest.utils_misc), 469
- find_id_for_screendump() (in module virttest.ppm_utils), 353
- find_pathregex() (in module virttest.utils_selinux), 498
- find_storage_pool_sources() (in module virttest.virsh), 523
- find_storage_pool_sources_as() (in module virttest.virsh), 523
- find_substring() (in module virttest.utils_misc), 469
- findall() (in module virttest.element_path), 309
- findall() (virttest.element_path.Path method), 309
- findall() (virttest.element_tree.ElementTree method), 309
- findfs_label() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- findfs_uuid() (virttest.utils_libguestfs.GuestfishPersistent method), 435
- findtext() (in module virttest.element_path), 309
- findtext() (virttest.element_path.Path method), 309
- findtext() (virttest.element_tree.ElementTree method), 310
- fini() (virttest.lvsb_base.SandboxBase method), 339
- finish() (virttest.remote_commander.remote_runner.CmdSlave method), 229
- firm_release_date (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- firmversion (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- firmware_vendor (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- attribute), 188
- Flag (class in virttest.utils_misc), 463
- flags (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- flags (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
- flags (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
- flatten_options() (virttest.lvsb_base.SandboxCommandBase static method), 340
- FloppyDisk (class in virttest.utils_disk), 421
- flush_ip6tables() (virttest.utils_net.IPv6Manager method), 481
- flush_stdin() (virttest.remote_commander.messenger.Messenger method), 225
- flush_until() (virttest.cartesian_config.Lexer method), 304
- foobar (virttest.libvirt_xml_unittest.Baz attribute), 332
- for_each() (virttest.lvsb_base.TestSandboxes method), 341
- ForAll (class in virttest.utils_misc), 464
- ForAllIP (class in virttest.utils_misc), 464
- ForAllPSE (class in virttest.utils_misc), 464
- ForbiddenBase (class in virttest.libvirt_xml.accessors), 170
- format (virttest.libvirt_xml.vol_xml.VolXML.Encryption attribute), 211
- format (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212
- format_disk() (virttest.utils_test.libguestfs.VirtTools method), 260
- format_guest_disk() (in module virttest.utils_misc), 469
- format_linux_disk() (in module virttest.utils_misc), 469
- format_msg() (virttest.remote_commander.messenger.Messenger method), 225
- format_str_for_message() (in module virttest.utils_misc), 470
- format_type (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- format_windows_disk() (in module virttest.utils_misc), 470
- forward (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 183
- forward_delay (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- forward_delay_hi (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- forward_interface (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 183
- Forwarders (virttest.libvirt_xml.network_xml.DNSXML attribute), 180
- FP (virttest.RFBDes.Des attribute), 279
- free_cpu() (virttest.utils_misc.NumaNod method), 465
- get_mac_address() (virttest.utils_net.VirtNet method),

- 485
 free_mac_address() (virttest.virt_vm.BaseVM method), 555
 freecell() (in module virttest.virsh), 523
 freememtotal() (in module virttest.staging.utils_memory), 256
 freepages() (in module virttest.virsh), 524
 frequency (virttest.libvirt_xml.vm_xml.VMClockXML.TimeXML attribute), 199
 from_dumpxml() (virttest.libvirt_xml.vm_xml.VMClockXML method), 199
 from_element() (virttest.libvirt_xml.devices.base.UntypedDeviceBase method), 122
 from_element() (virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase method), 145
 fromkeys() (virttest.staging.backports.collections.OrderedDict.OrderedDict class method), 231
 fromkeys() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict class method), 235
 fromkeys() (virttest.staging.backports.simplejson.OrderedDict class method), 242
 fromstring() (in module virttest.element_tree), 310
 fsck() (virttest.utils_libguestfs.GuestfishPersistent method), 436
 fsfreeze() (virttest.guest_agent.QemuAgent method), 315
 FSFREEZE_STATUS_FROZEN (virttest.guest_agent.QemuAgent attribute), 313
 FSFREEZE_STATUS_THAWED (virttest.guest_agent.QemuAgent attribute), 313
 fsthaw() (virttest.guest_agent.QemuAgent method), 315
 func (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
 func1() (virttest.versionable_class_unittest.BB method), 506
 func1() (virttest.versionable_class_unittest.VM method), 507
 func1() (virttest.versionable_class_unittest.VM1 method), 507
 func2() (virttest.versionable_class_unittest.BB method), 506
 func2() (virttest.versionable_class_unittest.VM1 method), 507
 func3() (virttest.versionable_class_unittest.VM method), 507
 func3() (virttest.versionable_class_unittest.VM1 method), 507
 function (virttest.cartesian_config.LOperators attribute), 302
 function (virttest.libvirt_xml.devices.hostdev.Hostdev.SourceAddress.UntypedAddress attribute), 130
 function (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
 function (virttest.libvirt_xml.nodedev_xml.PCIXML.Address attribute), 187
G
 g_nic_name (virttest.utils_net.VirtIface attribute), 485
 g_nic_name (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 492
 gdb (GDB class in virttest.utils_gdb), 426
 GDBCmdError, 427
 GDBError, 427
 gen_rv_file() (in module virttest.utils_spice), 500
 get_dir() (in module virttest.remote_commander.remote_runner), 231
 generate_bytes() (virttest.utils_net.VirtIface class method), 485
 generate_bytes() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 493
 generate_data_source() (virttest.postprocess_iozone.IOzonePlotter method), 353
 generate_id() (virttest.lvm.LVM method), 335
 generate_id_for_screendump() (in module virttest.ppm_utils), 353
 generate_ifname() (virttest.utils_net.VirtNet method), 485
 generate_mac_address() (virttest.utils_net.VirtNet method), 485
 generate_mac_address_simple() (in module virttest.utils_net), 487
 generate_params() (virttest.qemu_qtree.QtreeDisk method), 374
 generate_params() (virttest.qemu_qtree.QtreeDisksContainer method), 375
 generate_params() (virttest.qemu_qtree.QtreeNode method), 375
 generate_random_id() (in module virttest.utils_misc), 470
 generate_random_string() (in module virttest.utils_misc), 470
 generate_tmp_file_name() (in module virttest.utils_misc), 470
 geometric_mean() (in module virttest.postprocess_iozone), 353
 geometry (virttest.libvirt_xml.devices.disk.Disk attribute), 127
 geometry (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
 get() (in module virttest.libvirt_xml.devices.librarian), 133
 get() (in module virttest.libvirt_xml.nwfilter_protocols.librarian), 133
 get() (in module virttest.libvirt_xml.devices.librarian), 133
 get() (virttest.qemu_devices.qbuses.QSparseBus method), 214

`get()` (virttest.qemu_devices.qcontainer.DevContainer method), 216

`get_active()` (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 183

`get_addr_list()` (virttest.utils_net.IPv6Manager method), 481

`get_address()` (virttest.ovirt.HostManager method), 347

`get_address()` (virttest.ovirt.VMManager method), 348

`get_address()` (virttest.virt_vm.BaseVM method), 555

`get_address_dict()` (virttest.libvirt_xml.nodedev_xml.PCIXML method), 187

`get_agent_channels()` (virttest.libvirt_xml.vm_xml.VMXML method), 203

`get_aid()` (virttest.qemu_devices.qdevices.QBaseDevice method), 220

`get_all_assets()` (in module virttest.asset), 290

`get_all_cells()` (in module virttest.utils_test.libvirt), 265

`get_all_cgroups()` (virttest.staging.utils_cgroup.Cgroup method), 248

`get_all_controllers()` (in module virttest.staging.utils_cgroup), 250

`get_all_ips()` (in module virttest.utils_net), 487

`get_all_nodes()` (virttest.utils_misc.Numainfo method), 465

`get_all_processors()` (virttest.libvirt_xml.sysinfo_xml.SysinfoXML method), 197

`get_all_protocols()` (virttest.libvirt_xml.nwfilter_xml.NwfilterXML method), 189

`get_all_rules()` (virttest.libvirt_xml.nwfilter_xml.NwfilterXML method), 189

`get_all_vms()` (virttest.utils_env.Env method), 423

`get_all_vol_paths()` (in module virttest.utils_test.libvirt), 265

`get_answer_file_path()` (virttest.tests.unattended_install.RemoteInstall method), 257

`get_answer_file_path()` (virttest.utils_disk.CdromInstallDisk method), 420

`get_answer_file_path()` (virttest.utils_disk.Disk method), 420

`get_append()` (virttest.utils_libguestfs.GuestfishPersistent method), 436

`get_arch()` (virttest.staging.utils_koji.RPMFileNameInfo method), 255

`get_archive_tarball_name()` (in module virttest.utils_misc), 470

`get_asset_info()` (in module virttest.asset), 290

`get_attach_method()` (virttest.utils_libguestfs.GuestfishPersistent method), 436

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.ah.Ah method), 140

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6 method), 141

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.all.All method), 142

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6 method), 143

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.arp.Arp method), 144

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.esp.Esp method), 146

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6 method), 148

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp method), 149

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6 method), 150

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp method), 151

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.ip.Ip method), 152

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6 method), 154

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.mac.Mac method), 155

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp method), 156

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp method), 157

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6 method), 158

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.stp.Stp method), 160

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp method), 161

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6 method), 162

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.udp.Udp method), 164

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6 method), 165

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite method), 166

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6 method), 167

`get_attr()` (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan method), 168

`get_attr()` (virttest.lvm.LogicalVolume method), 336

`get_attr()` (virttest.lvm.PhysicalVolume method), 337

`get_attr()` (virttest.lvm.Volume method), 337

`get_attr()` (virttest.lvm.VolumeGroup method), 338

`get_autostart()` (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 183

`get_autosync()` (virttest.utils_libguestfs.GuestfishPersistent method), 436

`get_available_configure_options()` (virttest.build_helper.GnuSourceBuildHelper method), 295

`get_backend()` (virttest.utils_libguestfs.GuestfishPersistent

- method), 436
- get_backend_cfg_path() (in module virttest.data_dir), 308
- get_backend_dir() (in module virttest.data_dir), 308
- get_backing_data_dir() (in module virttest.data_dir), 308
- get_backingfile() (virttest.qemu_monitor.HumanMonitor method), 362
- get_backingfile() (virttest.qemu_monitor.QMPMonitor method), 370
- get_blk_devices() (virttest.libvirt_vm.VM method), 326
- get_blkdevio_params() (virttest.libvirt_xml.vm_xml.VMXML static method), 203
- get_blkio_params() (virttest.libvirt_xml.vm_xml.VMXML static method), 203
- get_block() (virttest.qemu_qtree.QtreeDisk method), 374
- get_block() (virttest.qemu_vm.VM method), 385
- get_block_old() (virttest.qemu_vm.VM method), 385
- get_boolean() (virttest.utils_config.SectionlessConfig method), 414
- get_bootable_part() (virttest.utils_test.libguestfs.GuestfishTool method), 259
- get_buddy_info() (in module virttest.staging.utils_memory), 256
- get_buses() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- get_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- get_by_properties() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- get_by_qid() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- get_cap() (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase method), 186
- get_cap_by_type() (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase static method), 186
- get_cartesian_parser_details() (in module virttest.standalone_test), 402
- get_cell() (virttest.libvirt_xml.capability_xml.TopologyXML method), 179
- get_cgroup_index() (virttest.staging.utils_cgroup.Cgroup method), 248
- get_cgroup_mountpoint() (in module virttest.staging.utils_cgroup), 250
- get_cgroup_name() (virttest.staging.utils_cgroup.Cgroup method), 249
- get_channel() (virttest.libvirt_xml.devices.graphics.Graphics method), 129
- get_chap_accounts() (virttest.iscsi.IscsiTGT method), 319
- get_children() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- get_children() (virttest.qemu_devices.qdevices.QDevice method), 221
- get_children() (virttest.qemu_devices.qdevices.QHPDrive method), 222
- get_children() (virttest.qemu_devices.qdevices.QRHDrive method), 223
- get_children() (virttest.qemu_qtree.QtreeNode method), 375
- get_client_session() (virttest.utils_conn.ConnectionBase method), 416
- get_cmd_line() (virttest.qemu_io.QemuIO method), 360
- get_cmd_options() (virttest.utils_v2v.Target method), 503
- get_command_output() (virttest.aexpect.ShellSession method), 285
- get_command_status() (virttest.aexpect.ShellSession method), 285
- get_command_status_output() (virttest.aexpect.ShellSession method), 285
- get_configure_command() (virttest.build_helper.GnuSourceBuildHelper method), 295
- get_configure_path() (virttest.build_helper.GnuSourceBuildHelper method), 295
- get_container_name() (virttest.video_maker.GstPythonVideoMaker method), 508
- get_context_from_str() (in module virttest.utils_selinux), 498
- get_context_of_file() (in module virttest.utils_selinux), 498
- get_context_of_process() (in module virttest.utils_selinux), 498
- get_correspond_ip() (in module virttest.utils_net), 487
- get_cpu_count() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 177
- get_cpu_count() (virttest.virt_vm.BaseVM method), 556
- get_cpu_flags() (in module virttest.utils_misc), 470
- get_cpu_info() (in module virttest.utils_misc), 470
- get_cpu_status() (in module virttest.utils_misc), 471
- get_cpu_topology() (virttest.utils_misc.NumaNode method), 466
- get_cpu_topology_in_cmdline() (virttest.libvirt_vm.VM method), 326
- get_cpu_topology_in_vm() (virttest.libvirt_vm.VM method), 326
- get_cpu_vendor() (in module virttest.utils_misc), 471
- get_current_branch() (in module virttest.version), 505
- get_current_memory_size() (virttest.virt_vm.BaseVM method), 556
- get_data_dir() (in module virttest.data_dir), 308
- get_data_dir() (in module virttest.ppm_utils), 354
- get_date() (in module virttest.utils_test), 276
- get_default_command() (virttest.staging.utils_koji.KojiClient method), 251
- get_default_format() (in module virttest.syslog_server), 407
- get_default_guest_os_info() (in module virttest.defaults), 309

`get_default_koji_tag()` (in module `virttest.staging.utils_koji`), 255

`get_defcon()` (in module `virttest.utils_selinux`), 498

`get_defined()` (`virttest.libvirt_xml.network_xml.NetworkXMLBase` method), 183

`get_deps_dir()` (in module `virttest.data_dir`), 308

`get_dev_major_minor()` (in module `virttest.utils_misc`), 471

`get_dev_pts_max_id()` (in module `virttest.utils_misc`), 471

`get_device()` (`virttest.qemu_devices.qbuses.QSparseBus` method), 214

`get_device()` (`virttest.utils_net.Macvtap` method), 483

`get_device_class()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_device_details()` (`virttest.libvirt_vm.VM` method), 326

`get_device_size()` (`virttest.libvirt_vm.VM` method), 326

`get_devices()` (`virttest.libvirt_xml.vm_xml.VMXMLBase` method), 209

`get_devs()` (`virttest.test_setup.PciAssignable` method), 409

`get_dicts()` (`virttest.cartesian_config.Parser` method), 306

`get_direct()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 436

`get_directory_structure()` (in module `virttest.bootstrap`), 293

`get_disk_address()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disk_all()` (`virttest.libvirt_xml.vm_xml.VMXML` method), 204

`get_disk_attr()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disk_blk()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disk_count()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disk_devices()` (`virttest.libvirt_vm.VM` method), 326

`get_disk_serial()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disk_source()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 204

`get_disks()` (`virttest.libvirt_vm.VM` method), 326

`get_download_dir()` (in module `virttest.data_dir`), 309

`get_driver_hardware_id()` (in module `virttest.utils_test`), 276

`get_driver_hardware_id()` (`virttest.tests.unattended_install.UnattendedInstallConfig` method), 257

`get_driver_info()` (`virttest.utils_v2v.WindowsVMCheck` method), 503

`get_driver_type_map()` (`virttest.utils_net.VMNetStyle` class method), 484

`get_e2attrs()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 436

`get_e2generation()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 436

`get_e2label()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 436

`get_e2uuid()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 436

`get_element()` (`virttest.video_maker.GstPythonVideoMaker` method), 508

`get_element_string()` (`virttest.xml_utils.XMLTreeFile` method), 563

`get_emulate_image_name()` (`virttest.lvm.EmulatedLVM` method), 334

`get_encoder_name()` (`virttest.video_maker.GstPythonVideoMaker` method), 508

`get_env_version()` (in module `virttest.utils_env`), 424

`get_event()` (`virttest.qemu_monitor.QMPMonitor` method), 370

`get_events()` (`virttest.qemu_monitor.QMPMonitor` method), 370

`get_feature()` (`virttest.libvirt_xml.capability_xml.CapabilityXML` method), 177

`get_feature()` (`virttest.libvirt_xml.vm_xml.VMCPUXML` method), 198

`get_feature_list()` (`virttest.libvirt_xml.capability_xml.CapabilityXML` method), 178

`get_feature_list()` (`virttest.libvirt_xml.vm_xml.VMCPUXML` method), 198

`get_feature_list()` (`virttest.libvirt_xml.vm_xml.VMFeaturesXML` method), 200

`get_feature_name()` (`virttest.libvirt_xml.capability_xml.CapabilityXML` method), 178

`get_feature_name()` (`virttest.libvirt_xml.vm_xml.VMCPUXML` method), 198

`get_feature_policy()` (`virttest.libvirt_xml.vm_xml.VMCPUXML` method), 198

`get_file_asset()` (in module `virttest.asset`), 290

`get_filename_without_arch()` (`virttest.staging.utils_koji.RPMFileNameInfo` method), 255

`get_filename_without_suffix()` (`virttest.staging.utils_koji.RPMFileNameInfo` method), 255

`get_filesystems_info()` (`virttest.utils_test.libguestfs.VirtTools` method), 260

`get_first_disk_devices()` (`virttest.libvirt_vm.VM` method), 326

`get_first_free_bus()` (`virttest.qemu_devices.qcontainer.DevContainer` method), 216

`get_first_mac_by_name()` (`virttest.libvirt_xml.vm_xml.VMXML` static method), 205

`get_float()` (`virttest.utils_config.SectionlessConfig` method), 414

- get_format() (virttest.qemu_storage.QemuImg method), 378
- get_free_disk() (in module virttest.utils_misc), 471
- get_free_mem() (in module virttest.utils_misc), 471
- get_free_slot() (virttest.qemu_devices.qbuses.QDriveBus method), 213
- get_free_slot() (virttest.qemu_devices.qbuses.QSparseBus method), 214
- get_free_space() (virttest.utils_test.RemoteDiskManager method), 275
- get_fsfreeze_status() (virttest.guest_agent.QemuAgent method), 315
- get_full_pci_id() (in module virttest.utils_misc), 471
- get_generic_service_command_generator() (virttest.staging.service.Factory.FactoryHelper method), 245
- get_generic_service_manager_type() (virttest.staging.service.Factory.FactoryHelper method), 245
- get_generic_service_result_parser() (virttest.staging.service.Factory.FactoryHelper method), 245
- get_git_version() (in module virttest.version), 505
- get_graphics_devices() (virttest.libvirt_xml.vm_xml.VMXMLElement method), 205
- get_greeting() (virttest.qemu_monitor.QMPMonitor method), 370
- get_grub_device() (virttest.utils_v2v.LinuxVMCheck method), 502
- get_guest_capabilities() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 178
- get_guest_ip_addr() (in module virttest.utils_net), 488
- get_guest_name_list() (in module virttest.standalone_test), 402
- get_guest_name_parser() (in module virttest.standalone_test), 402
- get_guest_os_info_list() (in module virttest.bootstrap), 293
- get_hash_from_file() (in module virttest.utils_misc), 471
- get_help_text() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- get_host_cpu_models() (in module virttest.utils_misc), 471
- get_host_default_gateway() (in module virttest.utils_net), 488
- get_host_iface() (in module virttest.utils_net), 488
- get_host_ip_address() (in module virttest.utils_net), 488
- get_host_ipv4_addr() (in module virttest.utils_test.libvirt), 265
- get_huge_page_size() (in module virttest.staging.utils_memory), 256
- get_hugepage_size() (virttest.test_setup.HugePageConfig method), 407
- get_id() (virttest.aexpect.Spawn method), 286
- get_id() (virttest.libvirt_vm.VM method), 326
- get_id() (virttest.utils_libguestfs.GuestfishRemote method), 457
- get_if_vlan_name() (virttest.ovs_utils.Machine method), 350
- get_iface_all() (virttest.libvirt_xml.vm_xml.VMXMLElement method), 205
- get_iface_by_mac() (virttest.libvirt_xml.vm_xml.VMXMLElement static method), 205
- get_iface_dev() (virttest.libvirt_xml.vm_xml.VMXMLElement static method), 205
- get_ifname() (virttest.libvirt_vm.VM method), 327
- get_ifname() (virttest.qemu_vm.VM method), 385
- get_ifname_host() (in module virttest.utils_test.libvirt), 265
- get_iftune_params() (virttest.libvirt_xml.vm_xml.VMXMLElement static method), 205
- get_image_blkdebug_filename() (in module virttest.storage), 404
- get_image_filename() (in module virttest.gluster), 313
- get_image_filename() (in module virttest.storage), 405
- get_image_filename_filesystem() (in module virttest.storage), 405
- get_image_info() (in module virttest.utils_misc), 471
- get_image_info() (in module virttest.utils_test), 276
- get_index() (virttest.utils_net.Interface method), 482
- get_installer() (virttest.installer.InstallerRegistry method), 317
- get_int() (virttest.utils_config.SectionlessConfig method), 482
- get_interfaces() (virttest.libvirt_xml.CapabilityXML method), 184
- get_interface_details() (in module virttest.utils_test.libvirt), 266
- get_interface_mac() (virttest.libvirt_vm.VM method), 327
- get_interfaces() (virttest.libvirt_vm.VM method), 327
- get_iommu_group_devices() (virttest.utils_misc.VFIOController method), 467
- get_ip() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- get_ip() (virttest.utils_net.Interface method), 482
- get_ip_address_by_interface() (in module virttest.utils_net), 488
- get_job_status() (virttest.qemu_vm.VM method), 385
- get_job_type() (virttest.libvirt_vm.VM method), 327
- get_key2filename_dict() (virttest.libvirt_xml.nodedev_xml.CAPXML static method), 185
- get_key2filename_dict() (virttest.libvirt_xml.nodedev_xml.PCIXML static method), 187
- get_key2filename_dict() (virttest.libvirt_xml.nodedev_xml.SystemXML static method), 188
- get_key2syspath_dict() (virttest.libvirt_xml.nodedev_xml.NodedevXML method), 186
- get_key2value_dict() (virttest.libvirt_xml.nodedev_xml.CAPXML

method), 185
get_key2value_dict() (virttest.libvirt_xml.nodedev_xml.NodedevXML method), 186
get_key2value_dict() (virttest.libvirt_xml.nodedev_xml.PCIXML method), 187
get_key2value_dict() (virttest.libvirt_xml.nodedev_xml.SystemXML method), 188
get_known_backends() (in module virttest.asset), 290
get_ksm_feature() (virttest.utils_misc.KSMController method), 464
get_ksmtuned_pid() (virttest.utils_misc.KSMController method), 464
get_kvm_module_list() (in module virttest.arch), 289
get_lexer() (virttest.cartesian_config.Lexer method), 304
get_linkv6_addr() (virttest.ovs_utils.Machine method), 350
get_linux_ifname() (in module virttest.utils_net), 488
get_list() (virttest.utils_config.SectionlessConfig method), 414
get_listens() (virttest.libvirt_xml.devices.graphics.Graphics method), 129
get_load_per_cpu() (in module virttest.staging.utils_cgroup), 250
get_log_file_dir() (in module virttest.utils_misc), 472
get_loss_ratio() (in module virttest.utils_test), 276
get_lvmdev() (virttest.utils_env.Env method), 423
get_mac() (virttest.utils_net.Interface method), 482
get_mac_address() (virttest.ovirt.VMManager method), 348
get_mac_address() (virttest.utils_net.VirtNet method), 486
get_mac_address() (virttest.virt_vm.BaseVM method), 556
get_macvtap_base_iface() (in module virttest.utils_net), 488
get_max_mem() (virttest.libvirt_vm.VM method), 327
get_mbr_id() (virttest.utils_test.libguestfs.GuestfishTools method), 259
get_md5() (virttest.utils_test.libguestfs.GuestfishTools method), 259
get_memory_info() (in module virttest.utils_test), 276
get_memory_size() (virttest.virt_vm.BaseVM method), 556
get_memsizes() (virttest.utils_libguestfs.GuestfishPersistent method), 436
get_modes() (virttest.installer.InstallerRegistry method), 317
get_module_params() (in module virttest.utils_misc), 472
get_monitor_filename() (in module virttest.qemu_monitor), 373
get_monitor_filenames() (in module virttest.qemu_monitor), 373
get_monitors_by_type() (virttest.qemu_vm.VM method), 385
get_most_common_image_size() (virttest.video_maker.GstPythonVideoMaker method), 508
get_multi_supported_hugepage_size() (virttest.test_setup.HugePageConfig method), 407
get_name_of_init() (virttest.staging.service.Factory.FactoryHelper method), 245
get_neigh_attach_interface() (in module virttest.utils_net), 488
get_neigh_mac() (in module virttest.utils_net), 488
get_neighbours_info() (in module virttest.utils_net), 488
get_net_all() (virttest.libvirt_xml.vm_xml.VMXML method), 205
get_net_dev() (virttest.libvirt_xml.vm_xml.VMXML static method), 205
get_net_if() (in module virttest.utils_net), 488
get_net_if_addrs() (in module virttest.utils_net), 489
get_net_if_addrs_win() (in module virttest.utils_net), 489
get_net_if_and_addrs() (in module virttest.utils_net), 489
get_net_if_operstate() (in module virttest.utils_net), 489
get_netmask() (virttest.utils_net.Interface method), 482
get_network() (virttest.utils_libguestfs.GuestfishPersistent method), 436
get_network_restart() (virttest.utils_v2v.WindowsVMCheck method), 503
get_next_check() (virttest.cartesian_config.Lexer method), 304
get_next_check_nw() (virttest.cartesian_config.Lexer method), 304
get_next_line() (virttest.cartesian_config.StrReader method), 307
get_node_cpus() (in module virttest.utils_misc), 472
get_node_cpus() (virttest.utils_misc.NumaNode method), 466
get_node_distance() (virttest.utils_misc.NumaInfo method), 465
get_node_num_huge_pages() (virttest.test_setup.HugePageConfig method), 408
get_nodes() (virttest.qemu_qtree.QtreeContainer method), 374
get_num_anon_huge_pages() (in module virttest.staging.utils_memory), 256
get_num_huge_pages() (in module virttest.staging.utils_memory), 256
get_num_huge_pages_free() (in module virttest.staging.utils_memory), 256
get_num_huge_pages_rsvd() (in module virttest.staging.utils_memory), 256
get_numa_memnode_params() (virttest.libvirt_xml.vm_xml.VMXML static method), 205
get_numa_memory_params()

(virttest.libvirt_xml.vm_xml.VMXML static method), 205
 get_numa_status() (in module virttest.utils_test.qemu), 273
 get_nvr_info() (virttest.staging.utils_koji.RPMFileNameInfo method), 255
 get_online_nodes() (virttest.utils_misc.NumaInfo method), 465
 get_open_fds() (in module virttest.utils_misc), 472
 get_output() (virttest.aexpect.Spawn method), 286
 get_paginator() (in module virttest.standalone_test), 402
 get_param() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
 get_params() (virttest.qemu_qtree.QtreeNode method), 375
 get_params() (virttest.utils_env_unittest.FakeVm method), 424
 get_params() (virttest.utils_net_unittest.FakeVm method), 492
 get_params() (virttest.virt_vm.BaseVM method), 556
 get_parent() (virttest.qemu_qtree.QtreeNode method), 375
 get_parent() (virttest.xml_utils.XMLTreeFile method), 563
 get_parent_map() (virttest.xml_utils.XMLTreeFile method), 563
 get_part_size() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 get_part_type() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 get_partitions_info() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 get_parts_list() (in module virttest.utils_test.libvirt), 266
 get_path() (in module virttest.utils_misc), 472
 get_path() (virttest.utils_libguestfs.GuestfishPersistent method), 436
 get_pci_devices() (virttest.libvirt_vm.VM method), 327
 get_pci_devices_in_group() (in module virttest.utils_misc), 472
 get_pci_group_by_id() (in module virttest.utils_misc), 472
 get_pci_iommu_group_id() (virttest.utils_misc.VFIOController method), 467
 get_pci_vendor_device() (in module virttest.utils_misc), 472
 get_peer() (virttest.qemu_vm.VM method), 385
 get_persistent() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
 get_pf_devs() (virttest.test_setup.PciAssignable method), 409
 get_pf_vf_info() (virttest.test_setup.PciAssignable method), 409
 get_pgroup() (virttest.utils_libguestfs.GuestfishPersistent method), 437
 get_pid() (virttest.aexpect.Spawn method), 286
 get_pid() (virttest.libvirt_vm.VM method), 327
 get_pid() (virttest.qemu_vm.VM method), 385
 get_pid() (virttest.test_setup.EGDConfig method), 407
 get_pid() (virttest.utils_libguestfs.GuestfishPersistent method), 437
 get_pid_cpu() (in module virttest.utils_misc), 472
 get_pid_from_file() (in module virttest.utils_misc), 473
 get_pid_path() (in module virttest.utils_misc), 473
 get_pids() (virttest.staging.utils_cgroup.Cgroup method), 249
 get_pkg_base_url() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkg_info() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkg_rpm_file_names() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkg_rpm_info() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkg_rpm_names() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkg_urls() (virttest.staging.utils_koji.KojiClient method), 251
 get_pkgs() (virttest.staging.utils_koji.KojiClient method), 252
 get_pool_details() (virttest.libvirt_xml.pool_xml.PoolXML static method), 192
 get_pool_uuid() (virttest.libvirt_storage.StoragePool method), 323
 get_port() (virttest.virt_vm.BaseVM method), 556
 get_portgroup() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
 get_power_management_list() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 178
 get_pretty_version_info() (in module virttest.version), 505
 get_primary_disk() (in module virttest.utils_test.libguestfs), 260
 get_primary_disk_fs_type() (virttest.utils_test.libguestfs.VirtTools method), 260
 get_primary_serial() (virttest.libvirt_xml.vm_xml.VMXML method), 205
 get_program() (virttest.utils_libguestfs.GuestfishPersistent method), 437
 get_property() (virttest.staging.utils_cgroup.Cgroup method), 249
 get_protocol() (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules method), 191
 get_protocol_attr() (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase method), 190

`get_pwd()` (`virttest.staging.utils_cgroup.CgroupModules` method), 250

`get_qemu()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 437

`get_qemu_best_cpu_model()` (in module `virttest.utils_misc`), 473

`get_qemu_binary()` (in module `virttest.utils_misc`), 473

`get_qemu_cpu_models()` (in module `virttest.utils_misc`), 473

`get_qemu_dst_binary()` (in module `virttest.utils_misc`), 473

`get_qemu_img_binary()` (in module `virttest.utils_misc`), 473

`get_qemu_io_binary()` (in module `virttest.utils_misc`), 473

`get_qid()` (`virttest.qemu_devices.qdevices.QBaseDevice` method), 220

`get_qname()` (`virttest.qemu_qtree.QtreeDisk` method), 374

`get_qtree()` (`virttest.qemu_qtree.QtreeContainer` method), 374

`get_qtree()` (`virttest.qemu_qtree.QtreeNode` method), 375

`get_raw()` (`virttest.utils_config.SectionlessConfig` method), 414

`get_readable_cdroms()` (in module `virttest.utils_test`), 276

`get_recovery_proc()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 437

`get_region_md5sum()` (in module `virttest.ppm_utils`), 354

`get_root()` (`virttest.utils_test.libguestfs.GuestfishTools` method), 259

`get_root_dir()` (in module `virttest.data_dir`), 309

`get_rule()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase` method), 190

`get_rule_index()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase` method), 190

`get_rules_dict()` (`virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase` method), 189

`get_same_group_devs()` (`virttest.test_setup.PciAssignable` method), 410

`get_scratch_base_url()` (`virttest.staging.utils_koji.KojiClient` method), 252

`get_scratch_pkg_urls()` (`virttest.staging.utils_koji.KojiClient` method), 252

`get_scratch_pkgs()` (`virttest.staging.utils_koji.KojiClient` method), 252

`get_screenshot()` (`virttest.utils_v2v.WindowsVMCheck` method), 503

`get_sebool_local()` (`virttest.utils_misc.SELinuxBoolean` method), 466

`get_sebool_remote()` (`virttest.utils_misc.SELinuxBoolean` method), 466

`get_seclabel()` (`virttest.libvirt_xml.vm_xml.VMXMLBase` method), 209

`get_secret_details_by_uuid()` (`virttest.libvirt_xml.secret_xml.SecretXML` static method), 194

`get_section_string()` (`virttest.libvirt_xml.base.LibvirtXMLBase` method), 176

`get_serial_console_filename()` (`virttest.libvirt_vm.VM` method), 327

`get_serial_console_filename()` (`virttest.qemu_vm.VM` method), 385

`get_serial_console_filenames()` (`virttest.libvirt_vm.VM` method), 327

`get_serial_console_filenames()` (`virttest.qemu_vm.VM` method), 386

`get_server_session()` (`virttest.utils_conn.ConnectionBase` method), 416

`get_service_manager()` (`virttest.service_unittest.TestServiceManager` method), 401

`get_session()` (`virttest.utils_net.IPv6Manager` method), 481

`get_session()` (`virttest.utils_sasl.SASL` method), 497

`get_session()` (`virttest.utils_test.VMStress` method), 276

`get_session_options()` (`virttest.staging.utils_koji.KojiClient` method), 252

`get_shared_meminfo()` (`virttest.libvirt_vm.VM` method), 327

`get_shared_meminfo()` (`virttest.qemu_vm.VM` method), 386

`get_shell_pid()` (`virttest.libvirt_vm.VM` method), 327

`get_shell_pid()` (`virttest.qemu_vm.VM` method), 386

`get_smp()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 437

`get_sorted_net_if()` (in module `virttest.utils_net`), 489

`get_source()` (`virttest.libvirt_xml.pool_xml.PoolXMLBase` method), 193

`get_source()` (`virttest.libvirt_xml.devices.character.CharacterBase` method), 123

`get_specific_service_command_generator()` (`virttest.staging.service.Factory.FactoryHelper` method), 245

`get_specific_service_result_parser()` (`virttest.staging.service.Factory.FactoryHelper` method), 245

`get_spice_var()` (`virttest.qemu_vm.VM` method), 386

`get_state()` (`virttest.qemu_devices.qcontainer.DevContainer` method), 216

`get_stats()` (`virttest.utils_net.Interface` method), 482

`get_status()` (in module `virttest.utils_selinux`), 499

`get_status()` (`virttest.aexpect.Spawn` method), 287

`get_status()` (`virttest.qemu_monitor.HumanMonitor` method), 363

`get_status()` (`virttest.qemu_monitor.QMPMonitor` method), 370

`get_stdout()` (`virttest.nfs_unittest.FakeService` method), 343

`get_stdout()` (`virttest.utils_misc_unittest.FakeCmd`

- method), 479
- get_stdout() (virttest.utils_net_unittest.TestBridge.FakeCmdget_type() (virttest.libvirt_xml.pool_xml.PoolXML static method), 492
- get_stp_status() (virttest.utils_net.Bridge method), 480
- get_string() (virttest.utils_config.SectionlessConfig method), 414
- get_stripped_output() (virttest.aexpect.Spawn method), 287
- get_structure() (virttest.utils_net.Bridge method), 480
- get_style() (virttest.utils_net.VMNetStyle class method), 485
- get_style() (virttest.utils_net_unittest.TestVmNetStyle method), 493
- get_sub_list() (virttest.RFBDes.Des method), 280
- get_support_machine_type() (in module virttest.utils_misc), 473
- get_syncserver() (virttest.utils_env.Env method), 423
- get_sysfs_path() (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase method), 186
- get_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.CAPXML method), 185
- get_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase method), 186
- get_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.PCIXML method), 187
- get_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.SystemXML method), 188
- get_tapname() (virttest.utils_net.Macvtap method), 483
- get_target_account_info() (virttest.iscsi.IscsiTGT method), 319
- get_target_hugepages() (virttest.test_setup.HugePageConfig method), 408
- get_target_id() (virttest.iscsi.IscsiLIO method), 318
- get_target_id() (virttest.iscsi.IscsiTGT method), 319
- get_targets() (virttest.libvirt_xml.devices.character.CharacterXML method), 123
- get_test_entrypoint_func() (in module virttest.utils_misc), 473
- get_test_provider_dir() (in module virttest.data_dir), 309
- get_test_provider_info() (in module virttest.asset), 290
- get_test_provider_names() (in module virttest.asset), 290
- get_test_provider_subdirs() (in module virttest.asset), 290
- get_test_providers_dir() (in module virttest.data_dir), 309
- get_testlog_filename() (virttest.virt_vm.BaseVM method), 556
- get_thread_cpu() (in module virttest.utils_misc), 473
- get_time() (in module virttest.utils_test), 277
- get_tmp_dir() (in module virttest.data_dir), 309
- get_tmp_files() (virttest.xml_utils_unittest.xml_test_data method), 565
- get_top_commit() (in module virttest.version), 505
- get_trace() (virttest.utils_libguestfs.GuestfishPersistent method), 437
- get_transparent_hugepage() (in module virttest.staging.utils_memory), 256
- get_type() (virttest.libvirt_xml.pool_xml.PoolXML static method), 192
- get_type_from_context() (in module virttest.utils_selinux), 499
- get_umask() (virttest.utils_libguestfs.GuestfishPersistent method), 437
- get_until() (virttest.cartesian_config.Lexer method), 304
- get_until_check() (virttest.cartesian_config.Lexer method), 304
- get_until_gen() (virttest.cartesian_config.Lexer method), 305
- get_until_no_white() (virttest.cartesian_config.Lexer method), 305
- get_uri() (virttest.utils_libguestfs.LibguestfsBase method), 458
- get_uri() (virttest.utils_v2v.Uri method), 503
- get_uri_base() (virttest.virsh.VirshBase method), 509
- get_uri_with_transport() (in module virttest.libvirt_vm), 331
- get_url() (virttest.tests.unattended_install.RemoteInstall method), 257
- get_used_mem() (virttest.libvirt_vm.VM method), 327
- get_vmlinux() (virttest.libvirt_vm.VM method), 327
- get_uuid() (virttest.virt_vm.BaseVM method), 556
- get_xml_by_name() (virttest.libvirt_xml.network_xml.NetworkXML static method), 181
- get_validates() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176
- get_vcpu_pids() (virttest.qemu_vm.VM method), 386
- get_vcpus_pid() (virttest.libvirt_vm.VM method), 327
- get_vdagent_status() (in module virttest.utils_spice), 500
- get_vendor_from_pci_id() (in module virttest.utils_misc), 474
- get_verbose() (virttest.utils_libguestfs.GuestfishPersistent method), 437
- get_version() (in module virttest.version), 505
- get_version() (virttest.base_installer.GitRepoInstaller method), 292
- get_version() (virttest.base_installer.KojiInstaller method), 292
- get_version() (virttest.base_installer.YumInstaller method), 293
- get_version() (virttest.openvswitch.OpenVSwitchControl class method), 344
- get_version() (virttest.openvswitch.ServiceManagerInterface class method), 346
- get_vf_devs() (virttest.test_setup.PciAssignable method), 410
- get_vf_num_by_id() (virttest.test_setup.PciAssignable method), 410
- get_vf_status() (virttest.test_setup.PciAssignable method), 410
- get_vfs_count() (virttest.test_setup.PciAssignable

method), 410
get_vhost_threads() (virttest.qemu_vm.VM method), 386
get_viosstor_info() (virttest.utils_v2v.WindowsVMCheck method), 503
get_virsh_mac_address() (virttest.libvirt_vm.VM method), 327
get_virt_test_open_fds() (in module virttest.utils_misc), 474
get_virtio_port_filename() (virttest.virt_vm.BaseVM method), 556
get_virtio_port_filenames() (virttest.qemu_vm.VM method), 386
get_virtio_port_filenames() (virttest.virt_vm.BaseVM method), 556
get_virtio_ports() (virttest.utils_virtio_port.VirtioPortTest method), 504
get_vlans_ifname() (virttest.ovs_utils.Machine method), 350
get_vm() (virttest.utils_env.Env method), 423
get_vm_info_with_inspector() (virttest.utils_test.libguestfs.VirtTools method), 260
get_vm_kernel() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_modprobe_conf() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_modules() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_os_info() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_os_vendor() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_parted() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_pci_list() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_rc_local() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_tty() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_type_map() (virttest.utils_net.VMNetStyle class method), 485
get_vm_video() (virttest.utils_v2v.LinuxVMCheck method), 502
get_vm_with_ports() (virttest.utils_virtio_port.VirtioPortTest method), 504
get_vm_with_single_port() (virttest.utils_virtio_port.VirtioPortTest method), 505
get_vm_with_worker() (virttest.utils_virtio_port.VirtioPortTest method), 505
get_vnc_port() (virttest.qemu_vm.VM method), 386
get_vol() (virttest.lvm.LVM method), 335
get_vol_details_by_name() (virttest.libvirt_xml.vol_xml.VolXML static method), 211
get_windows_disk_drive() (in module virttest.utils_test), 277
get_windows_drive_letters() (in module virttest.utils_misc), 474
get_windows_event_info() (virttest.utils_v2v.WindowsVMCheck method), 504
get_windows_file_abs_path() (in module virttest.utils_test), 277
get_windows_nic_attribute() (in module virttest.utils_net), 489
get_winutils_vol() (in module virttest.utils_misc), 474
get_writable_features() (virttest.utils_misc.KSMController method), 464
get_xml() (virttest.libvirt_vm.VM method), 327
get_xml() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176
get_xmltreefile() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176
get_xpath() (virttest.xml_utils.XMLTreeFile method), 563
get_xpath_elements() (virttest.xml_utils_unittest.test_XMLTreeFile method), 564
getbasecmd() (virttest.remote_commander.remote_master.CmdMaster method), 227
getcls() (virttest.versionable_class.Manager method), 506
getenforce() (virttest.libvirt_vm.VM method), 327
getfd() (virttest.qemu_monitor.HumanMonitor method), 363
getfd() (virttest.qemu_monitor.QMPMonitor method), 370
getiterator() (virttest.element_tree.ElementTree method), 310
getKey() (virttest.RFBDes.Des method), 279
getroot() (virttest.element_tree.ElementTree method), 310
getsource() (in module virttest.remote_commander.remote_master), 229
getstderr() (virttest.remote_commander.remote_master.CmdMaster method), 227
getstdout() (virttest.remote_commander.remote_master.CmdMaster method), 227
GitRepoInstaller (class in virttest.base_installer), 292
GitRepoInstaller (class in virttest.qemu_installer), 359
GitRepoParamHelper (class in virttest.build_helper), 294
glob() (virttest.utils_libguestfs.GuestfishPersistent method), 437
glob_expand() (virttest.utils_libguestfs.GuestfishPersistent method), 437
gluster_brick_create() (in module virttest.gluster), 313

- gluster_brick_delete() (in module virttest.gluster), 313
- GlusterBrickError, 313
- GlusterError, 313
- glusterfs_mount() (in module virttest.gluster), 313
- GnuSourceBuildHelper (class in virttest.build_helper), 294
- GnuSourceBuildInvalidSource, 295
- GnuSourceBuildParamHelper (class in virttest.build_helper), 296
- graceful_shutdown() (virttest.qemu_vm.VM method), 386
- Graphics (class in virttest.libvirt_xml.devices.graphics), 128
- gratuitous (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attr attribute), 144
- gratuitous (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr attribute), 155
- grep() (virttest.utils_libguestfs.GuestfishPersistent method), 437
- grepi() (virttest.utils_libguestfs.GuestfishPersistent method), 437
- group (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
- group (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212
- grub_install() (virttest.utils_libguestfs.GuestfishPersistent method), 438
- GstPythonVideoMaker (class in virttest.video_maker), 508
- guess_type() (virttest.qemu_qtree.QtreeBus method), 374
- guess_type() (virttest.qemu_qtree.QtreeDev method), 374
- guess_type() (virttest.qemu_qtree.QtreeNode method), 375
- guest_active() (in module virttest.utils_test.qemu), 273
- guest_capabilities (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 178
- Guestfish (class in virttest.utils_libguestfs), 427
- GuestfishPersistent (class in virttest.utils_libguestfs), 427
- GuestfishRemote (class in virttest.utils_libguestfs), 457
- GuestfishSession (class in virttest.utils_libguestfs), 457
- GuestfishTools (class in virttest.utils_test.libguestfs), 259
- GuestFSModiDisk (class in virttest.utils_disk), 421
- guestmount() (in module virttest.utils_libguestfs), 458
- guestmount() (virttest.utils_test.libguestfs.VirtTools method), 260
- GuestSuspend (class in virttest.utils_test.qemu), 269
- GuestWorker (class in virttest.qemu_virtio_port), 379
- ## H
- handle() (virttest.syslog_server.RequestHandlerTcp method), 406
- handle() (virttest.syslog_server.RequestHandlerUdp method), 406
- handle_prompts() (in module virttest.remote), 393
- handle_starttag() (virttest.staging.utils_koji.KojiDirIndexParser method), 253
- hard_limit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- hard_limit_unit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- has_command_help_match() (in module virttest.virsh), 524
- has_device() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- has_element() (virttest.video_maker.GstPythonVideoMaker method), 508
- has_feature() (virttest.libvirt_xml.vm_xml.VMFeaturesXML method), 200
- has_help_command() (in module virttest.virsh), 524
- has_hmp_cmd() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- has_key() (virttest.propcan.PropCan method), 357
- has_option() (virttest.qemu_devices.qcontainer.DevContainer method), 216
- has_qmp_cmd() (virttest.qemu_devices.qcontainer.DevContainer method), 217
- has_swap() (virttest.libvirt_vm.VM method), 328
- has_vmware_tools() (virttest.utils_v2v.LinuxVMCheck method), 502
- hash_name() (virttest.cartesian_config.Label method), 304
- hash_val (virttest.cartesian_config.Label attribute), 304
- hash_var (virttest.cartesian_config.Label attribute), 304
- hash_variant() (virttest.cartesian_config.Label method), 304
- have_similar_img() (in module virttest.ppm_utils), 354
- haz_defcon() (in module virttest.bootstrap), 293
- hardware_serial (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- hardware_uuid (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- hardware_vendor (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188
- head() (virttest.utils_libguestfs.GuestfishPersistent method), 438
- head_n() (virttest.utils_libguestfs.GuestfishPersistent method), 438
- hello_time (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- hello_time_hi (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- help() (in module virttest.virsh), 524
- help() (virttest.utils_libguestfs.GuestfishPersistent method), 438
- help_command() (in module virttest.virsh), 524
- help_command_group() (in module virttest.virsh), 524
- help_command_only() (in module virttest.virsh), 525
- hexdump() (virttest.utils_libguestfs.GuestfishPersistent

- method), 438
- hook_fill_scsi_hbas() (virttest.qemu_devices.qcontainer.DevContainer.cpu() (virttest.qemu_vm.VM method), 386
- method), 217
- hotplug_verified() (virttest.qemu_devices.qcontainer.DevContainer method), 217
- host (virttest.libvirt_xml.network_xml.DNSXML attribute), 180
- hotunplug_nic() (virttest.qemu_vm.VM method), 386
- host (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
- HPNotSupportedError, 407
- host_ip (virttest.libvirt_xml.network_xml.DNSXML.HostXML attribute), 180
- http_server() (in module virttest.http_server), 317
- host_name (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- HTTPRequestHandler (class in virttest.http_server), 316
- Hostdev (class in virttest.libvirt_xml.devices.hostdev), 130
- Hub (class in virttest.libvirt_xml.devices.hub), 130
- Hostdev.SourceAddress (class in virttest.libvirt_xml.devices.hostdev), 130
- Hub.Address (class in virttest.libvirt_xml.devices.hub), 130
- Hostdev.SourceAddress.UntypedAddress (class in virttest.libvirt_xml.devices.hostdev), 130
- hugepages (virttest.libvirt_xml.vm_xml.VMMemBackingXML attribute), 201
- human_monitor_cmd() (virttest.qemu_monitor.HumanMonitor method), 363
- human_monitor_cmd() (virttest.qemu_monitor.Monitor method), 366
- human_monitor_cmd() (virttest.qemu_monitor.QMPMonitor method), 370
- hostname (virttest.libvirt_xml.network_xml.DNSXML.HostXML attribute), 180
- HumanMonitor (class in virttest.qemu_monitor), 361
- hostname() (in module virttest.virsh), 525
- HwAddrGetError, 480
- hostnames (virttest.libvirt_xml.network_xml.DNSXML.HostXML attribute), 180
- HwAddrSetError, 481
- hosts (virttest.libvirt_xml.devices.disk.Disk.DiskSource attribute), 126
- HWOpStartError, 481
- hosts (virttest.libvirt_xml.network_xml.IPXML attribute), 181
- hwtype (virttest.libvirt_xml.nwfilter_protocols arp.Arp.Attr attribute), 144
- hosts (virttest.libvirt_xml.pool_xml.SourceXML attribute), 193
- hwtype (virttest.libvirt_xml.nwfilter_protocols rarp.Rarp.Attr attribute), 155
- HostStress (class in virttest.utils_test), 275
- hyperv_relaxed_state (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- hotplug() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- hyperv_spinlocks_retries (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- hotplug_domain_vcpu() (in module virttest.utils_test.libvirt), 266
- hyperv_spinlocks_state (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- hotplug_hmp() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- hyperv_vapic_state (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- hotplug_hmp() (virttest.qemu_devices.qdevices.QDevice method), 221
- hypervisor_type (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- hotplug_hmp() (virttest.qemu_devices.qdevices.QHPDrive method), 222
- I
- hotplug_hmp() (virttest.qemu_devices.qdevices.QRHD Drive method), 223
- Icmp (class in virttest.libvirt_xml.nwfilter_protocols.icmp), 148
- hotplug_hmp_nd() (virttest.qemu_devices.qdevices.QDevice method), 221
- Icmp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.icmp), 148
- hotplug_nic() (virttest.qemu_vm.VM method), 386
- Icmpv6 (class in virttest.libvirt_xml.nwfilter_protocols.icmpv6), 149
- hotplug_qmp() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- Icmpv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.icmpv6), 149
- hotplug_qmp() (virttest.qemu_devices.qdevices.QDevice method), 221
- identifier (virttest.cartesian_config.LAnd attribute), 301
- hotplug_qmp() (virttest.qemu_devices.qdevices.QRHD Drive method), 223
- identifier (virttest.cartesian_config.LAppend attribute), 301
- hotplug_qmp_nd() (virttest.qemu_devices.qdevices.QDevice method), 221
- identifier (virttest.cartesian_config.LApplyPreDict attribute), 301

- identifier (virttest.cartesian_config.LCoc attribute), 301
- identifier (virttest.cartesian_config.LColon attribute), 301
- identifier (virttest.cartesian_config.LComa attribute), 301
- identifier (virttest.cartesian_config.LCond attribute), 301
- identifier (virttest.cartesian_config.LDefault attribute), 301
- identifier (virttest.cartesian_config.LDel attribute), 301
- identifier (virttest.cartesian_config.LDot attribute), 301
- identifier (virttest.cartesian_config.LEndL attribute), 301
- identifier (virttest.cartesian_config.LIdentifier attribute), 302
- identifier (virttest.cartesian_config.LInclude attribute), 302
- identifier (virttest.cartesian_config.LIndent attribute), 302
- identifier (virttest.cartesian_config.LLBracket attribute), 302
- identifier (virttest.cartesian_config.LLRBracket attribute), 302
- identifier (virttest.cartesian_config.LNo attribute), 302
- identifier (virttest.cartesian_config.LNotCond attribute), 302
- identifier (virttest.cartesian_config.LOnly attribute), 302
- identifier (virttest.cartesian_config.LOperators attribute), 302
- identifier (virttest.cartesian_config.LOr attribute), 303
- identifier (virttest.cartesian_config.LPrepend attribute), 303
- identifier (virttest.cartesian_config.LRBracket attribute), 303
- identifier (virttest.cartesian_config.LRegExpAppend attribute), 303
- identifier (virttest.cartesian_config.LRegExpPrepend attribute), 303
- identifier (virttest.cartesian_config.LRegExpSet attribute), 303
- identifier (virttest.cartesian_config.LRegExpStart attribute), 303
- identifier (virttest.cartesian_config.LRegExpStop attribute), 303
- identifier (virttest.cartesian_config.LRRBracket attribute), 303
- identifier (virttest.cartesian_config.LSet attribute), 303
- identifier (virttest.cartesian_config.LString attribute), 303
- identifier (virttest.cartesian_config.LUpdateFileMap attribute), 304
- identifier (virttest.cartesian_config.LVariant attribute), 304
- identifier (virttest.cartesian_config.LVariants attribute), 304
- identifier (virttest.cartesian_config.LWhite attribute), 304
- identifier (virttest.cartesian_config.Token attribute), 307
- idx_of_next_named_bus() (virttest.qemu_devices.qcontainer.DevContainer method), 217
- if_nametoindex() (in module virttest.utils_net), 489
- if_set_macaddress() (in module virttest.utils_net), 489
- iface_begin() (in module virttest.virsh), 525
- iface_bridge() (in module virttest.virsh), 525
- iface_commit() (in module virttest.virsh), 525
- iface_define() (in module virttest.virsh), 525
- iface_destroy() (in module virttest.virsh), 526
- iface_dumpxml() (in module virttest.virsh), 526
- iface_edit() (in module virttest.virsh), 526
- iface_list() (in module virttest.virsh), 526
- iface_mac() (in module virttest.virsh), 526
- iface_name() (in module virttest.virsh), 526
- iface_rollback() (in module virttest.virsh), 527
- iface_start() (in module virttest.virsh), 527
- iface_unbridge() (in module virttest.virsh), 527
- iface_undefine() (in module virttest.virsh), 527
- IfChangeAddrError, 482
- IfChangeBrError, 482
- ifname (virttest.utils_net.Qemuiface attribute), 483
- IfNotInBridgeError, 482
- Igmp (class in virttest.libvirt_xml.nwfilter_protocols.igmp), 150
- Igmp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.igmp), 150
- ignore_status (virttest.utils_libguestfs.LibguestfsBase attribute), 458
- ignore_status (virttest.virsh.VirshBase attribute), 509
- image_average_hash() (in module virttest.ppm_utils), 354
- image_comparison() (in module virttest.ppm_utils), 354
- image_compression (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129
- image_crop() (in module virttest.ppm_utils), 354
- image_crop_save() (in module virttest.ppm_utils), 355
- image_fuzzy_compare() (in module virttest.ppm_utils), 355
- image_histogram_compare() (in module virttest.ppm_utils), 355
- image_md5sum() (in module virttest.ppm_utils), 355
- image_read_from_ppm_file() (in module virttest.ppm_utils), 355
- image_verify_ppm_file() (in module virttest.ppm_utils), 355
- image_write_to_ppm_file() (in module virttest.ppm_utils), 356
- images_define_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 217
- images_define_by_variables() (virttest.qemu_devices.qcontainer.DevContainer method), 217
- ImageUnbootableError, 381
- img_ham_distance() (in module virttest.ppm_utils), 356
- img_similar() (in module virttest.ppm_utils), 356

`import_from_export_domain()`
(`virttest.ovirt.VMManager` method), 349

`import_passfd()` (in module `virttest.passfd_setup`), 351

`import_src()` (`virttest.remote_commander.remote_runner.Commander` method), 230

`import_vm_to_ovirt()` (in module `virttest.utils_v2v`), 504

`inbound` (`virttest.libvirt_xml.devices.interface.Interface.Bandwidth` attribute), 132

`include_pkg_config_path()`
(`virttest.build_helper.GnuSourceBuildHelper` method), 295

`IncompatibleTypeError`, 374

`index` (`virttest.libvirt_xml.devices.controller.Controller` attribute), 124

`info()` (`virttest.qemu_monitor.HumanMonitor` method), 363

`info()` (`virttest.qemu_monitor.Monitor` method), 366

`info()` (`virttest.qemu_monitor.QMPMonitor` method), 370

`info()` (`virttest.qemu_storage.QemuImg` method), 378

`info_block()` (`virttest.qemu_monitor.Monitor` method), 366

`info_numa()` (`virttest.qemu_monitor.Monitor` method), 366

`InfoBlocks` (class in `virttest.qemu_monitor_unittest`), 373

`InfoNumaTests` (class in `virttest.qemu_monitor_unittest`), 373

`init` (`virttest.libvirt_xml.vm_xml.VMOSXML` attribute), 202

`init()` (`virttest.staging.utils_cgroup.CgroupModules` method), 250

`init_db()` (`virttest.openvswitch.OpenVSwitch` method), 344

`init_new()` (`virttest.openvswitch.OpenVSwitch` method), 344

`init_sandboxes()` (`virttest.lvsb_base.TestSandboxes` method), 341

`init_system()` (`virttest.openvswitch.OpenVSwitchSystem` method), 345

`initargs` (`virttest.libvirt_xml.vm_xml.VMOSXML` attribute), 202

`initialize()` (`virttest.staging.utils_cgroup.Cgroup` method), 249

`INITIALIZED` (`virttest.propcan.PropCanBase` attribute), 357

`initrd` (`virttest.libvirt_xml.vm_xml.VMOSXML` attribute), 202

`initrd_cat()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`initrd_list()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inject_nmi()` (in module `virttest.virsh`), 527

`inner_cmd()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`Input` (class in `virttest.libvirt_xml.devices.input`), 131

`Input.Address` (class in `virttest.libvirt_xml.devices.input`), 131

`input_bus` (`virttest.libvirt_xml.devices.input.Input` attribute), 131

`insert()` (`virttest.qemu_devices.qbuses.QSparseBus` method), 214

`insert()` (`virttest.qemu_devices.qcontainer.DevContainer` method), 218

`insert_break()` (`virttest.utils_gdb.GDB` method), 426

`insert_break()` (`virttest.utils_libvirtd.LibvirtdSession` method), 462

`inspect_get_arch()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inspect_get_distro()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inspect_get_filesystems()`
(`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inspect_get_hostname()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inspect_get_major_version()`
(`virttest.utils_libguestfs.GuestfishPersistent` method), 438

`inspect_get_minor_version()`
(`virttest.utils_libguestfs.GuestfishPersistent` method), 439

`inspect_get_mountpoints()`
(`virttest.utils_libguestfs.GuestfishPersistent` method), 439

`inspect_get_roots()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 439

`inspect_os()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 439

`install()` (`virttest.base_installer.BaseInstaller` method), 291

`install()` (`virttest.base_installer.NoopInstaller` method), 292

`install()` (`virttest.build_helper.GnuSourceBuildHelper` method), 295

`install()` (`virttest.build_helper.LinuxKernelBuildHelper` method), 296

`install()` (`virttest.test_setup.EGDConfig` method), 407

`install()` (`virttest.utils_netperf.NetperfPackage` method), 495

`install_disktest_on_vm()` (in module `virttest.utils_misc`), 474

`install_host_kernel()` (in module `virttest.utils_misc`), 474

`install_package()` (`virttest.libvirt_vm.VM` method), 328

`install_rv_win()` (in module `virttest.utils_spice`), 501

`install_usbclerk_win()` (in module `virttest.utils_spice`), 501

`installer_test` (class in `virttest.installer_unittest`), 318

`InstallerRegistry` (class in `virttest.installer`), 317

`instances` (`virttest.lvsb_base.SandboxBase` attribute), 339

- int_list_to_mac_str() (virttest.utils_net.VirtIface class method), 485
- int_list_to_mac_str() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 493
- interactive() (virttest.remote_commander.remote_runner.CommanderStateCmds method), 230
- Interface (class in virttest.libvirt_xml.devices.interface), 131
- Interface (class in virttest.utils_net), 482
- interface (virttest.libvirt_xml.nodedev_xml.NetXML attribute), 185
- Interface.Address (class in virttest.libvirt_xml.devices.interface), 131
- Interface.Bandwidth (class in virttest.libvirt_xml.devices.interface), 132
- Interface.Driver (class in virttest.libvirt_xml.devices.interface), 132
- Interface.Filterref (class in virttest.libvirt_xml.devices.interface), 132
- iothreads (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
- iotune (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- iotune (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- IOWrapper (class in virttest.remote_commander.messenger), 224
- IOzoneAnalyzer (class in virttest.postprocess_iozone), 351
- IOzonePlotter (class in virttest.postprocess_iozone), 352
- Ip (class in virttest.libvirt_xml.nwfilter_protocols.ip), 151
- ip (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- IP (virttest.RFBDes.Des attribute), 279
- ip (virttest.utils_net.VirtIface attribute), 485
- ip (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
- Ip.Attr (class in virttest.libvirt_xml.nwfilter_protocols.ip), 152
- ip_link_ctl() (virttest.utils_net.Macvtap method), 483
- ip_protocol (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr attribute), 152
- ip_protocol (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr attribute), 153
- IPAddress (class in virttest.utils_net), 481
- IPAddrGetError, 481
- ipset (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
- ipset (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 140
- ipset (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 142
- ipset (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
- ipset (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
- ipset (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147
- ipset (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 148
- ipset (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 150
- ipset (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
- ipset (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 157
- ipset (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
- ipset (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
- ipset (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
- ipset (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163
- ipset (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr attribute), 165
- ipset (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr attribute), 166
- ipset (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr attribute), 167
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr attribute), 139
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr attribute), 140
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr attribute), 142
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr attribute), 143
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr attribute), 146
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr attribute), 147
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 149
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 150
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr attribute), 151
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr attribute), 157
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr attribute), 158
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr attribute), 161
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr attribute), 162
- ipsetflags (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr attribute), 163

ipsetflags (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr method), 439
attribute), 165 is_dir_opts() (virttest.utils_libguestfs.GuestfishPersistent
ipsetflags (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr method), 439
attribute), 166 is_disabled() (in module virttest.utils_selinux), 499
ipsetflags (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr (virttest.utils_v2v.LinuxVMCheck
attribute), 167 method), 502
Ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.ipv6), is_dst() (virttest.utils_test.qemu.MigrationData method),
153 270
Ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.ipv6), is_enforcing() (in module virttest.utils_selinux), 499
153 is_esx() (virttest.libvirt_vm.VM method), 328
ipv6_from_mac_addr() (in module virttest.utils_net), 490 is_exported() (virttest.nfs.Exportfs method), 342
IPv6Manager (class in virttest.utils_net), 481 is_fifo() (virttest.utils_libguestfs.GuestfishPersistent
IPXML (class in virttest.libvirt_xml.network_xml), 180 method), 440
is_alive() (in module virttest.virsh), 527 is_fifo_opts() (virttest.utils_libguestfs.GuestfishPersistent
is_alive() (virttest.aexpect.Spawn method), 287 method), 440
is_alive() (virttest.libvirt_vm.VM method), 328 is_file() (virttest.utils_libguestfs.GuestfishPersistent
is_alive() (virttest.ovirt.VMManager method), 349 method), 440
is_alive() (virttest.qemu_vm.VM method), 386 is_file_opts() (virttest.utils_libguestfs.GuestfishPersistent
is_alive() (virttest.utils_env_unittest.FakeVm method), method), 440
424 is_finished() (virttest.remote_commander.remote_interface.BaseCmd
is_alive() (virttest.utils_net_unittest.FakeVm method), method), 226
492 is_installed() (virttest.openvswitch.OpenVSwitchSystem
is_alive() (virttest.utils_test.BackgroundTest method), method), 346
275 is_irrelevant() (virttest.cartesian_config.NoFilter
is_alive() (virttest.virt_vm.BaseVM method), 556 method), 305
is_async() (virttest.remote_commander.remote_interface.BaseCmd method), 306
method), 226 is_ksm_running() (virttest.utils_misc.KSMController
is_autostart() (virttest.libvirt_vm.VM method), 328 method), 464
is_bound_to_stub() (virttest.test_setup.PciAssignable is_lv() (virttest.utils_libguestfs.GuestfishPersistent
method), 410 method), 440
is_blockdev() (virttest.utils_libguestfs.GuestfishPersistent is_lxc() (virttest.libvirt_vm.VM method), 328
method), 439 inmodule_loaded() (virttest.utils_misc.KSMController
is_blockdev_opts() (virttest.utils_libguestfs.GuestfishPersistent method), 464
method), 439 is_mount() (in module virttest.utils_disk), 422
is_brport() (virttest.utils_net.Interface method), 482 is_mounted() (in module virttest.utils_misc), 474
is_cgroup() (virttest.staging.utils_cgroup.Cgroup is_mounted() (virttest.nfs.Nfs method), 343
method), 249 is_mounted() (virttest.nfs.NFSCliet method), 342
is_chardev() (virttest.utils_libguestfs.GuestfishPersistent is_net_virtio() (virttest.utils_v2v.LinuxVMCheck
method), 439 method), 503
is_chardev_opts() (virttest.utils_libguestfs.GuestfishPersistent is_netperf_running() (virttest.utils_netperf.NetperfClient
method), 439 method), 495
is_command_valid() (virttest.staging.utils_koji.KojiClient is_not_disabled() (in module virttest.utils_selinux), 499
method), 252 is_paused() (virttest.libvirt_vm.VM method), 328
is_config() (virttest.utils_libguestfs.GuestfishPersistent is_paused() (virttest.ovirt.VMManager method), 349
method), 439 is_paused() (virttest.qemu_vm.VM method), 387
is_config_valid() (virttest.staging.utils_koji.KojiClient is_paused() (virttest.virt_vm.BaseVM method), 556
method), 252 is_permissive() (in module virttest.utils_selinux), 499
is_dead() (in module virttest.virsh), 527 is_persistent() (virttest.libvirt_vm.VM method), 328
is_dead() (virttest.libvirt_vm.VM method), 328 is_pkg_spec_build_valid()
is_dead() (virttest.ovirt.VMManager method), 349 (virttest.staging.utils_koji.KojiClient method),
is_dead() (virttest.qemu_vm.VM method), 386 252
is_dead() (virttest.virt_vm.BaseVM method), 556 is_pkg_spec_tag_valid() (virttest.staging.utils_koji.KojiClient
is_defunct() (virttest.aexpect.Spawn method), 287 method), 253
is_dir() (virttest.utils_libguestfs.GuestfishPersistent

- `is_pkg_valid()` (virttest.staging.utils_koji.KojiClient method), 253
- `is_pool_active()` (virttest.libvirt_storage.StoragePool method), 323
- `is_pool_persistent()` (virttest.libvirt_storage.StoragePool method), 323
- `is_port_free()` (in module virttest.utils_misc), 475
- `is_qemu()` (virttest.libvirt_vm.VM method), 328
- `is_ready()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_remote_image()` (virttest.storage.QemuImg method), 404
- `is_responsive()` (virttest.aexpect.ShellSession method), 285
- `is_responsive()` (virttest.qemu_monitor.Monitor method), 366
- `is_root_cgroup()` (virttest.staging.utils_cgroup.Cgroup method), 249
- `is_running()` (virttest.lvsb_base.SandboxSession method), 340
- `is_running()` (virttest.utils_libvirtd.Libvirtd method), 462
- `is_same_contents()` (virttest.xml_utils_unittest.xml_test_data method), 565
- `is_server_running()` (virttest.utils_netperf.NetperfServer method), 495
- `is_socket()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_socket_opts()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_src()` (virttest.utils_test.qemu.MigrationData method), 270
- `is_symlink()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_target_running()` (virttest.utils_netperf.Netperf method), 494
- `is_up()` (virttest.utils_net.Interface method), 482
- `is_valid()` (virttest.staging.utils_koji.KojiPkgSpec method), 254
- `is_virtual()` (virttest.ovs_utils.Machine method), 350
- `is_virtual_network_dev()` (in module virttest.utils_net), 490
- `is_whole_device()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_working()` (virttest.utils_libvirtd.LibvirtdSession method), 462
- `is_xen()` (virttest.libvirt_vm.VM method), 328
- `is_zero()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `is_zero_device()` (virttest.utils_libguestfs.GuestfishPersistent method), 440
- `isclass()` (in module virttest.versionable_class), 506
- `isCmdMsg()` (virttest.remote_commander.remote_interface.CmdMessage method), 227
- `Iscsi` (class in virttest.iscsi), 318
- `iscsi_discover()` (in module virttest.iscsi), 319
- `iscsi_get_nodes()` (in module virttest.iscsi), 319
- `iscsi_get_sessions()` (in module virttest.iscsi), 319
- `iscsi_login()` (in module virttest.iscsi), 319
- `iscsi_login_setup()` (virttest.utils_test.RemoteDiskManager method), 275
- `iscsi_logout()` (in module virttest.iscsi), 319
- `iscsi_node_del()` (in module virttest.iscsi), 319
- `iscsi_test` (class in virttest.iscsi_unittest), 320
- `Iscsidev` (class in virttest.qemu_storage), 377
- `Iscsidev` (class in virttest.storage), 403
- `IscsiLIO` (class in virttest.iscsi), 318
- `IscsiTGT` (class in virttest.iscsi), 319
- `iselement()` (in module virttest.element_tree), 310
- `item_separator` (virttest.staging.backports.simplejson.encoder.JSONEncoder attribute), 234
- `item_separator` (virttest.staging.backports.simplejson.JSONEncoder attribute), 242
- `items()` (virttest.propcan.PropCan method), 357
- `items()` (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- `items()` (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 235
- `items()` (virttest.staging.backports.simplejson.OrderedDict method), 242
- `iter` (virttest.utils_net_unittest.TestBridge.FakeCmd attribute), 492
- `iterencode()` (virttest.staging.backports.simplejson.encoder.JSONEncoder method), 235
- `iterencode()` (virttest.staging.backports.simplejson.encoder.JSONEncoder method), 235
- `iterencode()` (virttest.staging.backports.simplejson.JSONEncoder method), 242
- `iteritems()` (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- `iteritems()` (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- `iteritems()` (virttest.staging.backports.simplejson.OrderedDict method), 242
- `iterkeys()` (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- `iterkeys()` (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- `iterkeys()` (virttest.staging.backports.simplejson.OrderedDict method), 242
- `iterparse` (class in virttest.element_tree), 310
- `itervalues()` (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 231
- `itervalues()` (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- `itervalues()` (virttest.staging.backports.simplejson.OrderedDict method), 242

J

join() (virttest.utils_test.BackgroundTest method), 275

jpeg_compression (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129

JSONDecodeError, 241

JSONDecoder (class in virttest.staging.backports.simplejson), 240

JSONDecoder (class in virttest.staging.backports.simplejson.decoder), 233

JSONEncoder (class in virttest.staging.backports.simplejson), 241

JSONEncoder (class in virttest.staging.backports.simplejson.encoder), 234

JSONEncoderForHTML (class in virttest.staging.backports.simplejson.encoder), 235

K

kargs (virttest.remote_commander.remote_interface.BaseCmd attribute), 226

kernel (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202

key (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212

key_separator (virttest.staging.backports.simplejson.encoder.JSONEncoder attribute), 235

key_separator (virttest.staging.backports.simplejson.JSONEncoder attribute), 242

keys() (virttest.propcan.PropCan method), 357

keys() (virttest.staging.backports.collections.OrderedDictOrderedDict method), 232

keys() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236

keys() (virttest.staging.backports.simplejson.OrderedDict method), 242

khugepaged_test() (virttest.test_setup.TransparentHugePageConfig method), 412

kill() (virttest.aexpect.Spawn method), 287

kill() (virttest.utils_gdb.GDB method), 426

kill() (virttest.utils_libvirtd.LibvirtdSession method), 462

kill_app() (in module virttest.utils_spice), 501

kill_bg_program() (virttest.utils_test.qemu.GuestSuspend method), 269

kill_process_by_pattern() (in module virttest.utils_misc), 475

kill_process_tree() (in module virttest.utils_misc), 475

kill_session() (virttest.lvsb_base.SandboxSession method), 340

kill_subprocess() (virttest.utils_libguestfs.GuestfishPersistent method), 441

kill_tail_threads() (in module virttest.aexpect), 288

KojiClient (class in virttest.staging.utils_koji), 251

KojiDirIndexParser (class in virttest.staging.utils_koji), 253

KojiDownloadError, 253

KojiInstaller (class in virttest.base_installer), 292

KojiPkgSpec (class in virttest.staging.utils_koji), 253

KojiScratchPkgSpec (class in virttest.staging.utils_koji), 254

kosher_args() (virttest.virsh.VirshConnectBack static method), 509

KSMConfig (class in virttest.test_setup), 408

KSMController (class in virttest.utils_misc), 464

KSMError, 464

KSMNotSupportedError, 465

KSMTunedError, 465

KSMTunedNotSupportedError, 465

kvm_flags_to_stresstests() (in module virttest.utils_misc), 475

kvm_hidden_state (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200

KVMInternalError, 381

KvmQtreeClassTest (class in virttest.qemu_qtree_unittest), 376

L

Label (class in virttest.cartesian_config), 304

label (virttest.libvirt_xml.devices.seclabel.Seclabel attribute), 136

label (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212

labels (virttest.cartesian_config.Node attribute), 306

LAnd (class in virttest.cartesian_config), 300

LAppend (class in virttest.cartesian_config), 301

LApplyPreDict (class in virttest.cartesian_config), 301

LASTBYTE (virttest.utils_net.VirtIface attribute), 485

LASTBYTE (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 492

launch() (virttest.utils_libguestfs.GuestfishPersistent method), 441

launch() (virttest.utils_test.BackgroundTest method), 275

lazy_refcounts (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212

lcd() (virttest.utils_libguestfs.GuestfishPersistent method), 441

lchown() (virttest.utils_libguestfs.GuestfishPersistent method), 441

LCoc (class in virttest.cartesian_config), 301

LColon (class in virttest.cartesian_config), 301

LComa (class in virttest.cartesian_config), 301

LCond (class in virttest.cartesian_config), 301

LDefault (class in virttest.cartesian_config), 301

LDel (class in virttest.cartesian_config), 301

LDot (class in virttest.cartesian_config), 301

Lease (in module virttest.libvirt_xml.devices.lease), 133

left_rotations (virttest.RFBDes.Des attribute), 280

- LEndBlock (class in virttest.cartesian_config), 301
- LEndL (class in virttest.cartesian_config), 301
- length (virttest.cartesian_config.LIndent attribute), 302
- Lexer (class in virttest.cartesian_config), 304
- LexerError, 305
- lgf_cmd_check() (in module virttest.utils_libguestfs), 458
- lgf_command() (in module virttest.utils_libguestfs), 458
- lgf_exec (virttest.utils_libguestfs.LibguestfsBase attribute), 458
- libguest_test_tool_cmd() (in module virttest.utils_libguestfs), 458
- LibguestfsBase (class in virttest.utils_libguestfs), 458
- LibguestfsCmdError, 458
- LibguestfsTest (class in virttest.utils_libguestfs_unittest), 462
- libvirt_pki_dir (virttest.utils_conn.TLSConnection attribute), 419
- libvirt_pki_private_dir (virttest.utils_conn.TLSConnection attribute), 419
- LibvirtConfigCommon (class in virttest.utils_config), 412
- LibvirtConfigCommonTest (class in virttest.utils_config_unittest), 414
- LibvirtConfigCommonTest.NoTypesConfig (class in virttest.utils_config_unittest), 414
- LibvirtConfigCommonTest.UndefinedTypeConfig (class in virttest.utils_config_unittest), 414
- LibvirtConfigCommonTest.UnimplementedConfig (class in virttest.utils_config_unittest), 414
- LibvirtConfigTest (class in virttest.utils_config_unittest), 414
- LibvirtConfigUnknownKeyError, 413
- LibvirtConfigUnknownKeyTypeError, 413
- Libvirtd (class in virttest.utils_libvirtd), 462
- libvirtd_is_running() (in module virttest.utils_libvirtd), 463
- libvirtd_restart() (in module virttest.utils_libvirtd), 463
- libvirtd_start() (in module virttest.utils_libvirtd), 463
- libvirtd_stop() (in module virttest.utils_libvirtd), 463
- LibvirtdConfig (class in virttest.utils_config), 413
- LibvirtdSession (class in virttest.utils_libvirtd), 462
- LibvirtdSysConfig (class in virttest.utils_config), 413
- LibvirtGuestsConfig (class in virttest.utils_config), 413
- LibvirtIface (class in virttest.utils_net), 482
- LibvirtNetwork (class in virttest.utils_test.libvirt), 261
- LibvirtPolkitConfig (class in virttest.test_setup), 408
- LibvirtQemuConfig (class in virttest.utils_config), 413
- libvirtxml (virttest.libvirt_xml.accessors.AccessorBase attribute), 169
- libvirtxml (virttest.libvirt_xml.accessors.XMLAttribute.Deliner attribute), 170
- libvirtxml (virttest.libvirt_xml.accessors.XMLAttribute.Getter attribute), 170
- libvirtxml (virttest.libvirt_xml.accessors.XMLAttribute.Setter attribute), 170
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementBool.Deliner attribute), 171
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementBool.Getter attribute), 171
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementBool.Setter attribute), 171
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementDict.Deliner attribute), 171
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementDict.Getter attribute), 172
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementDict.Setter attribute), 172
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementInt.Deliner attribute), 172
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), 172
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), 173
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementList.Deliner attribute), 173
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 173
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementNest.Deliner attribute), 174
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementNest.Getter attribute), 174
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementNest.Setter attribute), 174
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementText.Deliner attribute), 175
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementText.Getter attribute), 175
- libvirtxml (virttest.libvirt_xml.accessors.XMLElementText.Setter attribute), 175
- LibvirtXMLAccessorError, 212
- LibvirtXMLBase (class in virttest.libvirt_xml.base), 175
- LibvirtXMLError, 212
- LibvirtXMLForbiddenError, 212
- LibvirtXMLNotFoundError, 212
- LibvirtXMLTestBase (class in virttest.libvirt_xml_unittest), 332
- LIdentifier (class in virttest.cartesian_config), 301
- LInclude (class in virttest.cartesian_config), 302
- LIndent (class in virttest.cartesian_config), 302
- line (virttest.cartesian_config.NoOnlyFilter attribute), 305
- linesep (virttest.utils_sasl.SASL attribute), 497
- link_state (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- LinuxKernelBuildHelper (class in virttest.build_helper), 296
- LinuxVMCheck (class in virttest.utils_v2v), 502

- `list` (`virttest.lvsbs.SandboxService` attribute), 341
- `list()` (`virttest.ovirt.ClusterManager` method), 346
- `list()` (`virttest.ovirt.DataCenterManager` method), 347
- `list()` (`virttest.ovirt.HostManager` method), 347
- `list()` (`virttest.ovirt.StorageDomainManager` method), 347
- `list()` (`virttest.ovirt.VMMManager` method), 349
- `list_br()` (`virttest.openvswitch.OpenVSwitchControl` method), 344
- `list_br()` (`virttest.openvswitch.OpenVSwitchControlCli_140` method), 345
- `list_br()` (`virttest.utils_net.Bridge` method), 480
- `list_devices()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_df()` (`virttest.utils_test.libguestfs.VirtTools` method), 260
- `list_disk_labels()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_events()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_filesystems()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_flags()` (`virttest.lvsb_base.SandboxCommandBase` method), 340
- `list_iface()` (`virttest.utils_net.Bridge` method), 480
- `list_long_options()` (`virttest.lvsb_base.SandboxCommandBase` method), 340
- `list_md_devices()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_missing_named_buses()` (`virttest.qemu_devices.qcontainer.DevContainer` method), 218
- `list_partitions()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `list_pools()` (`virttest.libvirt_storage.StoragePool` method), 323
- `list_ports()` (`virttest.openvswitch.OpenVSwitchControlCli_140` method), 345
- `list_pos()` (`virttest.lvsb_base.SandboxCommandBase` method), 340
- `list_short_options()` (`virttest.lvsb_base.SandboxCommandBase` method), 340
- `list_users()` (`virttest.utils_sasl.SASL` method), 497
- `list_volumes()` (`virttest.libvirt_storage.PoolVolume` method), 321
- `listen` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129
- `listen_addr` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129
- `listen_addr` (`virttest.utils_conn.TCPConnection` attribute), 418
- `listen_addr` (`virttest.utils_conn.TLSConnection` attribute), 419
- `listen_cmds()` (`virttest.remote_commander.remote_master.CommanderMas` method), 228
- `listen_errors()` (`virttest.remote_commander.remote_master.CommanderMas` method), 228
- `listen_messenger()` (`virttest.remote_commander.remote_master.Commander` method), 228
- `listen_streams()` (`virttest.remote_commander.remote_master.CommanderM` method), 228
- `listen_type` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129
- `listening_on()` (`virttest.utils_net.IPAddress` method), 481
- `listens` (`virttest.libvirt_xml.devices.graphics.Graphics` attribute), 129
- `live_snapshot()` (`virttest.qemu_monitor.HumanMonitor` method), 363
- `live_snapshot()` (`virttest.qemu_monitor.QMPMonitor` method), 370
- `live_snapshot()` (`virttest.qemu_vm.VM` method), 387
- `ll()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 441
- `LLBracket` (class in `virttest.cartesian_config`), 302
- `LLRBracket` (class in `virttest.cartesian_config`), 302
- `LNo` (class in `virttest.cartesian_config`), 302
- `LNotCond` (class in `virttest.cartesian_config`), 302
- `load()` (in module `virttest.staging.backports.simplejson`), 239
- `load_ksm_module()` (`virttest.utils_misc.KSMController` method), 464
- `load_modules()` (`virttest.base_installer.BaseInstaller` method), 291
- `load_modules()` (`virttest.base_installer.FailedInstaller` method), 292
- `load_setup_modules()` (in module `virttest.common`), 308
- `load_stress()` (in module `virttest.utils_test`), 277
- `load_stress_tool()` (`virttest.utils_test.HostStress` method), 275
- `load_stress_tool()` (`virttest.utils_test.VMStress` method), 276
- `load_xml_module()` (in module `virttest.libvirt_xml.base`), 177
- `loader` (`virttest.libvirt_xml.vm_xml.VMOSXML` attribute), 202
- `loads()` (in module `virttest.staging.backports.simplejson`), 239
- `loadvm()` (`virttest.qemu_vm.VM` method), 387
- `loadvm()` (`virttest.virt_vm.BaseVM` method), 557
- `local_runner()` (in module `virttest.utils_net`), 490
- `local_runner_status()` (in module `virttest.utils_net`), 490
- `LocalSourceDirHelper` (class in `virttest.build_helper`), 296
- `LocalSourceDirInstaller` (class in `virttest.base_installer`), 292
- `LocalSourceDirInstaller` (class in `virttest.qemu_installer`), 360
- `LocalSourceDirParamHelper` (class in `virttest.build_helper`), 296

- LocalSourceTarInstaller (class in virttest.base_installer), 292
- LocalSourceTarInstaller (class in virttest.qemu_installer), 360
- LocalTarHelper (class in virttest.build_helper), 297
- LocalTarParamHelper (class in virttest.build_helper), 297
- lock (virttest.utils_params.Params attribute), 496
- lock_db() (virttest.utils_net.DbNet method), 480
- lock_file() (in module virttest.utils_misc), 475
- lock_safe() (in module virttest.utils_env), 424
- locked (virttest.libvirt_xml.vm_xml.VMMemBackingXML attribute), 201
- log() (virttest.syslog_server.RequestHandler method), 406
- LOG_ALERT (virttest.syslog_server.RequestHandler attribute), 405
- LOG_AUTH (virttest.syslog_server.RequestHandler attribute), 405
- LOG_AUTHPRIV (virttest.syslog_server.RequestHandler attribute), 405
- LOG_CRIT (virttest.syslog_server.RequestHandler attribute), 405
- LOG_CRON (virttest.syslog_server.RequestHandler attribute), 405
- LOG_DAEMON (virttest.syslog_server.RequestHandler attribute), 406
- LOG_DEBUG (virttest.syslog_server.RequestHandler attribute), 406
- LOG_EMERG (virttest.syslog_server.RequestHandler attribute), 406
- LOG_ERR (virttest.syslog_server.RequestHandler attribute), 406
- LOG_FTP (virttest.syslog_server.RequestHandler attribute), 406
- LOG_INFO (virttest.syslog_server.RequestHandler attribute), 406
- LOG_KERN (virttest.syslog_server.RequestHandler attribute), 406
- log_last_traceback() (in module virttest.utils_misc), 475
- log_line() (in module virttest.utils_misc), 475
- LOG_LOCAL0 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL1 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL2 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL3 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL4 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL5 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL6 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LOCAL7 (virttest.syslog_server.RequestHandler attribute), 406
- LOG_LPR (virttest.syslog_server.RequestHandler attribute), 406
- LOG_MAIL (virttest.syslog_server.RequestHandler attribute), 406
- log_message() (virttest.http_server.HTTPRequestHandler method), 317
- LOG_NEWS (virttest.syslog_server.RequestHandler attribute), 406
- LOG_NOTICE (virttest.syslog_server.RequestHandler attribute), 406
- LOG_SYSLOG (virttest.syslog_server.RequestHandler attribute), 406
- LOG_USER (virttest.syslog_server.RequestHandler attribute), 406
- LOG_UUCP (virttest.syslog_server.RequestHandler attribute), 406
- LOG_WARNING (virttest.syslog_server.RequestHandler attribute), 406
- LogicalVolume (class in virttest.lvm), 336
- login() (virttest.virt_vm.BaseVM method), 557
- LOGIN_TIMEOUT (virttest.virt_vm.BaseVM attribute), 553
- LOGIN_WAIT_TIMEOUT (virttest.virt_vm.BaseVM attribute), 553
- LoginAuthenticationError, 391
- LoginBadClientError, 391
- LoginError, 391
- LoginProcessTerminatedError, 391
- LoginTimeoutError, 391
- LogLockError, 465
- long_name (virttest.cartesian_config.Label attribute), 304
- LOnly (class in virttest.cartesian_config), 302
- lookup_by_storagedomains() (virttest.ovirt.VMManager method), 349
- lookup_vm_class() (virttest.virt_vm.BaseVM static method), 557
- loop_assert() (virttest.utils_net_unittest.TestVirtIface method), 493
- LOperators (class in virttest.cartesian_config), 302
- LOR (class in virttest.cartesian_config), 303
- LPrepend (class in virttest.cartesian_config), 303
- LRBracket (class in virttest.cartesian_config), 303
- LRegExpAppend (class in virttest.cartesian_config), 303
- LRegExpPrepend (class in virttest.cartesian_config), 303
- LRegExpSet (class in virttest.cartesian_config), 303
- LRegExpStart (class in virttest.cartesian_config), 303
- LRegExpStop (class in virttest.cartesian_config), 303
- LRRBracket (class in virttest.cartesian_config), 303
- ls() (virttest.utils_libguestfs.GuestfishPersistent method), 441
- LSet (class in virttest.cartesian_config), 303
- lstat() (virttest.utils_libguestfs.GuestfishPersistent

- method), 441
 - lstatlist() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - LString (class in virttest.cartesian_config), 303
 - LUpdateFileMap (class in virttest.cartesian_config), 304
 - lv_check() (in module virttest.staging.lv_utils), 244
 - LVariant (class in virttest.cartesian_config), 304
 - LVariants (class in virttest.cartesian_config), 304
 - lvcreate() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - LVM (class in virttest.lvm), 335
 - lvm_canonical_lv_name() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvm_clear_filter() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvm_remove_all() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvm_set_filter() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - LVMdev (class in virttest.qemu_storage), 377
 - LVMdev (class in virttest.storage), 403
 - lvremove() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvrename() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvresize() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvresize_free() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvs() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvs_full() (virttest.utils_libguestfs.GuestfishPersistent method), 442
 - lvuuid() (virttest.utils_libguestfs.GuestfishPersistent method), 443
 - LWhite (class in virttest.cartesian_config), 304
- ## M
- Mac (class in virttest.libvirt_xml.nwfilter_protocols.mac), 154
 - mac (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
 - mac (virttest.utils_net.VirtIface attribute), 485
 - mac (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
 - Mac.Attr (class in virttest.libvirt_xml.nwfilter_protocols.mac), 154
 - mac_address (virttest.libvirt_xml.devices.interface.Interface attribute), 133
 - mac_index() (virttest.utils_net.DbNet method), 480
 - mac_index() (virttest.utils_net.ParamsNet method), 483
 - mac_index() (virttest.utils_net.VirtNet method), 486
 - mac_is_valid() (virttest.utils_net.VirtIface class method), 485
 - mac_is_valid() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 493
 - mac_list() (virttest.utils_net.VMNet method), 484
 - mac_prefix (virttest.utils_net_unittest.TestVmNetSubclasses attribute), 494
 - mac_str_to_int_list() (virttest.utils_net.VirtIface class method), 485
 - mac_str_to_int_list() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 493
 - Machine (class in virttest.ovs_utils), 350
 - machine (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
 - machine_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 218
 - Macvtap (class in virttest.utils_net), 482
 - MacvtapCreationError, 483
 - MacvtapGetBaseInterfaceError, 483
 - main() (in module virttest.rss_client), 400
 - make() (virttest.build_helper.GnuSourceBuildHelper method), 295
 - make() (virttest.build_helper.LinuxKernelBuildHelper method), 296
 - make() (virttest.remote_build.Builder method), 397
 - make_callable() (virttest.libvirt_xml.accessors.AccessorGeneratorBase method), 169
 - make_clean() (virttest.build_helper.GnuSourceBuildHelper method), 295
 - make_clean() (virttest.build_helper.LinuxKernelBuildHelper method), 296
 - make_create_command() (virttest.libvirt_vm.VM method), 328
 - make_create_command() (virttest.qemu_vm.VM method), 387
 - make_emulate_image() (virttest.lvm.EmulatedLVM method), 334
 - make_forbidden() (virttest.libvirt_xml.accessors.AccessorGeneratorBase method), 169
 - make_guest_kernel() (virttest.build_helper.LinuxKernelBuildHelper method), 296
 - make_install() (virttest.build_helper.GnuSourceBuildHelper method), 295
 - make_installer() (in module virttest.installer), 317
 - make_non_parallel() (virttest.build_helper.GnuSourceBuildHelper method), 295
 - make_parallel() (virttest.build_helper.GnuSourceBuildHelper method), 295
 - make_sandbox_command_line() (virttest.lvsb_base.SandboxBase method), 339
 - make_sandbox_command_line() (virttest.lvsb_base.SandboxCommandBase method), 340

make_sandboxes() (in module virttest.lvsb), 339
 make_scanner() (in module virttest.staging.backports.simplejson.scanner), 236
 make_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.PCIXML static method), 187
 make_sysfs_sub_path() (virttest.libvirt_xml.nodedev_xml.SystemXML static method), 188
 make_volume() (virttest.lvm.EmulatedLVM method), 335
 man() (virttest.utils_libguestfs.GuestfishPersistent method), 443
 managed (virttest.libvirt_xml.devices.hostdev.Hostdev attribute), 130
 managedsave() (in module virttest.virsh), 528
 managedsave() (virttest.libvirt_vm.VM method), 329
 managedsave_remove() (in module virttest.virsh), 528
 Manager (class in virttest.versionable_class), 505
 marshal_from (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 174
 marshal_from_address() (virttest.libvirt_xml.nodedev_xml.PCIXML static method), 187
 marshal_from_boots() (virttest.libvirt_xml.vm_xml.VMOSXML static method), 202
 marshal_from_cell() (virttest.libvirt_xml.vm_xml.VMCPUXML static method), 198
 marshal_from_cpu() (virttest.libvirt_xml.capability_xml.CellXML static method), 179
 marshal_from_filterref() (virttest.libvirt_xml.nwfilter_xml.NwfilterXML static method), 190
 marshal_from_forward_iface() (virttest.libvirt_xml.network_xml.NetworkXMLBase static method), 184
 marshal_from_forwarder() (virttest.libvirt_xml.network_xml.DNSXML static method), 180
 marshal_from_host() (virttest.libvirt_xml.devices.disk.Disk.DiskSource static method), 126
 marshal_from_host() (virttest.libvirt_xml.network_xml.IPXML static method), 181
 marshal_from_host() (virttest.libvirt_xml.pool_xml.SourceXML static method), 193
 marshal_from_hostname() (virttest.libvirt_xml.network_xml.DNSXML.HostXML static method), 180
 marshal_from_memnode() (virttest.libvirt_xml.vm_xml.VMXMLBase static method), 209
 marshal_from_page() (virttest.libvirt_xml.vm_xml.VMHugepagesXML static method), 201
 marshal_from_pages() (virttest.libvirt_xml.capability_xml.CellXML static method), 179
 marshal_from_parameter() (virttest.libvirt_xml.devices.interface.Interface.Filterref static method), 133
 static method), 132
 marshal_from_route() (virttest.libvirt_xml.network_xml.NetworkXMLBase static method), 184
 marshal_from_seclabel() (virttest.libvirt_xml.devices.disk.Disk.DiskSource static method), 126
 marshal_from_sibling() (virttest.libvirt_xml.capability_xml.CellXML static method), 179
 marshal_from_source() (virttest.libvirt_xml.devices.rng.Rng.Backend static method), 136
 marshal_from_sources() (virttest.libvirt_xml.devices.console.Console static method), 124
 marshal_from_sources() (virttest.libvirt_xml.devices.serial.Serial static method), 137
 marshal_from_timer() (virttest.libvirt_xml.vm_xml.VMClockXML static method), 199
 marshal_from_vcpupins() (virttest.libvirt_xml.vm_xml.VMCPUTuneXML static method), 197
 marshal_to (virttest.libvirt_xml.accessors.XMLElementList.Delter attribute), 173
 marshal_to (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173
 marshal_to_address() (virttest.libvirt_xml.nodedev_xml.PCIXML static method), 187
 marshal_to_boots() (virttest.libvirt_xml.vm_xml.VMOSXML static method), 202
 marshal_to_cell() (virttest.libvirt_xml.vm_xml.VMCPUXML static method), 198
 marshal_to_cpu() (virttest.libvirt_xml.capability_xml.CellXML static method), 179
 marshal_to_filterref() (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase static method), 190
 marshal_to_forward_iface() (virttest.libvirt_xml.network_xml.NetworkXMLBase static method), 184
 marshal_to_forwarder() (virttest.libvirt_xml.network_xml.DNSXML static method), 180
 marshal_to_host() (virttest.libvirt_xml.devices.disk.Disk.DiskSource static method), 126
 marshal_to_host() (virttest.libvirt_xml.network_xml.IPXML static method), 181
 marshal_to_host() (virttest.libvirt_xml.pool_xml.SourceXML static method), 194
 marshal_to_hostname() (virttest.libvirt_xml.network_xml.DNSXML.HostXML static method), 180
 marshal_to_memnode() (virttest.libvirt_xml.vm_xml.VMXMLBase static method), 209
 marshal_to_page() (virttest.libvirt_xml.vm_xml.VMHugepagesXML static method), 201
 marshal_to_pages() (virttest.libvirt_xml.capability_xml.CellXML static method), 179
 marshal_to_parameter() (virttest.libvirt_xml.devices.interface.Interface.Filterref static method), 133

[marshal_to_route\(\)](#) (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 443
[marshal_to_seclabel\(\)](#) (virttest.libvirt_xml.devices.disk.Disk.DiskSource attribute), 202
[marshal_to_sibling\(\)](#) (virttest.libvirt_xml.capability_xml.CellXML attribute), 197
[marshal_to_source\(\)](#) (virttest.libvirt_xml.devices.rng.Rng.Backend attribute), 135
[marshal_to_sources\(\)](#) (virttest.libvirt_xml.devices.console.Console attribute), 197
[marshal_to_sources\(\)](#) (virttest.libvirt_xml.devices.serial.Serial attribute), 179
[marshal_to_timer\(\)](#) (virttest.libvirt_xml.vm_xml.VMClockXML attribute), 199
[marshal_to_vcpupins\(\)](#) (virttest.libvirt_xml.vm_xml.VMCPMXMLClass in virttest.libvirt_xml.devices.memory), 134
[master_id\(\)](#) (virttest.utils_test.qemu.MultihostMigration method), 271
[match](#) (virttest.libvirt_xml.vm_xml.VMCPXML attribute), 198
[match\(\)](#) (virttest.cartesian_config.Filter method), 300
[match\(\)](#) (virttest.cartesian_config.Lexer method), 305
[match_bus\(\)](#) (virttest.qemu_devices.qbuses.QSparseBus method), 214
[match_patterns\(\)](#) (virttest.aexpect.Expect method), 280
[match_patterns_multiline\(\)](#) (virttest.aexpect.Expect method), 280
[max_age](#) (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attribute), 159
[max_age_hi](#) (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attribute), 159
[max_disks\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 443
[max_mem](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[max_mem_rt](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[max_mem_rt_slots](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[max_mem_rt_unit](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[max_mem_unit](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[maxvcpus\(\)](#) (in module virttest.virsh), 528
[mb](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[md5eval\(\)](#) (in module virttest.ppm_utils), 356
[md_create\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 443
[md_detail\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 443
[md_stat\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 443
[md_stop\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 443
[mem_enabled](#) (virttest.libvirt_xml.vm_xml.VMPXML attribute), 202
[mem_file](#) (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase attribute), 197
[mem_model](#) (virttest.libvirt_xml.devices.memory.Memory attribute), 135
[mem_snap_type](#) (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase attribute), 197
[mem_unit](#) (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
[Memballoon](#) (class in virttest.libvirt_xml.devices.memballoon), 134
[Memory](#) (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
[Memory.Address](#) (class in virttest.libvirt_xml.devices.memory), 134
[Memory.Source](#) (class in virttest.libvirt_xml.devices.memory), 134
[Memory.Target](#) (class in virttest.libvirt_xml.devices.memory), 134
[memtotal\(\)](#) (in module virttest.staging.utils_memory), 256
[memtune](#) (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 209
[memtune_get\(\)](#) (in module virttest.virsh), 528
[memtune_list\(\)](#) (in module virttest.virsh), 528
[memtune_set\(\)](#) (in module virttest.virsh), 528
[Messenger](#) (class in virttest.remote_commander.messenger), 225
[MessengerError](#), 225, 227
[metadata\(\)](#) (in module virttest.virsh), 528
[mig_cancelled\(\)](#) (virttest.qemu_vm.VM method), 387
[mig_failed\(\)](#) (virttest.qemu_vm.VM method), 387
[mig_finished\(\)](#) (virttest.qemu_vm.VM method), 387
[mig_succeeded\(\)](#) (virttest.qemu_vm.VM method), 387
[might_match\(\)](#) (virttest.cartesian_config.Filter method), 300
[might_pass\(\)](#) (virttest.cartesian_config.NoFilter method), 305
[might_pass\(\)](#) (virttest.cartesian_config.OnlyFilter method), 306
[migrate\(\)](#) (in module virttest.utils_test.qemu), 273
[migrate\(\)](#) (in module virttest.virsh), 529
[migrate\(\)](#) (virttest.libvirt_vm.VM method), 329
[migrate\(\)](#) (virttest.qemu_monitor.HumanMonitor method), 363
[migrate\(\)](#) (virttest.qemu_monitor.QMPMonitor method), 371
[migrate\(\)](#) (virttest.qemu_vm.VM method), 387
[migrate\(\)](#) (virttest.utils_test.qemu.MultihostMigration method), 273

- method), 271
- migrate() (virttest.virt_vm.BaseVM method), 557
- migrate_compcache() (in module virttest.virsh), 529
- migrate_getspeed() (in module virttest.virsh), 529
- migrate_set_downtime() (virttest.qemu_monitor.HumanMonitor method), 363
- migrate_set_downtime() (virttest.qemu_monitor.QMPMonitor method), 371
- migrate_set_speed() (virttest.qemu_monitor.HumanMonitor method), 364
- migrate_set_speed() (virttest.qemu_monitor.QMPMonitor method), 371
- migrate_setmaxdowntime() (in module virttest.virsh), 529
- migrate_setspeed() (in module virttest.virsh), 529
- MIGRATE_TIMEOUT (virttest.virt_vm.BaseVM attribute), 554
- migrate_vms() (virttest.utils_test.qemu.MultihostMigration method), 272
- migrate_vms_dest() (virttest.utils_test.qemu.MultihostMigration method), 272
- migrate_vms_src() (virttest.utils_test.qemu.MultihostMigration method), 272
- migrate_vms_src() (virttest.utils_test.qemu.MultihostMigrationExec method), 272
- migrate_vms_src() (virttest.utils_test.qemu.MultihostMigrationFd method), 273
- migrate_vms_src() (virttest.utils_test.qemu.MultihostMigrationRdma method), 273
- migrate_wait() (virttest.utils_test.qemu.MultihostMigration method), 272
- migrate_wait() (virttest.utils_test.qemu.MultihostMigrationExec method), 273
- migrate_wait() (virttest.utils_test.qemu.MultihostMigrationFd method), 273
- MIGRATION_PROTOS (virttest.qemu_vm.VM attribute), 381
- MIGRATION_PROTOS (virttest.virt_vm.BaseVM attribute), 554
- migration_scenario() (virttest.utils_test.qemu.MultihostMigration method), 272
- MigrationData (class in virttest.utils_test.qemu), 270
- MigrationTest (class in virttest.utils_test.libvirt), 261
- min_guarantee (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- min_guarantee_unit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- mirror (virttest.libvirt_xml.devices.disk.Disk attribute), 127
- mirror (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 195
- MissingDepsDirError, 308
- MissingIncludeError, 305
- mk_cgroup() (virttest.staging.utils_cgroup.Cgroup method), 249
- mk_cgroup_cgcreate() (virttest.staging.utils_cgroup.Cgroup method), 249
- mk_label() (in module virttest.utils_test.libvirt), 266
- mk_lwp() (in module virttest.utils_test.libvirt), 266
- mkdir() (virttest.utils_libguestfs.GuestfishPersistent method), 443
- mkdir_mode() (virttest.utils_libguestfs.GuestfishPersistent method), 443
- mkdir_p() (virttest.utils_libguestfs.GuestfishPersistent method), 443
- mkfifo() (virttest.utils_libguestfs.GuestfishPersistent method), 443
- mkfs() (in module virttest.utils_test.libvirt), 266
- mkfs() (virttest.utils_libguestfs.GuestfishPersistent method), 443
- mkfs_opts() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mklost_and_found() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mkmountpoint() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mknod() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mknod_b() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mknod_c() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mkswap() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mkswap_file() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mkswap_L() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- mkswap_U() (virttest.utils_libguestfs.GuestfishPersistent method), 444
- MockHMPMonitor (class in virttest.qemu_devices_unittest), 359
- MockMonitor (class in virttest.qemu_monitor_unittest), 373
- mode (virttest.libvirt_xml.devices.hostdev.Hostdev attribute), 130
- mode (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
- mode (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 198
- mode (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 198
- mode (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212
- model (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 178
- model (virttest.libvirt_xml.devices.controller.Controller attribute), 124

model (virttest.libvirt_xml.devices.interface.Interface attribute), 133

model (virttest.libvirt_xml.devices.memballoon.Memballoon attribute), 134

model (virttest.libvirt_xml.devices.seclabel.Seclabel attribute), 137

model (virttest.libvirt_xml.vm_xml.VMCPXML attribute), 198

model_heads (virttest.libvirt_xml.devices.video.Video attribute), 138

model_ram (virttest.libvirt_xml.devices.video.Video attribute), 138

model_type (virttest.libvirt_xml.devices.sound.Sound attribute), 138

model_type (virttest.libvirt_xml.devices.video.Video attribute), 138

model_type (virttest.libvirt_xml.devices.watchdog.Watchdog attribute), 138

model_vram (virttest.libvirt_xml.devices.video.Video attribute), 138

modprobe() (virttest.utils_libguestfs.GuestfishPersistent method), 444

ModuleLoad (class in virttest.virsh_unittest), 552

ModuleLoadCheckVirsh (class in virttest.virsh_unittest), 552

ModuleWrapper (class in virttest.versionable_class), 506

Monitor (class in virttest.qemu_monitor), 365

monitor (virttest.qemu_vm.VM attribute), 388

MonitorConnectError, 366

MonitorError, 366

MonitorLockError, 366

MonitorNotSupportedCmdError, 366

MonitorNotSupportedError, 366

MonitorProtocolError, 367

MonitorSocketError, 367

monotonic_time() (in module virttest.utils_misc), 475

more() (virttest.utils_libguestfs.GuestfishPersistent method), 444

mount() (in module virttest.utils_disk), 422

mount() (in module virttest.utils_misc), 475

mount() (virttest.nfs.Nfs method), 343

mount() (virttest.utils_libguestfs.GuestfishPersistent method), 444

mount_all() (virttest.utils_disk.GuestFSModiDisk method), 421

mount_hugepage_fs() (virttest.test_setup.HugePageConfig method), 408

mount_loop() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mount_options() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mount_ro() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mount_vfs() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mountpoints() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mounts() (virttest.utils_disk.GuestFSModiDisk method), 421

mounts() (virttest.utils_libguestfs.GuestfishPersistent method), 445

mouse_button() (virttest.qemu_monitor.HumanMonitor method), 364

mouse_move() (virttest.qemu_monitor.HumanMonitor method), 364

move_mouse() (in module virttest.virsh), 530

move_mouse() (virttest.utils_v2v.WindowsVMCheck method), 504

msg (virttest.remote_commander.remote_interface.StdStream attribute), 227

msg_age (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159

msg_age_hi (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159

MultihostMigration (class in virttest.utils_test.qemu), 270

MultihostMigrationExec (class in virttest.utils_test.qemu), 272

MultihostMigrationFd (class in virttest.utils_test.qemu), 273

MultihostMigrationRdma (class in virttest.utils_test.qemu), 273

N

name (virttest.cartesian_config.Label attribute), 304

name (virttest.cartesian_config.LOperators attribute), 302

name (virttest.cartesian_config.Node attribute), 306

name (virttest.libvirt_xml.devices.interface.Interface.Filterref attribute), 133

name (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184

name (virttest.libvirt_xml.network_xml.PortgroupXML attribute), 185

name (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186

name (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193

name (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199

name (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212

name (virttest.lvsb_base.SandboxCommandBase attribute), 340

name_is_valid() (virttest.utils_net.VirtIface class method), 485

name_is_valid() (virttest.utils_net_unittest.TestVirtIface.VirtIface class method), 493

namedtuple() (in module virttest.staging.backports.collections.namedtuple),

- 232
- nat_port (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- nc_copy_between_remotes() (in module virttest.remote), 393
- need_reexport() (virttest.nfs.Exportfs method), 342
- needs_restart() (virttest.virt_vm.BaseVM method), 557
- NegativeCondition (class in virttest.cartesian_config), 305
- neigh_reachable() (in module virttest.utils_net), 490
- net_autostart() (in module virttest.virsh), 530
- net_create() (in module virttest.virsh), 530
- net_define() (in module virttest.virsh), 530
- net_destroy() (in module virttest.virsh), 530
- net_dhcp_leases() (in module virttest.virsh), 530
- net_dumpxml() (in module virttest.virsh), 531
- net_event() (in module virttest.virsh), 531
- net_info() (in module virttest.virsh), 531
- net_list() (in module virttest.virsh), 531
- net_name() (in module virttest.virsh), 531
- net_start() (in module virttest.virsh), 532
- net_state_dict() (in module virttest.virsh), 532
- net_undefine() (in module virttest.virsh), 532
- net_update() (in module virttest.virsh), 532
- net_uuid() (in module virttest.virsh), 532
- netdev_extra_params (virttest.utils_net.QemuIface attribute), 483
- netdev_id (virttest.utils_net.QemuIface attribute), 483
- netdst (virttest.utils_net.VirtIface attribute), 485
- netdst (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
- NetError, 483
- netmask (virttest.libvirt_xml.network_xml.IPXML attribute), 181
- Netperf (class in virttest.utils_netperf), 494
- NetperfClient (class in virttest.utils_netperf), 494
- NetperfError, 495
- NetperfPackage (class in virttest.utils_netperf), 495
- NetperfPackageError, 495
- NetperfServer (class in virttest.utils_netperf), 495
- NetperfTestError, 496
- NetserverError, 496
- nettests_cartesian (virttest.utils_net_unittest.TestVmNetSubclass attribute), 494
- nettype (virttest.utils_net.VirtIface attribute), 485
- nettype (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
- NetworkTestBase (class in virttest.libvirt_network_unittest), 320
- NetworkXML (class in virttest.libvirt_xml.network_xml), 181
- NetworkXMLBase (class in virttest.libvirt_xml.network_xml), 182
- NetworkXMLTest (class in virttest.libvirt_network_unittest), 320
- NetXML (class in virttest.libvirt_xml.nodedev_xml), 185
- new_all_networks_dict() (virttest.libvirt_xml.network_xml.NetworkXML static method), 181
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.ah.Ah method), 140
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6 method), 141
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.all.All method), 142
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6 method), 143
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.arp.Arp method), 144
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.esp.Esp method), 146
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6 method), 148
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp method), 149
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6 method), 150
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp method), 151
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.ip.Ip method), 152
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6 method), 154
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.mac.Mac method), 155
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp method), 156
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp method), 157
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6 method), 158
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.stp.Stp method), 160
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp method), 161
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6 method), 163
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.udp.Udp method), 164
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6 method), 165
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite method), 166
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6 method), 168
- new_attr() (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan method), 168

new_auth() (virttest.libvirt_xml.devices.disk.Disk class method), 124
 method), 127
 new_bandwidth() (virttest.libvirt_xml.devices.interface.Interface class method), 126
 method), 133
 new_controller_address() (virttest.libvirt_xml.devices.controller.Controller class method), 131
 method), 124
 new_disk_address() (virttest.libvirt_xml.devices.disk.Disk class method), 127
 method), 127
 new_disk_source() (virttest.libvirt_xml.devices.disk.Disk class method), 127
 method), 127
 new_disk_vol_name() (in module virttest.utils_test.libvirt), 266
 new_dns() (virttest.libvirt_xml.network_xml.NetworkXMLBase class method), 184
 method), 184
 new_driver() (virttest.libvirt_xml.devices.interface.Interface class method), 133
 method), 133
 new_encryption() (virttest.libvirt_xml.devices.disk.Disk class method), 127
 method), 127
 new_encryption() (virttest.libvirt_xml.vol_xml.VolXML class method), 211
 method), 211
 new_filterref() (virttest.libvirt_xml.devices.interface.Interface class method), 133
 method), 133
 new_from_dict() (virttest.libvirt_xml.devices.address.Address class method), 122
 class method), 122
 new_from_dict() (virttest.libvirt_xml.devices.base.UntypedDeviceBase class method), 122
 class method), 122
 new_from_dict() (virttest.libvirt_xml.devices.controller.Controller class method), 124
 class method), 124
 new_from_dict() (virttest.libvirt_xml.devices.disk.Disk.Address class method), 125
 class method), 125
 new_from_dict() (virttest.libvirt_xml.devices.hub.Hub.Address class method), 131
 class method), 131
 new_from_dict() (virttest.libvirt_xml.devices.input.Input.Address class method), 131
 class method), 131
 new_from_dict() (virttest.libvirt_xml.devices.interface.Interface.Address class method), 132
 class method), 132
 new_from_dict() (virttest.libvirt_xml.devices.memory.Memory.Address class method), 134
 class method), 134
 new_from_dict() (virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase class method), 145
 class method), 145
 new_from_dumpxml() (virttest.libvirt_xml.nodedev_xml.NodedevXML static method), 186
 static method), 186
 new_from_dumpxml() (virttest.libvirt_xml.pool_xml.PoolXML static method), 192
 static method), 192
 new_from_dumpxml() (virttest.libvirt_xml.vm_xml.VMXML static method), 205
 static method), 205
 new_from_element() (virttest.libvirt_xml.devices.address.Address class method), 122
 class method), 122
 new_from_element() (virttest.libvirt_xml.devices.base.TypedDeviceBase class method), 122
 class method), 122
 new_from_element() (virttest.libvirt_xml.devices.base.UntypedDeviceBase class method), 122
 class method), 122
 new_from_element() (virttest.libvirt_xml.devices.controller.Controller class method), 124
 class method), 124
 new_from_element() (virttest.libvirt_xml.devices.disk.Disk class method), 124
 class method), 124
 new_from_element() (virttest.libvirt_xml.devices.disk.Disk.Address class method), 126
 class method), 126
 new_from_element() (virttest.libvirt_xml.devices.hub.Hub.Address class method), 131
 class method), 131
 new_from_element() (virttest.libvirt_xml.devices.input.Input.Address class method), 131
 class method), 131
 new_from_element() (virttest.libvirt_xml.devices.interface.Interface.Address class method), 132
 class method), 132
 new_from_element() (virttest.libvirt_xml.devices.memory.Memory.Address class method), 134
 class method), 134
 new_from_element() (virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase class method), 145
 class method), 145
 new_from_element() (virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase class method), 145
 class method), 145
 new_from_filter_dumpxml() (virttest.libvirt_xml.nwfilter_xml.NwfilterXML static method), 189
 static method), 189
 new_from_inactive_dumpxml() (virttest.libvirt_xml.vm_xml.VMXML static method), 206
 static method), 206
 new_from_net_dumpxml() (virttest.libvirt_xml.network_xml.NetworkXML static method), 181
 static method), 181
 new_from_secret_dumpxml() (virttest.libvirt_xml.secret_xml.SecretXML static method), 194
 static method), 194
 new_from_snapshot_dumpxml() (virttest.libvirt_xml.snapshot_xml.SnapshotXML static method), 196
 static method), 196
 new_from_vol_dumpxml() (virttest.libvirt_xml.vol_xml.VolXML static method), 211
 static method), 211
 new_host() (virttest.libvirt_xml.network_xml.DNSXML static method), 180
 static method), 180
 new_iface_address() (virttest.libvirt_xml.devices.hub.Hub static method), 131
 static method), 131
 new_iface_address() (virttest.libvirt_xml.devices.interface.Interface static method), 133
 static method), 133
 new_iface_address() (virttest.libvirt_xml.devices.input.Input static method), 131
 static method), 131
 new_iface_address() (virttest.libvirt_xml.devices.base.TypedDeviceBase static method), 122
 static method), 122
 new_iface_address() (virttest.libvirt_xml.devices.controller.Controller static method), 124
 static method), 124
 new_iface_address() (virttest.libvirt_xml.devices.disk.Disk static method), 125
 static method), 125
 new_iface_address() (virttest.libvirt_xml.devices.hub.Hub static method), 131
 static method), 131
 new_iface_address() (virttest.libvirt_xml.devices.input.Input static method), 131
 static method), 131
 new_iface_address() (virttest.libvirt_xml.devices.interface.Interface static method), 132
 static method), 132
 new_iface_address() (virttest.libvirt_xml.devices.memory.Memory static method), 134
 static method), 134
 new_iface_address() (virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase static method), 145
 static method), 145
 new_iface_address() (virttest.libvirt_xml.devices.input.Input static method), 131
 static method), 131
 new_iface_address() (virttest.libvirt_xml.devices.disk.Disk static method), 128
 static method), 128
 new_iface_address() (virttest.libvirt_xml.devices.memory.Memory static method), 135
 static method), 135
 new_iface_address() (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules static method), 191
 static method), 191
 new_session() (virttest.lvsb_base.SandboxSession static method), 341
 static method), 341
 new_session() (virttest.utils_libguestfs.GuestfishPersistent static method), 445
 static method), 445
 new_session() (virttest.virsh.VirshConnectBack static method), 509
 static method), 509
 new_session() (virttest.virsh.VirshPersistent static method), 509
 static method), 509

- 510
- `new_source_address()` (virttest.libvirt_xml.devices.hostdev.Hostdev virttest.libvirt_xml.nodedev_xml), 185
- `method`), 130
- `new_untyped_address()` (virttest.libvirt_xml.devices.hostdev.Hostdev virttest.libvirt_xml.nodedev_xml), 186
- `method`), 130
- `new_vol()` (virttest.libvirt_xml.vol_xml.VolXML static `method`), 211
- `NewPoolTest` (class in virttest.libvirt_storage_unittest), 323
- `next()` (in module virttest.staging.backports), 243
- `next()` (virttest.element_tree.iterparse method), 310
- `next_nw()` (in module virttest.cartesian_config), 307
- `Nfs` (class in virttest.nfs), 342
- `nfs_exported()` (in module virttest.nfs), 343
- `nfs_test` (class in virttest.nfs_unittest), 343
- `NFSClient` (class in virttest.nfs), 342
- `nh_stderr` (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
- `nh_stdin` (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
- `nh_stdout` (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
- `nic_extra_params` (virttest.utils_net.QemuIface attribute), 483
- `nic_lookup()` (virttest.utils_net.VMNet method), 484
- `nic_model` (virttest.utils_net.VirtIface attribute), 485
- `nic_model` (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
- `nic_name` (virttest.utils_net.VirtIface attribute), 485
- `nic_name` (virttest.utils_net_unittest.TestVirtIface.VirtIface attribute), 493
- `nic_name_index()` (virttest.utils_net.VMNet method), 484
- `nic_name_list()` (virttest.utils_net.VMNet method), 484
- `nmi()` (virttest.qemu_monitor.HumanMonitor method), 364
- `nmi()` (virttest.qemu_monitor.QMPMonitor method), 371
- `no_filter()` (virttest.cartesian_config.Parser method), 306
- `Node` (class in virttest.cartesian_config), 305
- `node` (virttest.libvirt_xml.devices.memory.Memory.Target attribute), 135
- `node_memtune()` (in module virttest.virsh), 533
- `node_size()` (in module virttest.staging.utils_memory), 256
- `nodecpumap()` (in module virttest.virsh), 533
- `nodecpustats()` (in module virttest.virsh), 533
- `nodedev_create()` (in module virttest.virsh), 533
- `nodedev_destroy()` (in module virttest.virsh), 533
- `nodedev_detach()` (in module virttest.virsh), 533
- `nodedev_dettach()` (in module virttest.virsh), 534
- `nodedev_dumpxml()` (in module virttest.virsh), 534
- `nodedev_list()` (in module virttest.virsh), 534
- `nodedev_reattach()` (in module virttest.virsh), 534
- `nodedev_reset()` (in module virttest.virsh), 534
- `NodedevXML` (class in virttest.libvirt_xml.nodedev_xml), 185
- `NodedevXMLBase` (class in virttest.libvirt_xml.nodedev_xml), 186
- `nodeinfo()` (in module virttest.virsh), 534
- `nodemask` (virttest.libvirt_xml.devices.memory.Memory.Source attribute), 134
- `nodememstats()` (in module virttest.virsh), 534
- `nodeset` (virttest.libvirt_xml.vm_xml.VMHugepagesXML.PageXML attribute), 200
- `nodesuspend()` (in module virttest.virsh), 535
- `NoFilter` (class in virttest.cartesian_config), 305
- `NoModuleError`, 292
- `none_or_int()` (in module virttest.qemu_devices.utils), 224
- `NoOnlyFilter` (class in virttest.cartesian_config), 305
- `NOPIInstaller` (class in virttest.base_installer), 292
- `normalize_connect_uri()` (in module virttest.libvirt_vm), 331
- `normalize_data_size()` (in module virttest.lvm), 338
- `normalize_data_size()` (in module virttest.utils_misc), 476
- `normalize_images()` (virttest.video_maker.GstPythonVideoMaker method), 508
- `nosharepages` (virttest.libvirt_xml.vm_xml.VMMemBackingXML attribute), 201
- `NotExpectedPoolTest` (class in virttest.libvirt_storage_unittest), 323
- `nr_devices()` (virttest.utils_libguestfs.GuestfishPersistent method), 445
- `ntfs_3g_probe()` (virttest.utils_libguestfs.GuestfishPersistent method), 445
- `ntfsresize_opts()` (virttest.utils_libguestfs.GuestfishPersistent method), 445
- `ntpdate()` (in module virttest.utils_test), 277
- `num` (virttest.libvirt_xml.capability_xml.TopologyXML attribute), 179
- `numa_cell` (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 198
- `numa_memnode` (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- `numa_memory` (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- `numa_node` (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
- `numa_nodes()` (in module virttest.staging.utils_memory), 256
- `NumaInfo` (class in virttest.utils_misc), 465
- `NumaNode` (class in virttest.utils_misc), 465
- `numatune()` (in module virttest.virsh), 535
- `nwfilter_define()` (in module virttest.virsh), 535
- `nwfilter_dumpxml()` (in module virttest.virsh), 535
- `nwfilter_edit()` (in module virttest.virsh), 535
- `nwfilter_list()` (in module virttest.virsh), 535

- nwfilter_undefine() (in module virttest.virsh), 536
 NwfilterRulesProtocol (class in virttest.libvirt_xml.nwfilter_xml), 188
 NwfilterXML (class in virttest.libvirt_xml.nwfilter_xml), 188
 NwfilterXMLBase (class in virttest.libvirt_xml.nwfilter_xml), 189
 NwfilterXMLRules (class in virttest.libvirt_xml.nwfilter_xml), 191
- ## O
- object_counts() (virttest.utils_params.Params method), 496
 object_params() (virttest.qemu_devices_unittest.ParamsDict method), 359
 object_params() (virttest.qemu_qtree_unittest.ParamsDict method), 376
 object_params() (virttest.utils_params.Params method), 496
 objects() (virttest.qemu_devices_unittest.ParamsDict method), 359
 objects() (virttest.qemu_qtree_unittest.ParamsDict method), 376
 objects() (virttest.utils_params.Params method), 496
 occupy_space() (virttest.utils_test.RemoteDiskManager method), 275
 offset (virttest.libvirt_xml.vm_xml.VMClockXML attribute), 199
 on_crash (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
 on_poweroff (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
 on_reboot (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
 only_filter() (virttest.cartesian_config.Parser method), 306
 OnlyFilter (class in virttest.cartesian_config), 306
 opcode (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attribute), 144
 opcode (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attribute), 155
 open() (virttest.utils_net.Macvtap method), 483
 open_macvtap() (in module virttest.utils_net), 490
 open_session() (virttest.lvsb_base.SandboxSession method), 341
 open_session() (virttest.utils_libguestfs.GuestfishPersistent method), 445
 open_tap() (in module virttest.utils_net), 490
 openflow_manager() (in module virttest.utils_net), 490
 OpenflowSwitchError, 483
 OpenVSwitch (class in virttest.openvswitch), 344
 OpenVSwitchControl (class in virttest.openvswitch), 344
 OpenVSwitchControlCli_140 (class in virttest.openvswitch), 344
 OpenVSwitchControlCli_CNT (class in virttest.openvswitch), 345
 OpenVSwitchControlDB_140 (class in virttest.openvswitch), 345
 OpenVSwitchControlDB_CNT (class in virttest.openvswitch), 345
 OpenVSwitchSystem (class in virttest.openvswitch), 345
 operation (virttest.libvirt_xml.accessors.AccessorBase attribute), 169
 operation (virttest.libvirt_xml.accessors.XMLAttribute.Delater attribute), 170
 operation (virttest.libvirt_xml.accessors.XMLAttribute.Getter attribute), 170
 operation (virttest.libvirt_xml.accessors.XMLAttribute.Setter attribute), 170
 operation (virttest.libvirt_xml.accessors.XMLElementBool.Delater attribute), 171
 operation (virttest.libvirt_xml.accessors.XMLElementBool.Getter attribute), 171
 operation (virttest.libvirt_xml.accessors.XMLElementBool.Setter attribute), 171
 operation (virttest.libvirt_xml.accessors.XMLElementDict.Delater attribute), 171
 operation (virttest.libvirt_xml.accessors.XMLElementDict.Getter attribute), 172
 operation (virttest.libvirt_xml.accessors.XMLElementDict.Setter attribute), 172
 operation (virttest.libvirt_xml.accessors.XMLElementInt.Delater attribute), 172
 operation (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), 172
 operation (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), 173
 operation (virttest.libvirt_xml.accessors.XMLElementList.Delater attribute), 173
 operation (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173
 operation (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 174
 operation (virttest.libvirt_xml.accessors.XMLElementNest.Delater attribute), 174
 operation (virttest.libvirt_xml.accessors.XMLElementNest.Getter attribute), 174
 operation (virttest.libvirt_xml.accessors.XMLElementNest.Setter attribute), 174
 operation (virttest.libvirt_xml.accessors.XMLElementText.Delater attribute), 175
 operation (virttest.libvirt_xml.accessors.XMLElementText.Getter attribute), 175
 operation (virttest.libvirt_xml.accessors.XMLElementText.Setter attribute), 175
 OptionMissing, 403
 orbital_nuclear_strike() (virttest.libvirt_xml.network_xml.NetworkXML method), 182

- OrderedDict (class in virttest.staging.backports.collections.OrderedDict), 231
- OrderedDict (class in virttest.staging.backports.simplejson), parent_xpath (virttest.libvirt_xml.accessors.XMLElementBool.Getter attribute), 171
- OrderedDict (class in virttest.staging.backports.simplejson.OrderedDict), parent_xpath (virttest.libvirt_xml.accessors.XMLElementBool.Setter attribute), 171
- os (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Delter attribute), 171
- os_inspects() (virttest.utils_disk.GuestFSModiDisk parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Getter attribute), 172 method), 421
- outbound (virttest.libvirt_xml.devices.interface.Interface.Bandwidth parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Setter attribute), 172 attribute), 132
- ovs_br_exists() (in module virttest.utils_net), 490
- ovs_vsctl() (virttest.openvswitch.OpenVSwitchControlCli_140 parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Delter attribute), 172 method), 345
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), 172
- owner (virttest.libvirt_xml.pool_xml.PoolXMLBase at-tribute), 193
- owner (virttest.libvirt_xml.vol_xml.VolXMLBase at-tribute), 212
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Delter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 174
- pages (virttest.libvirt_xml.capability_xml.CellXML at-tribute), 179
- pages (virttest.libvirt_xml.vm_xml.VMHugepagesXML attribute), 201
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Delter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Getter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Setter attribute), 174
- pagesize (virttest.libvirt_xml.devices.memory.Memory.Source parent_xpath (virttest.libvirt_xml.accessors.XMLElementText.Delter attribute), 175 attribute), 134
- pagesize_unit (virttest.libvirt_xml.devices.memory.Memory.Source parent_xpath (virttest.libvirt_xml.accessors.XMLElementText.Getter attribute), 175 attribute), 134
- Panic (class in virttest.libvirt_xml.devices.panic), 135
- Parallel (class in virttest.libvirt_xml.devices.parallel), 135
- parallel() (in module virttest.utils_misc), 476
- parameters (virttest.libvirt_xml.devices.interface.Interface.Filterref attribute), 133
- ParamNotFound, 496
- Params (class in virttest.utils_params), 496
- params (virttest.utils_test.libguestfs.GuestfishTools at-tribute), 259
- ParamsDict (class in virttest.qemu_devices_unittest), 359
- ParamsDict (class in virttest.qemu_qtree_unittest), 376
- ParamsNet (class in virttest.utils_net), 483
- parent (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
- parent_name (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase attribute), 197
- parent_xpath (virttest.libvirt_xml.accessors.XMLAttribute.Delter attribute), 170
- parent_xpath (virttest.libvirt_xml.accessors.XMLAttribute.Getter attribute), 170
- parent_xpath (virttest.libvirt_xml.accessors.XMLAttribute.Setter attribute), 170
- parent_xpath (virttest.libvirt_xml.accessors.XMLAttribute.Source attribute), 170
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementBool.Delter attribute), 171
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementBool.Getter attribute), 171
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementBool.Setter attribute), 171
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Delter attribute), 171
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Getter attribute), 172
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementDict.Setter attribute), 172
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Delter attribute), 172
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), 172
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Delter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Delter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Getter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementNest.Setter attribute), 174
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementText.Delter attribute), 175
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementText.Getter attribute), 175
- parent_xpath (virttest.libvirt_xml.accessors.XMLElementText.Setter attribute), 175
- parse() (in module virttest.element_tree), 310
- parse() (virttest.element_tree.ElementTree method), 310
- parse() (virttest.staging.utils_koji.KojiPkgSpec method), 254
- parse() (virttest.staging.utils_koji.KojiScratchPkgSpec method), 255
- parse() (virttest.xml_utils.TemplateXML method), 562
- parse_arp() (in module virttest.utils_net), 490
- parse_environment() (virttest.utils_libguestfs.GuestfishPersistent method), 445
- parse_environment_list() (virttest.utils_libguestfs.GuestfishPersistent method), 445
- parse_file() (virttest.cartesian_config.Parser method), 306
- parse_file() (virttest.postprocess_iozone.IOzoneAnalyzer method), 352
- parse_filter() (in module virttest.cartesian_config), 307
- parse_func_name() (virttest.remote_commander.remote_runner.CmdSlave

- method), 229
- parse_header_byte_range() (virttest.http_server.HTTPRequestHandler method), 317
- parse_info_block() (virttest.qemu_qtree.QtreeDisksContainer method), 375
- parse_info_numa() (virttest.qemu_monitor.Monitor class method), 366
- parse_info_qtree() (virttest.qemu_qtree.QtreeContainer method), 374
- parse_string() (virttest.cartesian_config.Parser method), 306
- Parser (class in virttest.cartesian_config), 306
- ParserClass (virttest.xml_utils.TemplateXML attribute), 562
- ParserError, 306
- part_add() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_del() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_disk() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_get_bootable() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_get_mbr_id() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_get_parttype() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_init() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_list() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_set_bootable() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_set_mbr_id() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_set_name() (virttest.utils_libguestfs.GuestfishPersistent method), 446
- part_to_dev() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- part_to_partnum() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- passfd_setup() (in module virttest.passfd_setup), 351
- passwd (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129
- patch() (virttest.build_helper.PatchHelper method), 297
- PatchHelper (class in virttest.build_helper), 297
- PatchParamHelper (class in virttest.build_helper), 297
- Path (class in virttest.element_path), 309
- path (virttest.libvirt_xml.devices.emulator.Emulator attribute), 128
- path (virttest.libvirt_xml.vol_xml.VolXMLBase attribute), 212
- pause() (virttest.libvirt_vm.VM method), 329
- pause() (virttest.qemu_vm.VM method), 388
- pause() (virttest.virt_vm.BaseVM method), 557
- PC1 (virttest.RFBDes.Des attribute), 279
- PC2 (virttest.RFBDes.Des attribute), 279
- pci_label_from_address() (in module virttest.utils_test.libvirt), 266
- PciAssignable (class in virttest.test_setup), 409
- pcic_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 218
- pcihost64 (virttest.libvirt_xml.devices.controller.Controller attribute), 124
- PCIXML (class in virttest.libvirt_xml.nodedev_xml), 187
- PCIXML.Address (class in virttest.libvirt_xml.nodedev_xml), 187
- period (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 197
- persistent (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- PhysicalVolume (class in virttest.lvm), 336
- PI() (in module virttest.element_tree), 310
- pid (virttest.remote_commander.remote_runner.CmdFinish attribute), 229
- pid_exists() (in module virttest.utils_misc), 476
- pin_cpu() (virttest.utils_misc.NumaNode method), 466
- pin_vm_threads() (in module virttest.utils_test.qemu), 274
- ping() (in module virttest.utils_test), 277
- ping() (virttest.ovs_utils.Machine method), 351
- ping4() (in module virttest.ovs_utils), 351
- ping6() (in module virttest.ovs_utils), 351
- ping_daemon() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- polkit_CA_dir (virttest.utils_conn.TLSConnection attribute), 419
- placement (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- playback_compression (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129
- plot_2d_graphs() (virttest.postprocess_iozone.IOzonePlotter method), 353
- plot_3d_graphs() (virttest.postprocess_iozone.IOzonePlotter method), 353
- plot_all() (virttest.postprocess_iozone.IOzonePlotter method), 353
- pm (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- pmsuspend() (virttest.libvirt_vm.VM method), 329
- pmwakeup() (virttest.libvirt_vm.VM method), 329
- PolkitConfigCleanupError, 411
- PolkitConfigError, 411
- PolkitRulesSetupError, 411
- PolkitWriteLibvirtdConfigError, 411
- pool_autostart() (in module virttest.virsh), 536
- pool_build() (in module virttest.virsh), 536

- pool_create() (in module virttest.virsh), 536
- pool_create_as() (in module virttest.virsh), 536
- pool_define() (in module virttest.virsh), 536
- pool_define() (virttest.libvirt_xml.pool_xml.PoolXML method), 192
- pool_define_as() (in module virttest.virsh), 537
- pool_delete() (in module virttest.virsh), 537
- pool_destroy() (in module virttest.virsh), 537
- pool_dumpxml() (in module virttest.virsh), 537
- pool_edit() (in module virttest.virsh), 538
- pool_exists() (virttest.libvirt_storage.StoragePool method), 323
- pool_info() (in module virttest.virsh), 538
- pool_info() (virttest.libvirt_storage.StoragePool method), 323
- pool_list() (in module virttest.virsh), 538
- pool_name() (in module virttest.virsh), 538
- pool_refresh() (in module virttest.virsh), 538
- pool_rename() (virttest.libvirt_xml.pool_xml.PoolXML static method), 192
- pool_start() (in module virttest.virsh), 538
- pool_state() (virttest.libvirt_storage.StoragePool method), 323
- pool_state_dict() (in module virttest.virsh), 539
- pool_type (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
- pool_undefine() (in module virttest.virsh), 539
- pool_undefine() (virttest.libvirt_xml.pool_xml.PoolXML method), 192
- pool_uuid() (in module virttest.virsh), 539
- PoolTestBase (class in virttest.libvirt_storage_unittest), 324
- PoolVolume (class in virttest.libvirt_storage), 321
- PoolVolumeTest (class in virttest.utils_test.libvirt), 262
- PoolXML (class in virttest.libvirt_xml.pool_xml), 191
- PoolXMLBase (class in virttest.libvirt_xml.pool_xml), 192
- pop() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
- pop() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- pop() (virttest.staging.backports.simplejson.OrderedDict method), 242
- popitem() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
- popitem() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- popitem() (virttest.staging.backports.simplejson.OrderedDict method), 242
- port (virttest.libvirt_xml.devices.graphics.Graphics attribute), 129
- port (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- port (virttest.utils_net.IPv6Manager attribute), 481
- port (virttest.utils_sasl.SASL attribute), 497
- port_hi (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 159
- port_to_br() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 345
- port_to_br() (virttest.utils_net.Bridge method), 480
- portgroup (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- PortgroupXML (class in virttest.libvirt_xml.network_xml), 184
- ports (virttest.libvirt_xml.devices.controller.Controller attribute), 124
- post_migration() (virttest.utils_test.qemu.MultihostMigration method), 272
- post_migration() (virttest.utils_test.qemu.MultihostMigrationExec method), 273
- postfix_parse() (in module virttest.cartesian_config), 307
- postprocess_env() (virttest.utils_test.qemu.MultihostMigration method), 272
- postprocess_image() (in module virttest.env_process), 310
- postprocess_images() (in module virttest.storage), 405
- postprocess_on_error() (in module virttest.env_process), 311
- postprocess_vm() (in module virttest.env_process), 311
- power_management_list (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 178
- pre_pool() (virttest.utils_test.libvirt.PoolVolumeTest method), 262
- pre_vol() (virttest.utils_test.libvirt.PoolVolumeTest method), 262
- pread() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- prefix (virttest.libvirt_xml.network_xml.IPXML attribute), 181
- prepare_directory() (virttest.ovs_utils.Machine method), 351
- prepare_for_migration() (virttest.utils_test.qemu.MultihostMigration method), 272
- prepare_guest_agent() (virttest.libvirt_vm.VM method), 329
- preprocess_env() (virttest.utils_test.qemu.MultihostMigration method), 272
- preprocess_image() (in module virttest.env_process), 311
- preprocess_image() (in module virttest.libguestfs), 261
- preprocess_image_backend() (in module virttest.storage), 405
- preprocess_images() (in module virttest.storage), 405
- preprocess_vm() (in module virttest.env_process), 311
- preseed_initrd() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
- present (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199

primary (virttest.libvirt_xml.devices.video.Video attribute), 138

primary_disk_virtio() (in module virttest.utils_test.libguestfs), 261

print_and_inc() (virttest.utils_net_unittest.TestVmNetSubclass.print method), 494

print_dicts() (in module virttest.cartesian_config), 307

print_dicts_default() (in module virttest.cartesian_config), 307

print_dicts_repr() (in module virttest.cartesian_config), 307

print_error() (in module virttest.standalone_test), 402

print_fail() (in module virttest.standalone_test), 402

print_guest_list() (in module virttest.standalone_test), 402

print_header() (in module virttest.standalone_test), 402

print_pass() (in module virttest.standalone_test), 402

print_skip() (in module virttest.standalone_test), 402

print_stdout() (in module virttest.standalone_test), 402

print_test_list() (in module virttest.standalone_test), 402

print_warn() (in module virttest.standalone_test), 403

PRIORITY_NAMES (virttest.syslog_server.RequestHandler attribute), 406

PrivateBridgeConfig (class in virttest.test_setup), 411

PrivateBridgeError, 411

PrivateOvsBridgeConfig (class in virttest.test_setup), 411

process() (in module virttest.env_process), 311

process_command() (in module virttest.env_process), 312

process_images() (in module virttest.env_process), 312

process_info_block() (virttest.qemu_vm.VM method), 388

process_mac() (virttest.utils_net.VMNet method), 484

process_or_children_is_defunct() (in module virttest.utils_misc), 476

process_results() (virttest.postprocess_iozone.IOzoneAnalyzer method), 352

ProcessingInstruction() (in module virttest.element_tree), 310

product (virttest.libvirt_xml.devices.disk.Disk attribute), 128

product (virttest.libvirt_xml.nodedev_xml.SystemXML attribute), 188

product (virttest.libvirt_xml.snapshot_xml.SnapshotXML attribute), 196

product_id (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187

program_is_alive() (in module virttest.utils_misc), 476

prompt (virttest.utils_net.IPv6Manager attribute), 481

prompt (virttest.utils_sasl.SASL attribute), 497

PROMPT_TIMEOUT (virttest.guest_agent.QemuAgent attribute), 314

PROMPT_TIMEOUT (virttest.qemu_monitor.HumanMonitor attribute), 361

PROMPT_TIMEOUT (virttest.qemu_monitor.QMPMonitor attribute), 367

PropCan (class in virttest.propcan), 357

PropCanBase (class in virttest.propcan), 357

PropCanInternal (class in virttest.propcan), 357

property_name (virttest.libvirt_xml.accessors.AccessorBase attribute), 169

property_name (virttest.libvirt_xml.accessors.XMLAttribute.Delter attribute), 170

property_name (virttest.libvirt_xml.accessors.XMLAttribute.Getter attribute), 170

property_name (virttest.libvirt_xml.accessors.XMLAttribute.Setter attribute), 170

property_name (virttest.libvirt_xml.accessors.XMLElementBool.Delter attribute), 171

property_name (virttest.libvirt_xml.accessors.XMLElementBool.Getter attribute), 171

property_name (virttest.libvirt_xml.accessors.XMLElementBool.Setter attribute), 171

property_name (virttest.libvirt_xml.accessors.XMLElementDict.Delter attribute), 172

property_name (virttest.libvirt_xml.accessors.XMLElementDict.Getter attribute), 172

property_name (virttest.libvirt_xml.accessors.XMLElementDict.Setter attribute), 172

property_name (virttest.libvirt_xml.accessors.XMLElementInt.Delter attribute), 172

property_name (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), 172

property_name (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), 173

property_name (virttest.libvirt_xml.accessors.XMLElementList.Delter attribute), 173

property_name (virttest.libvirt_xml.accessors.XMLElementList.Getter attribute), 173

property_name (virttest.libvirt_xml.accessors.XMLElementList.Setter attribute), 174

property_name (virttest.libvirt_xml.accessors.XMLElementNest.Delter attribute), 174

property_name (virttest.libvirt_xml.accessors.XMLElementNest.Getter attribute), 174

property_name (virttest.libvirt_xml.accessors.XMLElementNest.Setter attribute), 174

property_name (virttest.libvirt_xml.accessors.XMLElementText.Delter attribute), 175

property_name (virttest.libvirt_xml.accessors.XMLElementText.Getter attribute), 175

property_name (virttest.libvirt_xml.accessors.XMLElementText.Setter attribute), 175

protocol (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137

protocol_tag (virttest.libvirt_xml.nwfilter_protocols.base.UntypedDeviceBase attribute), 145

protocol_type (virttest.libvirt_xml.devices.console.Console

- attribute), 124
- protocol_type (virttest.libvirt_xml.devices.serial.Serial attribute), 137
- protocol_type (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- protocolid (virttest.libvirt_xml.nwfilter_protocols.mac.Mac attribute), 154
- protocoltype (virttest.libvirt_xml.nwfilter_protocols.arp.Arp.Attribute), 144
- protocoltype (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attribute), 155
- pull_file() (virttest.remote.Remote_Package method), 392
- pull_file() (virttest.utils_netperf.NetperfPackage method), 495
- push_file() (virttest.remote.Remote_Package method), 392
- pvcreeate() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvremove() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvresize() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvresize_size() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvs() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvs_full() (virttest.utils_libguestfs.GuestfishPersistent method), 447
- pvspinlock_state (virttest.libvirt_xml.vm_xml.VMFeaturesXML attribute), 200
- pvuuid() (virttest.utils_libguestfs.GuestfishPersistent method), 448
- pwd() (in module virttest.virsh), 539
- py_encode_basestring_ascii() (in module virttest.staging.backports.simplejson.encoder), 235
- qemu_agent_command() (in module virttest.virsh), 539
- qemu_attach() (in module virttest.virsh), 539
- qemu_has_option() (in module virttest.utils_misc), 476
- qemu_monitor_command() (in module virttest.virsh), 540
- qemu_monitor_event() (in module virttest.virsh), 540
- qemu_verison() (in module virttest.versionable_class_unittest), 508
- QemuAgent (class in virttest.guest_agent), 313
- QemuAutif (class in virttest.utils_net), 483
- QemuImg (class in virttest.libvirt_storage), 321
- QemuImg (class in virttest.qemu_storage), 377
- QemuImg (class in virttest.storage), 403
- QemuIO (class in virttest.qemu_io), 360
- QemuIOParamError, 360
- QemuIOShellSession (class in virttest.qemu_io), 360
- QemuIOSystem (class in virttest.qemu_io), 360
- QemuSegFaultError, 381
- QFloppy (class in virttest.qemu_devices.qdevices), 222
- QFloppyBus (class in virttest.qemu_devices.qbuses), 213
- QGlobal (class in virttest.qemu_devices.qdevices), 222
- QHPDrive (class in virttest.qemu_devices.qdevices), 222
- QIDEBus (class in virttest.qemu_devices.qbuses), 213
- QMPCmdError, 367
- QMPMonitor (class in virttest.qemu_monitor), 367
- QName (class in virttest.element_tree), 310
- QNoAddrCustomBus (class in virttest.qemu_devices.qbuses), 213
- QOldDrive (class in virttest.qemu_devices.qdevices), 223
- QOldFloppyBus (class in virttest.qemu_devices.qbuses), 213
- QPCIBus (class in virttest.qemu_devices.qbuses), 213
- QPCISwitchBus (class in virttest.qemu_devices.qbuses), 213
- QRHDrive (class in virttest.qemu_devices.qdevices), 223
- QSCSIBus (class in virttest.qemu_devices.qbuses), 214
- QSparseBus (class in virttest.qemu_devices.qbuses), 214
- QStrictCustomBus (class in virttest.qemu_devices.qbuses), 215
- QStringDevice (class in virttest.qemu_devices.qdevices), 223
- QtreeBus (class in virttest.qemu_qtree), 374
- QtreeContainer (class in virttest.qemu_qtree), 374
- QtreeContainerTest (class in virttest.qemu_qtree_unittest), 376
- QtreeDev (class in virttest.qemu_qtree), 374
- QtreeDisk (class in virttest.qemu_qtree), 374
- QtreeDiskContainerTest (class in virttest.qemu_qtree_unittest), 376
- QtreeDisksContainer (class in virttest.qemu_qtree), 374
- QtreeNode (class in virttest.qemu_qtree), 375
- query() (virttest.qemu_monitor.HumanMonitor method), 364

Q

- Q (class in virttest.versionable_class_unittest), 506
- Q1 (class in virttest.versionable_class_unittest), 507
- Q_Container (class in virttest.versionable_class_unittest), 507
- q_dict (virttest.cartesian_config.Node attribute), 306
- QAHCIbus (class in virttest.qemu_devices.qbuses), 213
- QBaseDevice (class in virttest.qemu_devices.qdevices), 219
- QBusUnitBus (class in virttest.qemu_devices.qbuses), 213
- QCustomDevice (class in virttest.qemu_devices.qdevices), 221
- QDenseBus (class in virttest.qemu_devices.qbuses), 213
- QDevice (class in virttest.qemu_devices.qdevices), 221
- QDrive (class in virttest.qemu_devices.qdevices), 222
- QDriveBus (class in virttest.qemu_devices.qbuses), 213

[query\(\)](#) (virttest.qemu_monitor.QMPMonitor method), [371](#)
[query_block_job\(\)](#) (virttest.qemu_monitor.HumanMonitor method), [364](#)
[query_block_job\(\)](#) (virttest.qemu_monitor.QMPMonitor method), [371](#)
[queues](#) (virttest.utils_net.QemuIface attribute), [483](#)
[quit\(\)](#) (in module virttest.virsh), [540](#)
[quit\(\)](#) (virttest.qemu_monitor.HumanMonitor method), [364](#)
[quit\(\)](#) (virttest.qemu_monitor.QMPMonitor method), [371](#)
[quit\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), [448](#)
[quota](#) (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), [198](#)
[QUSBBus](#) (class in virttest.qemu_devices.qbuses), [215](#)

R

[radix](#) (virttest.libvirt_xml.accessors.XMLElementInt.Getter attribute), [173](#)
[radix](#) (virttest.libvirt_xml.accessors.XMLElementInt.Setter attribute), [173](#)
[RangeList](#) (class in virttest.libvirt_xml.network_xml), [185](#)
[Rarp](#) (class in virttest.libvirt_xml.nwfilter_protocols.rarp), [155](#)
[Rarp.Attr](#) (class in virttest.libvirt_xml.nwfilter_protocols.rarp), [155](#)
[rate](#) (virttest.libvirt_xml.devices.rng.Rng attribute), [136](#)
[raw_decode\(\)](#) (virttest.staging.backports.simplejson.decoder.JSONDecoder method), [233](#)
[raw_decode\(\)](#) (virttest.staging.backports.simplejson.JSONDecoder method), [240](#)
[raw_ping\(\)](#) (in module virttest.utils_test), [278](#)
[raw_status_parser\(\)](#) (in module virttest.staging.service), [246](#)
[Rawdev](#) (class in virttest.storage), [404](#)
[rawio](#) (virttest.libvirt_xml.devices.disk.Disk attribute), [128](#)
[rawio](#) (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), [196](#)
[re_numa_node_info](#) (virttest.qemu_monitor.Monitor attribute), [366](#)
[re_numa_nodes](#) (virttest.qemu_monitor.Monitor attribute), [366](#)
[read\(\)](#) (virttest.remote_commander.messenger.IOWrapper method), [225](#)
[read\(\)](#) (virttest.remote_commander.messenger.StdIOWrapper method), [226](#)
[read\(\)](#) (virttest.xml_utils.XMLTreeFile method), [563](#)
[read_bytes_sec](#) (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), [127](#)
[read_config\(\)](#) (virttest.staging.utils_koji.KojiClient method), [253](#)
[read_file\(\)](#) (virttest.utils_disk.GuestFSModiDisk method), [421](#)
[read_file\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), [448](#)
[read_from_meminfo\(\)](#) (in module virttest.staging.utils_memory), [256](#)
[read_from_node_meminfo\(\)](#) (virttest.utils_misc.NumaInfo method), [465](#)
[read_from_numa_maps\(\)](#) (in module virttest.staging.utils_memory), [256](#)
[read_from_numastat\(\)](#) (in module virttest.staging.utils_memory), [256](#)
[read_from_smaps\(\)](#) (in module virttest.staging.utils_memory), [256](#)
[read_from_vmstat\(\)](#) (in module virttest.staging.utils_memory), [257](#)
[read_iops_sec](#) (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), [127](#)
[read_msg\(\)](#) (virttest.remote_commander.messenger.Messenger method), [225](#)
[read_nonblocking\(\)](#) (virttest.aexpect.Expect method), [280](#)
[read_nonblocking\(\)](#) (virttest.qemu_virtio_port.GuestWorker method), [380](#)
[READ_OBJECTS_TIMEOUT](#) (virttest.guest_agent.QemuAgent attribute), [314](#)
[READ_OBJECTS_TIMEOUT](#) (virttest.qemu_monitor.QMPMonitor attribute), [367](#)
[read_until_any_line_matches\(\)](#) (virttest.aexpect.Expect method), [281](#)
[read_until_last_line_matches\(\)](#) (virttest.aexpect.Expect method), [281](#)
[read_until_last_word_matches\(\)](#) (virttest.aexpect.Expect method), [282](#)
[read_until_output_matches\(\)](#) (virttest.aexpect.Expect method), [282](#)
[read_until_output_matches\(\)](#) (virttest.virsh.VirshSession method), [511](#)
[read_until_prompt\(\)](#) (virttest.aexpect.ShellSession method), [285](#)
[readdir\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), [448](#)
[readonly](#) (virttest.libvirt_xml.devices.disk.Disk attribute), [128](#)
[readonly](#) (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), [196](#)
[readonly](#) (virttest.virsh.VirshBase attribute), [509](#)
[readonly](#) (virttest.virsh.VirshPersistent attribute), [510](#)
[ready](#) (virttest.libvirt_xml.devices.disk.Disk attribute), [128](#)
[ready](#) (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), [196](#)
[rebase\(\)](#) (virttest.libvirt_storage.QemuImg method), [322](#)

- rebase() (virttest.qemu_storage.QemuImg method), 378
- reboot() (in module virttest.virsh), 540
- reboot() (virttest.libvirt_vm.VM method), 329
- reboot() (virttest.qemu_vm.VM method), 388
- reboot() (virttest.virt_vm.BaseVM method), 557
- REBOOT_TIMEOUT (virttest.virt_vm.BaseVM attribute), 554
- reboot_windows() (virttest.utils_v2v.WindowsVMCheck method), 504
- reconnect() (virttest.qemu_virtio_port.GuestWorker method), 380
- RECORD_RE (virttest.syslog_server.RequestHandler attribute), 406
- recover_fds() (virttest.remote_commander.remote_runner.RemoteRunner method), 229
- recover_paths() (virttest.remote_commander.remote_runner.RemoteRunner method), 229
- recv() (virttest.lvsb_base.SandboxBase method), 339
- recv() (virttest.lvsb_base.SandboxSession method), 341
- recvrr() (virttest.lvsb_base.SandboxBase method), 339
- recvrr() (virttest.lvsb_base.SandboxSession method), 341
- recvout() (virttest.lvsb_base.SandboxBase method), 339
- recvout() (virttest.lvsb_base.SandboxSession method), 341
- Redirdev (in module virttest.libvirt_xml.devices.redirdev), 135
- reduce_pvc() (virttest.lvm.VolumeGroup method), 338
- refresh_neigh_table() (in module virttest.utils_net), 490
- register() (in module virttest.funcatexit), 312
- register() (virttest.installer.InstallerRegistry method), 317
- register() (virttest.lvm.LVM method), 335
- register_cmd() (virttest.remote_commander.remote_runner.RemoteRunner method), 230
- register_lvmdev() (virttest.utils_env.Env method), 423
- register_syncserver() (virttest.utils_env.Env method), 423
- register_vm() (virttest.utils_env.Env method), 423
- relabel (virttest.libvirt_xml.devices.seclabel.Seclabel attribute), 137
- release_devs() (virttest.test_setup.PciAssignable method), 410
- reload_loss_idx() (virttest.qemu_virtio_port.ThRecvCheck method), 380
- reload_modules() (virttest.base_installer.BaseInstaller method), 291
- reload_modules_if_needed() (virttest.base_installer.BaseInstaller method), 291
- remote_agent() (in module virttest.remote_commander.remote_runner), 231
- remote_commander() (in module virttest.remote), 394
- remote_commander() (virttest.virt_vm.BaseVM method), 558
- remote_ip (virttest.virsh.VirshConnectBack attribute), 509
- remote_ip (virttest.virsh.VirshPersistent attribute), 510
- remote_libvirtdconf (virttest.utils_conn.TCPConnection attribute), 418
- remote_login() (in module virttest.remote), 394
- remote_login() (virttest.virt_vm.BaseVM method), 558
- RemotePackage (class in virttest.remote), 392
- remote_pwd (virttest.virsh.VirshPersistent attribute), 510
- remote_scp() (in module virttest.remote), 395
- remote_syslibvirtd (virttest.utils_conn.TCPConnection attribute), 418
- remote_user (virttest.virsh.VirshPersistent attribute), 510
- RemoteDiskManager (class in virttest.utils_test), 275
- RemoteFile (class in virttest.remote), 391
- RemoteFileTest (class in virttest.remote_unittest), 397
- RemoteInstall (class in virttest.tests.unattended_install), 257
- remotely_control_libvirtd() (in module virttest.utils_test.libvirt), 266
- RemoteRunner (class in virttest.remote), 391
- RemoteSourceTarInstaller (class in virttest.base_installer), 292
- RemoteSourceTarInstaller (class in virttest.qemu_installer), 360
- RemoteTarHelper (class in virttest.build_helper), 297
- RemoteTarParamHelper (class in virttest.build_helper), 298
- remove() (virttest.libvirt_storage.QemuImg method), 322
- remove() (virttest.libvirt_vm.VM method), 329
- remove() (virttest.lvm.LogicalVolume method), 336
- remove() (virttest.lvm.PhysicalVolume method), 337
- remove() (virttest.lvm.StorageVolumeGroup method), 338
- remove() (virttest.qemu_devices.qbuses.QSparseBus method), 215
- remove() (virttest.qemu_devices.qcontainer.DevContainer method), 218
- remove() (virttest.qemu_storage.QemuImg method), 379
- remove() (virttest.remote.RemoteFile method), 391
- remove() (virttest.xml_utils.XMLTreeFile method), 563
- remove() (virttest.yumrepo.YumRepo method), 566
- remove_agent_channels() (virttest.libvirt_xml.vm_xml.VMXML method), 206
- remove_all_boots() (virttest.libvirt_xml.vm_xml.VMXML method), 206
- remove_all_device_by_type() (virttest.libvirt_xml.vm_xml.VMXML method), 206
- remove_all_graphics() (virttest.libvirt_xml.vm_xml.VMXML method), 206
- remove_by_xpath() (virttest.xml_utils.XMLTreeFile method), 564
- remove_command_echo() (virttest.aexpect.ShellSession

- class method), 286
- remove_domain() (in module virttest.virsh), 540
- remove_feature() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 259
- method), 178
- remove_feature() (virttest.libvirt_xml.vm_xml.VMCPXML method), 199
- remove_feature() (virttest.libvirt_xml.vm_xml.VMFeaturesXML method), 200
- remove_last_nonempty_line() (virttest.aexpect.ShellSession class method), 286
- remove_package() (virttest.libvirt_vm.VM method), 329
- remove_path() (virttest.utils_test.RemoteDiskManager method), 275
- remove_vg() (virttest.utils_test.RemoteDiskManager method), 275
- remove_with_storage() (virttest.libvirt_vm.VM method), 329
- render() (virttest.yumrepo.YumRepo method), 566
- reopen() (virttest.utils_libguestfs.GuestfishPersistent method), 448
- replace_child() (virttest.qemu_qtree.QtreeNode method), 375
- replace_image_file_content() (virttest.utils_disk.GuestFSModiDisk method), 421
- report() (virttest.postprocess_iozone.IOzoneAnalyzer method), 352
- report_comparison() (virttest.postprocess_iozone.IOzoneAnalyzer method), 352
- request_devs() (virttest.test_setup.PciAssignable method), 410
- RequestHandler (class in virttest.syslog_server), 405
- RequestHandlerTcp (class in virttest.syslog_server), 406
- RequestHandlerUdp (class in virttest.syslog_server), 406
- required_dargs (virttest.libvirt_xml.accessors.XMLElementBool attribute), 171
- required_dargs (virttest.libvirt_xml.accessors.XMLElementInt attribute), 173
- required_dargs (virttest.libvirt_xml.accessors.XMLElementText attribute), 174
- required_dargs (virttest.libvirt_xml.accessors.XMLElementText attribute), 175
- required_dargs (virttest.libvirt_xml.accessors.XMLElementText attribute), 175
- requires_action() (virttest.cartesian_config.NoFilter method), 305
- requires_action() (virttest.cartesian_config.OnlyFilter method), 306
- reroot() (virttest.xml_utils.XMLTreeFile method), 564
- rescan() (virttest.lvm.LVM method), 335
- reserve() (virttest.qemu_devices.qbuses.QSparseBus method), 215
- reset() (in module virttest.virsh), 540
- reset_export() (virttest.nfs.Exportfs method), 342
- reset_interface() (virttest.utils_test.libguestfs.GuestfishTools method), 483
- reset_ip() (virttest.utils_net.ParamsNet method), 483
- reset_logging() (in module virttest.standalone_test), 403
- reset_mac() (virttest.utils_net.ParamsNet method), 483
- reset_state() (virttest.qemu_devices.qcontainer.DevContainer method), 218
- resize() (virttest.lvm.LogicalVolume method), 336
- resize() (virttest.lvm.PhysicalVolume method), 337
- resize2fs() (virttest.utils_libguestfs.GuestfishPersistent method), 448
- resize2fs_M() (virttest.utils_libguestfs.GuestfishPersistent method), 448
- resize2fs_size() (virttest.utils_libguestfs.GuestfishPersistent method), 448
- resolve_task_cgroup_path() (in module virttest.staging.utils_cgroup), 250
- RESPONSE_TIMEOUT (virttest.guest_agent.QemuAgent attribute), 314
- RESPONSE_TIMEOUT (virttest.qemu_monitor.QMPMonitor attribute), 367
- rest_line() (virttest.cartesian_config.Lexer method), 305
- rest_line_as_LString() (virttest.cartesian_config.Lexer method), 305
- rest_line_gen() (virttest.cartesian_config.Lexer method), 305
- rest_line_no_white() (virttest.cartesian_config.Lexer method), 305
- restart() (virttest.nfs_unittest.FakeService method), 343
- restart() (virttest.openvswitch.ServiceManagerInterface method), 346
- restart() (virttest.openvswitch.ServiceManagerSystemD method), 346
- restart() (virttest.openvswitch.ServiceManagerSysvinit method), 346
- restart() (virttest.utils_libvirtd.Libvirtd method), 462
- restart() (virttest.utils_libvirtd.LibvirtdSession method), 463
- restart_guest_network() (in module virttest.utils_net), 490
- restart_ksm() (virttest.utils_misc.KSMController method), 464
- restart_ksmtuned() (virttest.utils_misc.KSMController method), 464
- restart_libvirtd (virttest.utils_conn.TLSConnection attribute), 419
- restart_libvirtd (virttest.utils_conn.UNIXConnection attribute), 419
- restart_vdagent() (in module virttest.utils_spice), 501
- restart_windows_guest_network() (in module virttest.utils_net), 491
- restart_windows_guest_network_by_devcon() (in module

virttest.utils_net), 491
 restart_windows_guest_network_by_key() (in module virttest.utils_net), 491
 restore() (in module virttest.virsh), 541
 restore() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176
 restore() (virttest.utils_config.SectionlessConfig method), 414
 restore() (virttest.xml_utils.TemplateXML method), 563
 restore() (virttest.xml_utils.XMLBackup method), 563
 restore() (virttest.xml_utils.XMLTreeFile method), 564
 restore_from_file() (virttest.libvirt_vm.VM method), 330
 restore_from_file() (virttest.qemu_vm.VM method), 389
 restore_from_file() (virttest.virt_vm.BaseVM method), 558
 RestoreconError, 497
 ResultParserTest (class in virttest.service_unittest), 400
 results (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
 results() (virttest.lvsb.TestBaseSandboxes method), 338
 resume() (in module virttest.virsh), 541
 resume() (virttest.libvirt_vm.VM method), 330
 resume() (virttest.ovirt.VMManager method), 349
 resume() (virttest.qemu_vm.VM method), 389
 resume() (virttest.virt_vm.BaseVM method), 558
 resume_guest_disk() (virttest.utils_test.qemu.GuestSuspend method), 269
 resume_guest_mem() (virttest.utils_test.qemu.GuestSuspend method), 269
 RETRY_STEP (virttest.staging.utils_koji.KojiClient attribute), 251
 RETRY_TIMEOUT (virttest.staging.utils_koji.KojiClient attribute), 251
 rm() (virttest.utils_libguestfs.GuestfishPersistent method), 448
 rm_cgroup() (virttest.staging.utils_cgroup.Cgroup method), 249
 rm_child_bus() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
 rm_cloned_image() (virttest.storage.QemuImg static method), 404
 rm_ker_cmd() (in module virttest.utils_misc), 476
 rm_rf() (virttest.utils_libguestfs.GuestfishPersistent method), 448
 rmmountpoint() (virttest.utils_libguestfs.GuestfishPersistent method), 448
 Rng (class in virttest.libvirt_xml.devices.rng), 136
 Rng.Backend (class in virttest.libvirt_xml.devices.rng), 136
 rng_model (virttest.libvirt_xml.devices.rng.Rng attribute), 136
 romfile (virttest.utils_net.QemuIface attribute), 483
 root_address (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 root_address_mask (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 root_cost (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 root_cost_hi (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 root_priority (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 root_priority_hi (virttest.libvirt_xml.nwfilter_protocols.stp.StpAttr attribute), 159
 rounded_memtotal() (in module virttest.staging.utils_memory), 257
 routes (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
 RPMFileNameInfo (class in virttest.staging.utils_koji), 255
 rsync() (virttest.utils_libguestfs.GuestfishPersistent method), 448
 rsync_in() (virttest.utils_libguestfs.GuestfishPersistent method), 449
 rsync_out() (virttest.utils_libguestfs.GuestfishPersistent method), 449
 rule_action (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules attribute), 191
 rule_direction (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules attribute), 191
 rule_priority (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules attribute), 191
 rule_statematch (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLRules attribute), 191
 run() (virttest.lvsb_base.SandboxBase method), 339
 run() (virttest.qemu_virtio_port.ThRecv method), 380
 run() (virttest.qemu_virtio_port.ThRecvCheck method), 380
 run() (virttest.qemu_virtio_port.ThSend method), 380
 run() (virttest.qemu_virtio_port.ThSendCheck method), 381
 run() (virttest.remote.RemoteRunner method), 391
 run() (virttest.utils_gdb.GDB method), 426
 run() (virttest.utils_libguestfs.GuestfishPersistent method), 449
 run() (virttest.utils_test.qemu.MultihostMigration method), 272
 run() (virttest.virsh_unittest.ModuleLoadCheckVirsh method), 552
 run_autotest() (in module virttest.utils_test), 278
 run_bg() (in module virttest.aexpect), 288
 run_debug() (virttest.qemu_virtio_port.ThRecvCheck method), 380
 run_exitfuncs() (in module virttest.funcatexit), 313
 run_fg() (in module virttest.aexpect), 288
 run_installers() (in module virttest.installer), 318
 run_mode (virttest.utils_libguestfs.GuestfishPersistent attribute), 449

`run_normal()` (`virttest.qemu_virtio_port.ThRecvCheck` method), 380
`run_once()` (`virttest.standalone_test.Test` method), 401
`run_tail()` (in module `virttest.aexpect`), 289
`run_tests()` (in module `virttest.standalone_test`), 403
`run_tests()` (in module `virttest.utils_misc`), 476
`run_virt_sub_test()` (in module `virttest.utils_test`), 278
`runner` (`virttest.utils_net.IPv6Manager` attribute), 481
`running()` (`virttest.lvsb_base.SandboxBase` method), 339
`RVConnectError`, 500

S

`safe_exit_loopback_threads()`
(`virttest.qemu_virtio_port.GuestWorker` method), 380
`safe_kill()` (in module `virttest.utils_misc`), 477
`SANDBOX_TYPE` (`virttest.lvsb_base.TestSandboxes` attribute), 341
`SandboxBase` (class in `virttest.lvsb_base`), 339
`SandboxCommandBase` (class in `virttest.lvsb_base`), 339
`SandboxException`, 340
`SandboxService` (class in `virttest.lvsbs`), 341
`SandboxSession` (class in `virttest.lvsb_base`), 340
`SASL` (class in `virttest.utils_sasl`), 496
`sasl_allowed_users` (`virttest.utils_conn.TCPConnection` attribute), 418
`sasl_pwd_cmd` (`virttest.utils_sasl.SASL` attribute), 497
`sasl_user_cmd` (`virttest.utils_sasl.SASL` attribute), 497
`sasl_user_pwd` (`virttest.utils_sasl.SASL` attribute), 497
`save()` (in module `virttest.virsh`), 541
`save()` (`virttest.utils_env.Env` method), 423
`save()` (`virttest.yumrepo.YumRepo` method), 566
`save_image_define()` (in module `virttest.virsh`), 541
`save_image_dumpxml()` (in module `virttest.virsh`), 541
`save_to_db()` (`virttest.utils_net.DbNet` method), 480
`save_to_file()` (`virttest.libvirt_vm.VM` method), 330
`save_to_file()` (`virttest.qemu_vm.VM` method), 389
`save_to_file()` (`virttest.virt_vm.BaseVM` method), 558
`savevm()` (`virttest.qemu_vm.VM` method), 389
`savevm()` (`virttest.virt_vm.BaseVM` method), 558
`sbox` (`virttest.RFBDes.Des` attribute), 280
`schedinfo()` (in module `virttest.virsh`), 541
`scheduler` (class in `virttest.scheduler`), 400
`scheduler()` (`virttest.scheduler.scheduler` method), 400
`scp_between_remotes()` (in module `virttest.remote`), 395
`scp_from_remote()` (in module `virttest.remote`), 396
`scp_new_cacert` (`virttest.utils_conn.TLSConnection` attribute), 419
`scp_to_remote()` (in module `virttest.remote`), 396
`SCPAAuthenticationError`, 392
`SCPAAuthenticationTimeoutError`, 392
`SCPErrror`, 392
`SCPTransferFailedError`, 392
`SCPTransferTimeoutError`, 392

`screendump()` (`virttest.libvirt_vm.VM` method), 330
`screendump()` (`virttest.qemu_monitor.HumanMonitor` method), 364
`screendump()` (`virttest.qemu_monitor.QMPMonitor` method), 371
`screendump()` (`virttest.qemu_vm.VM` method), 389
`screenshot()` (in module `virttest.virsh`), 542
`screenshot_test()` (in module `virttest.virsh`), 542
`scrub_device()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 449
`scrub_file()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 449
`scrub_freespace()` (`virttest.utils_libguestfs.GuestfishPersistent` method), 449
`Sctp` (class in `virttest.libvirt_xml.nwfilter_protocols.sctp`), 156
`Sctp.Attr` (class in `virttest.libvirt_xml.nwfilter_protocols.sctp`), 156
`Sctp_ipv6` (class in `virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6`), 157
`Sctp_ipv6.Attr` (class in `virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6`), 157
`Seclabel` (class in `virttest.libvirt_xml.devices.seclabel`), 136
`seclabel` (`virttest.libvirt_xml.vm_xml.VMXMLBase` attribute), 210
`seclabels` (`virttest.libvirt_xml.devices.disk.Disk.DiskSource` attribute), 126
`SeCmdError`, 497
`secret` (`virttest.libvirt_xml.devices.disk.Disk.Encryption` attribute), 127
`secret` (`virttest.libvirt_xml.vol_xml.VolXML.Encryption` attribute), 211
`secret_define()` (in module `virttest.virsh`), 542
`secret_dumpxml()` (in module `virttest.virsh`), 542
`secret_ephemeral` (`virttest.libvirt_xml.secret_xml.SecretXMLBase` attribute), 195
`secret_get_value()` (in module `virttest.virsh`), 542
`secret_list()` (in module `virttest.virsh`), 542
`secret_private` (`virttest.libvirt_xml.secret_xml.SecretXMLBase` attribute), 195
`secret_set_value()` (in module `virttest.virsh`), 543
`secret_type` (`virttest.libvirt_xml.devices.disk.Disk.Auth` attribute), 126
`secret_undefine()` (in module `virttest.virsh`), 543
`secret_usage` (`virttest.libvirt_xml.devices.disk.Disk.Auth` attribute), 126
`secret_usage` (`virttest.libvirt_xml.pool_xml.SourceXML` attribute), 194
`secret_uuid` (`virttest.libvirt_xml.devices.disk.Disk.Auth` attribute), 126
`secret_uuid` (`virttest.libvirt_xml.pool_xml.SourceXML` attribute), 194

- SecretXML (class in virttest.libvirt_xml.secret_xml), 194
- SecretXMLBase (class in virttest.libvirt_xml.secret_xml), 194
- SectionlessConfig (class in virttest.utils_config), 413
- SectionlessConfigTest (class in virttest.utils_config_unittest), 415
- selinux_enforcing() (in module virttest.utils_misc), 477
- SELinuxBoolean (class in virttest.utils_misc), 466
- SelinuxError, 497
- SemanageError, 497
- send() (virttest.aexpect.Spawn method), 287
- send() (virttest.lvsb_base.SandboxBase method), 339
- send() (virttest.lvsb_base.SandboxSession method), 341
- send_args_cmd() (virttest.qemu_monitor.HumanMonitor method), 364
- send_args_cmd() (virttest.qemu_monitor.QMPMonitor method), 371
- send_ctrl() (virttest.aexpect.Spawn method), 287
- send_fd() (virttest.qemu_vm.VM method), 389
- send_head_range() (virttest.http_server.HTTPRequestHandler method), 317
- send_key() (virttest.qemu_vm.VM method), 389
- send_key() (virttest.virt_vm.BaseVM method), 558
- send_msg() (virttest.remote_commander.remote_runner.Commander method), 230
- send_signal() (virttest.utils_gdb.GDB method), 426
- send_stdin() (virttest.remote_commander.remote_master.CmdMaster method), 228
- send_string() (virttest.virt_vm.BaseVM method), 558
- send_win32_key() (virttest.utils_v2v.WindowsVMCheck method), 504
- sender_address (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 160
- sender_address_mask (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 160
- sender_priority (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 160
- sender_priority_hi (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 160
- sendkey() (in module virttest.virsh), 543
- sendkey() (virttest.qemu_monitor.HumanMonitor method), 365
- sendkey() (virttest.qemu_monitor.QMPMonitor method), 372
- sendline() (virttest.aexpect.Spawn method), 287
- SEP (virttest.staging.utils_koji.KojiPkgSpec attribute), 254
- SEP (virttest.staging.utils_koji.KojiScratchPkgSpec attribute), 255
- Serial (class in virttest.libvirt_xml.devices.serial), 137
- serial (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- serial (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapshotXML attribute), 196
- serial_login() (virttest.virt_vm.BaseVM method), 558
- SERIAL_TYPE_ISA (virttest.guest_agent.QemuAgent attribute), 314
- SERIAL_TYPE_VIRTIO (virttest.guest_agent.QemuAgent attribute), 314
- server_cn (virttest.utils_conn.TLSConnection attribute), 419
- server_ifname (virttest.utils_net.IPv6Manager attribute), 481
- server_ip (virttest.utils_conn.ConnectionBase attribute), 416
- server_ip (virttest.utils_net.IPv6Manager attribute), 481
- server_ip (virttest.utils_sasl.SASL attribute), 497
- server_ipv6_addr (virttest.utils_net.IPv6Manager attribute), 481
- server_libvirtdconf (virttest.utils_conn.TLSConnection attribute), 419
- server_pwd (virttest.utils_conn.ConnectionBase attribute), 416
- server_pwd (virttest.utils_net.IPv6Manager attribute), 482
- server_pwd (virttest.utils_sasl.SASL attribute), 497
- server_setup() (virttest.utils_conn.ConnectionBase attribute), 416
- server_syslibvirtd (virttest.utils_conn.TLSConnection attribute), 419
- server_user (virttest.utils_conn.ConnectionBase attribute), 416
- server_user (virttest.utils_net.IPv6Manager attribute), 482
- server_user (virttest.utils_sasl.SASL attribute), 497
- service_libvirtd_control() (in module virttest.utils_libvirtd), 463
- service_name (virttest.lvsbs.SandboxService attribute), 341
- service_setup() (in module virttest.utils_test), 278
- ServiceManager (class in virttest.openvswitch), 346
- ServiceManagerInterface (class in virttest.openvswitch), 346
- ServiceManagerSystemD (class in virttest.openvswitch), 346
- ServiceManagerSysvinit (class in virttest.openvswitch), 346
- session (virttest.utils_net.IPv6Manager attribute), 482
- session (virttest.utils_sasl.SASL attribute), 497
- SESSION_COUNTER (virttest.utils_libguestfs.GuestfishPersistent attribute), 427
- session_id (virttest.lvsb_base.SandboxSession attribute), 341
- SnapshotXML (virttest.utils_libguestfs.GuestfishPersistent attribute), 449

- session_id (virttest.virsh.VirshPersistent attribute), 510
- SessionManagerTest (class in virttest.virsh_unittest), 552
- set_active() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_agent_channel() (virttest.libvirt_xml.vm_xml.VMXML method), 206
- set_aid() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- set_append() (virttest.utils_libguestfs.GuestfishPersistent method), 449
- set_attach_method() (virttest.utils_libguestfs.GuestfishPersistent method), 449
- set_autostart() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_autosync() (virttest.utils_libguestfs.GuestfishPersistent method), 449
- set_backend() (virttest.utils_libguestfs.GuestfishPersistent method), 449
- set_backing_data_dir() (in module virttest.data_dir), 309
- set_block_job_speed() (virttest.qemu_monitor.HumanMonitor method), 365
- set_block_job_speed() (virttest.qemu_monitor.QMPMonitor method), 372
- set_block_prop() (virttest.qemu_qtree.QtreeDisk method), 374
- set_boolean() (virttest.utils_config.SectionlessConfig method), 414
- set_callback() (virttest.utils_gdb.GDB method), 426
- set_callback() (virttest.utils_libvirtd.LibvirtdSession method), 463
- set_cap() (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase method), 186
- set_cgroup() (virttest.staging.utils_cgroup.Cgroup method), 249
- set_channel() (virttest.libvirt_xml.devices.graphics.Graphics method), 129
- set_chap_acls_target() (virttest.iscsi.IscsiLIO method), 318
- set_chap_auth_target() (virttest.iscsi.IscsiLIO method), 318
- set_chap_auth_target() (virttest.iscsi.IscsiTGT method), 319
- set_clean() (virttest.qemu_devices.qcontainer.DevContainers method), 218
- set_client_session() (virttest.utils_conn.ConnectionBase method), 416
- set_commander() (virttest.remote_commander.remote_master.RemoteMaster method), 228
- set_console_getty() (virttest.libvirt_vm.VM method), 330
- set_context_of_file() (in module virttest.utils_selinux), 499
- set_controller() (virttest.libvirt_xml.vm_xml.VMXMLBase method), 210
- set_controller_multifunction() (in module virttest.utils_test.libvirt), 266
- set_cpu_mode() (virttest.libvirt_xml.vm_xml.VMXML static method), 206
- set_cpu_status() (in module virttest.utils_misc), 477
- set_debug() (virttest.utils_libguestfs.LibguestfsBase method), 458
- set_debugdir() (virttest.standalone_test.Test method), 401
- set_default_format() (in module virttest.syslog_server), 407
- set_default_koji_tag() (in module virttest.staging.utils_koji), 255
- set_defcon() (in module virttest.bootstrap), 293
- set_defcon() (in module virttest.utils_selinux), 499
- set_defined() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_device() (virttest.qemu_devices.qbuses.QSparseBus method), 215
- set_devices() (virttest.libvirt_xml.vm_xml.VMXMLBase method), 210
- set_direct() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_dirty() (virttest.qemu_devices.qcontainer.DevContainer method), 218
- set_disks() (virttest.libvirt_xml.snapshot_xml.SnapshotXML method), 196
- set_domain_state() (in module virttest.utils_test.libvirt), 267
- set_e2attr() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_e2generation() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_e2label() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_e2uuid() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_env() (virttest.test_setup.TransparentHugePageConfig method), 412
- set_fast() (virttest.cartesian_config.Lexer method), 305
- set_feature() (virttest.libvirt_xml.capability_xml.CapabilityXML method), 178
- set_feature() (virttest.libvirt_xml.vm_xml.VMCPUXML method), 199
- set_float() (virttest.utils_config.SectionlessConfig method), 414
- set_guest_agent() (in module virttest.utils_test.libvirt), 267
- set_guest_network_status_by_devcon() (in module virttest.utils_net), 491
- set_hugepages() (virttest.test_setup.HugePageConfig method), 408
- set_if_none() (virttest.propcan.PropCan method), 357
- set_if_not_defined() (virttest.libvirt_xml.accessors.AccessorGeneratorBase method), 169
- set_if_value_not_none() (virttest.propcan.PropCan method), 357

- method), 357
- set_ignore_status() (virttest.utils_libguestfs.LibguestfsBase method), 458
- set_install_params() (virttest.base_installer.BaseInstaller method), 291
- set_install_params() (virttest.base_installer.BaseLocalSourceInstaller method), 291
- set_install_params() (virttest.base_installer.GitRepoInstaller method), 292
- set_install_params() (virttest.base_installer.KojiInstaller method), 292
- set_install_params() (virttest.base_installer.LocalSourceDirInstaller method), 292
- set_install_params() (virttest.base_installer.LocalSourceTarInstaller method), 292
- set_install_params() (virttest.base_installer.RemoteSourceTarInstaller method), 292
- set_install_params() (virttest.base_installer.YumInstaller method), 293
- set_int() (virttest.utils_config.SectionlessConfig method), 414
- set_ip() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_ip() (virttest.utils_net.Interface method), 482
- set_job_speed() (virttest.qemu_vm.VM method), 389
- set_kernel_console() (virttest.libvirt_vm.VM method), 330
- set_kernel_param() (virttest.libvirt_vm.VM method), 330
- set_ksm_feature() (virttest.utils_misc.KSMController method), 464
- set_label() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_linesep() (virttest.aexpect.Spawn method), 287
- set_link() (virttest.qemu_monitor.HumanMonitor method), 365
- set_link() (virttest.qemu_monitor.QMPMonitor method), 372
- set_link() (virttest.qemu_vm.VM method), 389
- set_list() (virttest.utils_config.SectionlessConfig method), 414
- set_listens() (virttest.libvirt_xml.devices.graphics.Graphics method), 130
- set_log_file() (virttest.aexpect.Tail method), 287
- set_log_file_dir() (in module virttest.utils_misc), 477
- set_mac() (virttest.utils_net.Interface method), 482
- set_mac_address() (virttest.utils_net.VirtNet method), 486
- set_memoryBacking_tag() (virttest.libvirt_xml.vm_xml.VMXML static method), 206
- set_memsize() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_multiqueues() (virttest.libvirt_xml.vm_xml.VMXML static method), 206
- set_net_if_ip() (in module virttest.utils_net), 491
- set_netmask() (virttest.utils_net.Interface method), 482
- set_network() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_next_line() (virttest.cartesian_config.StrReader method), 307
- set_node_num_huge_pages() (virttest.test_setup.HugePageConfig method), 408
- set_num_huge_pages() (in module virttest.staging.utils_memory), 257
- set_operands() (virttest.cartesian_config.LApplyPreDict method), 301
- set_operands() (virttest.cartesian_config.LOperators method), 302
- set_operands() (virttest.cartesian_config.LUpdateFileMap method), 304
- set_output_func() (virttest.aexpect.Tail method), 287
- set_output_params() (virttest.aexpect.Tail method), 288
- set_output_prefix() (virttest.aexpect.Tail method), 288
- set_param() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- set_param() (virttest.qemu_devices.qdevices.QDrive method), 222
- set_param() (virttest.qemu_devices.qdevices.QFloppy method), 222
- set_param() (virttest.qemu_devices.qdevices.QOldDrive method), 223
- set_parent() (virttest.qemu_qtree.QTreeNode method), 375
- set_path() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_persistent() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_pgroup() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_pm_suspend() (virttest.libvirt_xml.vm_xml.VMXML static method), 207
- set_pool_autostart() (virttest.libvirt_storage.StoragePool method), 323
- set_portgroup() (virttest.libvirt_xml.network_xml.NetworkXMLBase method), 184
- set_prev_indent() (virttest.cartesian_config.Lexer method), 305
- set_primary_serial() (virttest.libvirt_xml.vm_xml.VMXML static method), 207
- set_program() (virttest.utils_libguestfs.GuestfishPersistent method), 450
- set_prompt() (virttest.aexpect.ShellSession method), 286
- set_property() (virttest.staging.utils_cgroup.Cgroup method), 249
- set_property_h() (virttest.staging.utils_cgroup.Cgroup method), 249
- set_qemu() (virttest.utils_libguestfs.GuestfishPersistent

method), 450

set_qtree() (virttest.qemu_qtree.QtreeNode method), 375

set_qtree_prop() (virttest.qemu_qtree.QtreeNode method), 375

set_raw() (virttest.utils_config.SectionlessConfig method), 414

set_recovery_proc() (virttest.utils_libguestfs.GuestfishPersistent method), 450

set_root_cgroup() (virttest.staging.utils_cgroup.Cgroup method), 250

set_root_serial_console() (virttest.libvirt_vm.VM method), 330

set_rule() (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase method), 190

set_seclabel() (virttest.libvirt_xml.vm_xml.VMXMLBase method), 210

set_server_session() (virttest.utils_conn.ConnectionBase method), 416

set_smp() (virttest.utils_libguestfs.GuestfishPersistent method), 451

set_source() (virttest.libvirt_xml.pool_xml.PoolXMLBase method), 193

set_sources() (virttest.libvirt_xml.devices.character.CharacterXMLBase method), 123

set_status() (in module virttest.utils_selinux), 499

set_status_test_command() (virttest.aexpect.ShellSession method), 286

set_strict() (virttest.cartesian_config.Lexer method), 305

set_string() (virttest.utils_config.SectionlessConfig method), 414

set_targets() (virttest.libvirt_xml.devices.character.CharacterXMLBase method), 123

set_termination_func() (virttest.aexpect.Tail method), 288

set_termination_params() (virttest.aexpect.Tail method), 288

set_timeout() (virttest.utils_libguestfs.LibguestfsBase method), 458

set_trace() (virttest.utils_libguestfs.GuestfishPersistent method), 451

set_transparent_hugepage() (in module virttest.staging.utils_memory), 257

set_uri() (virttest.virsh.VirshPersistent method), 510

set_uuid() (virttest.utils_libguestfs.GuestfishPersistent method), 451

set_validates() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176

set_verbose() (virttest.utils_libguestfs.GuestfishPersistent method), 451

set_vg() (virttest.lvm.PhysicalVolume method), 337

set_virsh() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176

set_vlanmode() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 344

set_vlanmode() (virttest.openvswitch.OpenVSwitchControlCli_140 method), 345

set_vm_disk() (in module virttest.utils_test.libvirt), 267

set_vm_vcpus() (virttest.libvirt_xml.vm_xml.VMXML static method), 207

set_win_guest_nic_status() (in module virttest.utils_net), 491

set_xml() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176

set_xmltreefile() (virttest.libvirt_xml.base.LibvirtXMLBase method), 176

setbasecmd() (virttest.remote_commander.remote_master.CmdMaster method), 228

setdefault() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232

setdefault() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236

setdefault() (virttest.staging.backports.simplejson.OrderedDict method), 242

setenforce() (virttest.libvirt_vm.VM method), 330

setenv() (virttest.utils_libguestfs.GuestfishPersistent method), 451

setKey() (virttest.RFBDes.Des method), 280

setmaxmem() (in module virttest.virsh), 543

setmem() (in module virttest.virsh), 543

setstderr() (virttest.remote_commander.remote_master.CmdMaster method), 228

setstdout() (virttest.remote_commander.remote_master.CmdMaster method), 228

setup() (in module virttest.bootstrap), 294

Setup() (virttest.installer_unittest.installer_test method), 318

setUp() (virttest.iscsi_unittest.iscsi_test method), 320

setUp() (virttest.libvirt_network_unittest.NetworkTestBase method), 320

setUp() (virttest.libvirt_storage_unittest.PoolTestBase method), 324

setUp() (virttest.libvirt_xml_unittest.LibvirtXMLTestBase method), 332

setUp() (virttest.libvirt_xml_unittest.testStubXML method), 334

setup() (virttest.lvm.EmulatedLVM method), 335

setup() (virttest.lvm.LVM method), 335

setup() (virttest.nfs.Nfs method), 343

setup() (virttest.nfs.NFSCClient method), 342

setUp() (virttest.nfs_unittest.nfs_test method), 343

setUp() (virttest.propcan_unittest.TestPropCan method), 358

setUp() (virttest.qemu_devices_unittest.Container method), 359

setUp() (virttest.qemu_qtree_unittest.QtreeDiskContainerTest method), 376

setup() (virttest.qemu_storage.Iscsidev method), 377

setup() (virttest.qemu_storage.LVMdev method), 377

setUp() (virttest.service_unittest.SystemdGeneratorTest method), 400	setup_bg_program() (virttest.utils_test.qemu.GuestSuspend method), 269
setUp() (virttest.service_unittest.SysVInitGeneratorTest method), 400	setup_boot_disk() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
setUp() (virttest.service_unittest.TestSystemdServiceManager method), 401	setup_cdrom() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
setUp() (virttest.service_unittest.TestSysVInitServiceManager method), 401	setup_import() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
setup() (virttest.test_setup.EGDCConfig method), 407	setup_local() (virttest.utils_misc.SELinuxBoolean method), 466
setup() (virttest.test_setup.HugePageConfig method), 408	setup_lv() (virttest.lvm.LVM method), 335
setup() (virttest.test_setup.KSMConfig method), 408	setup_nfs() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
setup() (virttest.test_setup.LibvirtPolkitConfig method), 409	setup_or_cleanup_gluster() (in module virttest.utils_test.libvirt), 267
setup() (virttest.test_setup.PrivateBridgeConfig method), 411	setup_or_cleanup_iscsi() (in module virttest.utils_test.libvirt), 267
setup() (virttest.test_setup.TransparentHugePageConfig method), 412	setup_or_cleanup_nfs() (in module virttest.utils_test.libvirt), 268
setup() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258	setup_pv() (virttest.lvm.EmulatedLVM method), 335
setUp() (virttest.utils_env_unittest.TestEnv method), 424	setup_pv() (virttest.lvm.LVM method), 335
setup() (virttest.utils_misc.SELinuxBoolean method), 466	setup_remote() (virttest.nfs.NFSClient method), 342
setUp() (virttest.utils_misc_unittest.TestNumaNode method), 479	setup_remote() (virttest.utils_misc.SELinuxBoolean method), 466
setUp() (virttest.utils_net.IPv6Manager method), 482	setup_stubs_add_chap_account() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestBridge method), 492	setup_stubs_cleanup() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestLibvirtIfc method), 492	setup_stubs_cleanup() (virttest.nfs_unittest.nfs_test method), 343
setUp() (virttest.utils_net_unittest.TestQemuIfc method), 492	setup_stubs_delete_chap_account() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestVirtIfc method), 493	setup_stubs_export_target() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestVmNet method), 493	setup_stubs_get_chap_accounts() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestVmNetStyle method), 493	setup_stubs_get_device_name() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494	setup_stubs_get_target_account_info() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_params_unittest.TestParams method), 496	setup_stubs_get_target_id() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.utils_sasl.SASL method), 497	setup_stubs_init() (virttest.iscsi_unittest.iscsi_test method), 320
setUp() (virttest.versionable_class_unittest.TestVersionableClass method), 507	setup_stubs_init() (virttest.nfs_unittest.nfs_test method), 343
setUp() (virttest.virsh_unittest.VirshClassHasHelpCommandTest method), 552	setup_stubs_is_mounted() (virttest.nfs_unittest.nfs_test method), 343
setUp() (virttest.virsh_unittest.VirshHasHelpCommandTest method), 552	
setUp() (virttest.virsh_unittest.VirshPersistentClassHasHelpCommandTest method), 553	
setUp() (virttest.xml_utils_unittest.test_templatized_xml method), 565	
setUp() (virttest.xml_utils_unittest.xml_test_data method), 565	

- method), 343
- setup_stubs_logged_in() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_login() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_portal_visible() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_set_chap_auth_initiator() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_set_chap_auth_target() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_set_initiatorName() (virttest.iscsi_unittest.iscsi_test method), 320
- setup_stubs_setup() (virttest.nfs_unittest.nfs_test method), 343
- setup_unattended_http_server() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
- setup_url() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
- setup_url_auto() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
- setup_vg() (virttest.lvm.LVM method), 336
- setup_virtio_win2003() (virttest.utils_disk.FloppyDisk method), 421
- setup_virtio_win2008() (virttest.utils_disk.CdromDisk method), 420
- setup_virtio_win2008() (virttest.utils_disk.FloppyDisk method), 421
- setup_win_driver_verifier() (in module virttest.utils_test.qemu), 274
- setvcpus() (in module virttest.virsh), 544
- sfdisk() (virttest.utils_libguestfs.GuestfishPersistent method), 451
- sfdisk_disk_geometry() (virttest.utils_libguestfs.GuestfishPersistent method), 451
- sfdisk_kernel_geometry() (virttest.utils_libguestfs.GuestfishPersistent method), 451
- sfdisk_l() (virttest.utils_libguestfs.GuestfishPersistent method), 452
- sfdisk_N() (virttest.utils_libguestfs.GuestfishPersistent method), 451
- sfdiskM() (virttest.utils_libguestfs.GuestfishPersistent method), 451
- sgio (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- sgio (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196
- sh() (virttest.utils_libguestfs.GuestfishPersistent method), 452
- sh_lines() (virttest.utils_libguestfs.GuestfishPersistent method), 452
- share (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- share (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196
- shares (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 198
- SHELL (virttest.utils_conn.SSHConnection attribute), 417
- shell() (virttest.remote_commander.remote_runner.CommanderSlaveCmds method), 230
- ShellCmdError, 283
- ShellError, 283
- ShellProcessTerminatedError, 283
- ShellSession (class in virttest.aexpect), 283
- ShellStatusError, 286
- ShellTimeoutError, 286
- shortname (virttest.cartesian_config.LUpdateFileMap attribute), 304
- show_config() (virttest.utils_misc.NumaNode method), 466
- shutdown() (in module virttest.virsh), 544
- shutdown() (virttest.guest_agent.QemuAgent method), 315
- shutdown() (virttest.libvirt_vm.VM method), 330
- shutdown() (virttest.ovirt.VMManager method), 349
- shutdown() (virttest.utils_libguestfs.GuestfishPersistent method), 452
- SHUTDOWN_MODE_HALT (virttest.guest_agent.QemuAgent attribute), 314
- SHUTDOWN_MODE_POWERDOWN (virttest.guest_agent.QemuAgent attribute), 314
- SHUTDOWN_MODE_REBOOT (virttest.guest_agent.QemuAgent attribute), 314
- sibling (virttest.libvirt_xml.capability_xml.CellXML attribute), 179
- signal_program() (in module virttest.utils_misc), 477
- simple_hotplug() (virttest.qemu_devices.qcontainer.DevContainer method), 218
- simple_unplug() (virttest.qemu_devices.qcontainer.DevContainer method), 219
- single_cmd_id (virttest.remote_commander.remote_interface.BaseCmd attribute), 226
- size (virttest.libvirt_xml.devices.memory.Memory.Target attribute), 135
- size (virttest.libvirt_xml.vm_xml.VMHugepagesXML.PageXML attribute), 200
- size (virttest.libvirt_xml.devices.memory.Memory.Target attribute), 135
- sleep() (virttest.utils_libguestfs.GuestfishPersistent method), 452

- slot (virttest.libvirt_xml.devices.hostdev.Hostdev.SourceAddress.UntypedAttribute), 130
- slot (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
- slot (virttest.libvirt_xml.nodedev_xml.PCIXML.Address attribute), 187
- SlotsCheckTest (class in virttest.utils_libguestfs_unittest), 462
- Smartcard (class in virttest.libvirt_xml.devices.smartcard), 137
- smartcard_mode (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- smartcard_type (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- smbios_mode (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202
- smoke_test() (virttest.staging.utils_cgroup.Cgroup method), 250
- snap_name (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase attribute), 197
- snapshot (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- snapshot (virttest.libvirt_xml.snapshot_xml.SnapshotXML attribute), 196
- snapshot() (virttest.ovirt.VMManager method), 349
- snapshot_apply() (virttest.qemu_storage.QemuImg method), 379
- snapshot_create() (in module virttest.virsh), 544
- snapshot_create() (virttest.libvirt_storage.QemuImg method), 322
- snapshot_create() (virttest.qemu_storage.QemuImg method), 379
- snapshot_create_as() (in module virttest.virsh), 544
- snapshot_current() (in module virttest.virsh), 544
- snapshot_del() (virttest.libvirt_storage.QemuImg method), 322
- snapshot_del() (virttest.qemu_storage.QemuImg method), 379
- snapshot_delete() (in module virttest.virsh), 545
- snapshot_dumpxml() (in module virttest.virsh), 545
- snapshot_edit() (in module virttest.virsh), 545
- snapshot_info() (in module virttest.virsh), 545
- snapshot_list() (in module virttest.virsh), 545
- snapshot_list() (virttest.qemu_storage.QemuImg method), 379
- snapshot_name (virttest.libvirt_xml.devices.disk.Disk.DiskSource attribute), 126
- snapshot_parent() (in module virttest.virsh), 546
- snapshot_revert() (in module virttest.virsh), 546
- SnapshotXML (class in virttest.libvirt_xml.snapshot_xml), 195
- SnapshotXML.SnapDiskXML (class in virttest.libvirt_xml.snapshot_xml), 195
- SnapshotXMLBase (class in virttest.libvirt_xml.snapshot_xml), 196
- soft_limit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- soft_limit_unit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
- somefunc() (virttest.virsh_unittest.TestVirshClosure static method), 552
- somemethod() (virttest.virsh_unittest.TestVirshClosure.SomeClass method), 552
- sort_fds_event() (in module virttest.remote_commander.remote_runner), 231
- Sound (class in virttest.libvirt_xml.devices.sound), 138
- source (virttest.libvirt_xml.devices.channel.Channel attribute), 123
- source (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- source (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- source (virttest.libvirt_xml.devices.memory.Memory attribute), 135
- source (virttest.libvirt_xml.devices.rng.Rng.Backend attribute), 136
- source (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- source (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
- source (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196
- source_address (virttest.libvirt_xml.devices.hostdev.Hostdev attribute), 130
- source_host (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- source_mode (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- source_service (virttest.libvirt_xml.devices.smartcard.Smartcard attribute), 137
- sourcebackupfile (virttest.xml_utils.XMLTreeFile attribute), 564
- SourceBuildFailed, 298
- SourceBuildParallelFailed, 298
- sourcefilename (virttest.xml_utils.XMLBackup attribute), 563
- sources (virttest.libvirt_xml.devices.character.CharacterBase attribute), 123
- sources (virttest.libvirt_xml.devices.console.Console attribute), 124
- sources (virttest.libvirt_xml.devices.serial.Serial attribute), 137
- SourceXML (class in virttest.libvirt_xml.pool_xml), 193
- sparse() (virttest.utils_libguestfs.GuestfishPersistent method), 452
- sparsify_disk() (virttest.utils_test.libguestfs.VirtTools method), 260

attribute), 152
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr
 attribute), 153
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.mac.Mac.Attr
 attribute), 154
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.rarp.Rarp.Attr
 attribute), 156
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr
 attribute), 160
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr
 attribute), 166
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr
 attribute), 167
 srcmacmask (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan.Attr
 attribute), 168
 srcportend (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr
 attribute), 152
 srcportend (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr
 attribute), 153
 srcportend (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr
 attribute), 157
 srcportend (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr
 attribute), 158
 srcportend (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr
 attribute), 161
 srcportend (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr
 attribute), 162
 srcportend (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr
 attribute), 164
 srcportend (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr
 attribute), 165
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.ip.Ip.Attr
 attribute), 152
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.ipv6.Ipv6.Attr
 attribute), 154
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr
 attribute), 157
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr
 attribute), 158
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr
 attribute), 161
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr
 attribute), 162
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr
 attribute), 164
 srcportstart (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr
 attribute), 165
 srv (virttest.libvirt_xml.network_xml.DNSXML attribute), 180
 SSH (virttest.utils_conn.SSHConnection attribute), 417
 SSH_ADD (virttest.utils_conn.SSHConnection attribute), 417
 SSH_AGENT (virttest.utils_conn.SSHConnection attribute), 417
 SSH_COPY_ID (virttest.utils_conn.SSHConnection attribute), 417
 ssh_id_rsa_path (virttest.utils_conn.SSHConnection attribute), 417
 SSH_KEYGEN (virttest.utils_conn.SSHConnection attribute), 417
 ssh_remote_auth (virttest.virsh.VirshPersistent attribute), 510
 ssh_rsa_pub_path (virttest.utils_conn.SSHConnection attribute), 417
 SSHCheckError, 417
 SSHCompletionError (in module virttest.utils_conn), 417
 SSHRmAuthKeysError, 417
 start() (in module virttest.virsh), 546
 start() (virttest.element_tree.TreeBuilder method), 310
 start() (virttest.libvirt_vm.VM method), 330
 start() (virttest.libvirt_xml.network_xml.NetworkXML method), 182
 start() (virttest.openvswitch.ServiceManagerInterface method), 346
 start() (virttest.openvswitch.ServiceManagerSystemD method), 346
 start() (virttest.openvswitch.ServiceManagerSysvinit method), 346
 start() (virttest.ovirt.VMManager method), 349
 start() (virttest.utils_libvirtd.Libvirtd method), 462
 start() (virttest.utils_libvirtd.LibvirtdSession method), 463
 start() (virttest.utils_netperf.NetperfClient method), 495
 start() (virttest.utils_netperf.NetperfServer method), 495
 start() (virttest.utils_test.BackgroundTest method), 275
 start() (virttest.video_maker.GstPythonVideoMaker method), 508
 start_auto_content_server_thread() (in module virttest.tests.unattended_install), 258
 start_file_logging() (virttest.standalone_test.Test method), 401
 start_ipm6 (virttest.utils_misc.KSMController method), 464
 start_ksmtuned() (virttest.utils_misc.KSMController method), 464
 start_ovs_vswitchd() (virttest.openvswitch.OpenVSwitch method), 344
 start_pool() (virttest.libvirt_storage.StoragePool method), 323
 start_suspend() (virttest.utils_test.qemu.GuestSuspend method), 269
 start_syslog_server_thread() (in module virttest.tests.unattended_install), 258
 start_tcpdump() (virttest.utils_env.Env method), 423
 start_unattended_server_thread() (in module virttest.tests.unattended_install), 258
 start_vdagent() (in module virttest.utils_spice), 501
 start_windows_service() (in module virttest.utils_test),

[278](#)
 startup() (virttest.test_setup.EGDCConfig method), [407](#)
 stat() (virttest.utils_libguestfs.GuestfishPersistent
method), [452](#)
 state (virttest.libvirt_xml.nwfilter_protocols.ah.Ah.Attr
attribute), [139](#)
 state (virttest.libvirt_xml.nwfilter_protocols.ah_ipv6.Ah_ipv6.Attr
attribute), [141](#)
 state (virttest.libvirt_xml.nwfilter_protocols.all.All.Attr
attribute), [142](#)
 state (virttest.libvirt_xml.nwfilter_protocols.all_ipv6.All_ipv6.Attr
attribute), [143](#)
 state (virttest.libvirt_xml.nwfilter_protocols.esp.Esp.Attr
attribute), [146](#)
 state (virttest.libvirt_xml.nwfilter_protocols.esp_ipv6.Esp_ipv6.Attr
attribute), [147](#)
 state (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr
attribute), [149](#)
 state (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr
attribute), [150](#)
 state (virttest.libvirt_xml.nwfilter_protocols.igmp.Igmp.Attr
attribute), [151](#)
 state (virttest.libvirt_xml.nwfilter_protocols.sctp.Sctp.Attr
attribute), [157](#)
 state (virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6.Sctp_ipv6.Attr
attribute), [158](#)
 state (virttest.libvirt_xml.nwfilter_protocols.tcp.Tcp.Attr
attribute), [161](#)
 state (virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6.Tcp_ipv6.Attr
attribute), [162](#)
 state (virttest.libvirt_xml.nwfilter_protocols.udp.Udp.Attr
attribute), [164](#)
 state (virttest.libvirt_xml.nwfilter_protocols.udp_ipv6.Udp_ipv6.Attr
attribute), [165](#)
 state (virttest.libvirt_xml.nwfilter_protocols.udplite.Udplite.Attr
attribute), [166](#)
 state (virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6.Udplite_ipv6.Attr
attribute), [167](#)
 state (virttest.libvirt_xml.snapshot_xml.SnapshotXMLBase
attribute), [197](#)
 state() (virttest.libvirt_vm.VM method), [331](#)
 state() (virttest.ovirt.HostManager method), [347](#)
 state() (virttest.ovirt.VMManager method), [349](#)
 state_dict() (virttest.libvirt_xml.network_xml.NetworkXML
method), [182](#)
 stats_period (virttest.libvirt_xml.devices.memballoon.Memballoon
attribute), [134](#)
 status() (virttest.nfs_unitest.FakeService method), [343](#)
 status() (virttest.openvswitch.OpenVSwitchControl
method), [344](#)
 status() (virttest.openvswitch.OpenVSwitchControlCli_140
method), [345](#)
 status() (virttest.openvswitch.ServiceManagerInterface
method), [346](#)
 status() (virttest.openswitch.ServiceManagerInterface
method), [346](#)
 statvfs() (virttest.utils_libguestfs.GuestfishPersistent
method), [452](#)
 StdErr (class in virttest.remote_commander.remote_interface),
[227](#)
 CmdMaster (class in virttest.remote_commander.remote_master.CmdMaster
attribute), [228](#)
 StdIOWrapper (class in
virttest.remote_commander.messenger), [225](#)
 StdIOWrapperIn (class in
virttest.remote_commander.messenger), [225](#)
 StdIOWrapperInBase64 (class in
virttest.remote_commander.messenger), [226](#)
 StdIOWrapperOut (class in
virttest.remote_commander.messenger), [226](#)
 StdIOWrapperOutBase64 (class in
virttest.remote_commander.messenger), [226](#)
 StdOut (class in virttest.remote_commander.remote_interface),
[227](#)
 stdout (virttest.remote_commander.remote_master.CmdMaster
attribute), [228](#)
 StdStream (class in virttest.remote_commander.remote_interface),
[227](#)
 stop() (virttest.lvsb_base.SandboxBase method), [339](#)
 stop() (virttest.openvswitch.ServiceManagerInterface
method), [346](#)
 stop() (virttest.openvswitch.ServiceManagerSystemD
method), [346](#)
 stop() (virttest.openvswitch.ServiceManagerSysvinit
method), [346](#)
 stop() (virttest.utils_gdb.GDB method), [426](#)
 stop() (virttest.utils_libvirtd.Libvirtd method), [462](#)
 stop() (virttest.utils_netperf.Netperf method), [494](#)
 stop() (virttest.utils_netperf.NetperfClient method), [495](#)
 stop() (virttest.utils_netperf.NetperfServer method), [496](#)
 stop_file_logging() (virttest.standalone_test.Test
method), [401](#)
 stop_ksm() (virttest.utils_misc.KSMController method),
[464](#)
 stop_ksmtuned() (virttest.utils_misc.KSMController
method), [464](#)
 stop_tcpdump() (virttest.utils_env.Env method), [423](#)
 stop_vdagent() (in module virttest.utils_spice), [501](#)
 stop_windows_service() (in module virttest.utils_test),
[278](#)
 storage_cleanup() (virttest.utils_v2v.VMCheck method),
[503](#)
 StorageDomainManager (class in virttest.ovirt), [347](#)
 StoragePool (class in virttest.libvirt_storage), [322](#)
 StorageXML (class in virttest.libvirt_xml.nodedev_xml),
[187](#)
 store_vm_register() (in module virttest.env_process), [312](#)
 Stp (class in virttest.libvirt_xml.nwfilter_protocols.stp),
[187](#)

158
 Stp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.stp), 349
 159
 str_bus_long() (virttest.qemu_devices.qcontainer.DevContainer method), 219
 str_bus_short() (virttest.qemu_devices.qcontainer.DevContainer method), 219
 str_long() (virttest.qemu_devices.qbuses.QSparseBus method), 215
 str_long() (virttest.qemu_devices.qcontainer.DevContainer method), 219
 str_long() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
 str_qtree() (virttest.qemu_qtree.QtreeNode method), 375
 str_short() (virttest.qemu_devices.qbuses.QSparseBus method), 215
 str_short() (virttest.qemu_devices.qcontainer.DevContainer method), 219
 str_short() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
 str_short() (virttest.qemu_qtree.QtreeNode method), 375
 StressError, 275
 string_to_bitlist() (in module virttest.utils_misc), 477
 strings() (virttest.utils_libguestfs.GuestfishPersistent method), 452
 strip_console_codes() (in module virttest.utils_misc), 477
 StrReader (class in virttest.cartesian_config), 306
 StubDeviceMeta (class in virttest.libvirt_xml.devices.base), 122
 Sub (class in virttest.xml_utils), 562
 sub() (virttest.remote.RemoteFile method), 391
 sub_else_add() (virttest.remote.RemoteFile method), 391
 subclass (virttest.libvirt_xml.accessors.XMLElementNest attribute), 174
 subclass (virttest.libvirt_xml.accessors.XMLElementNest attribute), 174
 subclass_dargs (virttest.libvirt_xml.accessors.XMLElementNest attribute), 174
 subclass_pre_init() (virttest.utils_net.VMNet method), 484
 SubdirGlobList (class in virttest.data_dir), 308
 SubdirList (class in virttest.data_dir), 308
 SubElement() (in module virttest.element_tree), 310
 substitute() (virttest.xml_utils.Sub method), 562
 summary_up_result() (in module virttest.utils_test), 279
 support_cmd() (virttest.qemu_storage.QemuImg method), 379
 supported() (virttest.utils_libguestfs.GuestfishPersistent method), 452
 SUPPORTED_SERIAL_TYPE (virttest.guest_agent.QemuAgent attribute), 314
 suspend() (in module virttest.virsh), 546
 suspend() (virttest.guest_agent.QemuAgent method), 315
 suspend() (virttest.ovirt.VMManager method), 349
 SUSPEND_MODE_DISK (virttest.guest_agent.QemuAgent attribute), 314
 SUSPEND_MODE_HYBRID (virttest.guest_agent.QemuAgent attribute), 314
 SUSPEND_MODE_RAM (virttest.guest_agent.QemuAgent attribute), 314
 SUSPEND_TYPE_DISK (virttest.utils_test.qemu.GuestSuspend attribute), 269
 SUSPEND_TYPE_MEM (virttest.utils_test.qemu.GuestSuspend attribute), 269
 swap_hard_limit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
 swap_limit_unit (virttest.libvirt_xml.vm_xml.VMMemTuneXML attribute), 201
 swapoff_device() (virttest.utils_libguestfs.GuestfishPersistent method), 452
 swapoff_file() (virttest.utils_libguestfs.GuestfishPersistent method), 452
 swapoff_label() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 swapoff_uuid() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 swapon_device() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 swapon_file() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 swapon_label() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 swapon_uuid() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 sync() (virttest.guest_agent.QemuAgent method), 315
 sync() (virttest.libvirt_xml.network_xml.NetworkXML method), 182
 sync() (virttest.libvirt_xml.vm_xml.VMXML method), 207
 sync() (virttest.utils_libguestfs.GuestfishPersistent method), 453
 sync_directories() (virttest.remote_build.Builder method), 397
 Sys (class in virttest.versionable_class_unittest), 507
 Sys1 (class in virttest.versionable_class_unittest), 507
 Sys_Container (class in virttest.versionable_class_unittest), 507
 sysfs_main_path (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 186
 sysinfo() (in module virttest.virsh), 546
 SysinfoXML (class in virttest.libvirt_xml.sysinfo_xml), 197

- [syslinux\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) method), [453](#)
[syslog_server\(\)](#) (in module [virttest.syslog_server](#)), [407](#)
[SysLogServerTcp](#) (class in [virttest.syslog_server](#)), [407](#)
[SysLogServerUdp](#) (class in [virttest.syslog_server](#)), [407](#)
[System](#) (class in [virttest.versionable_class_unittest](#)), [507](#)
[System1](#) (class in [virttest.versionable_class_unittest](#)), [507](#)
[System_Container](#) (class in [virttest.versionable_class_unittest](#)), [507](#)
[system_version\(\)](#) (in module [virttest.versionable_class_unittest](#)), [508](#)
[system_wakeup\(\)](#) ([virttest.qemu_monitor.HumanMonitor](#) method), [365](#)
[system_wakeup\(\)](#) ([virttest.qemu_monitor.QMPMonitor](#) method), [372](#)
[systemd_command_generator\(\)](#) (in module [virttest.staging.service](#)), [246](#)
[systemd_list_parser\(\)](#) (in module [virttest.staging.service](#)), [246](#)
[systemd_result_parser\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[systemd_status_parser\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[SystemdGeneratorTest](#) (class in [virttest.service_unittest](#)), [400](#)
[SystemXML](#) (class in [virttest.libvirt_xml.nodedev_xml](#)), [188](#)
[sysvinit_command_generator\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[sysvinit_list_parser\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[sysvinit_result_parser\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[sysvinit_status_parser\(\)](#) (in module [virttest.staging.service](#)), [247](#)
[SysVInitGeneratorTest](#) (class in [virttest.service_unittest](#)), [400](#)
- ## T
- [tag_name](#) ([virttest.libvirt_xml.accessors.XMLAttribute.Del](#) attribute), [170](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLAttribute.Get](#) attribute), [170](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLAttribute.Set](#) attribute), [170](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementBool.Del](#) attribute), [171](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementBool.Get](#) attribute), [171](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementBool.Set](#) attribute), [171](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementDict.Del](#) attribute), [172](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementDict.Get](#) attribute), [172](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementDict.Set](#) attribute), [172](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementInt.Del](#) attribute), [172](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementInt.Get](#) attribute), [173](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementInt.Set](#) attribute), [173](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementNest.Del](#) attribute), [174](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementNest.Get](#) attribute), [174](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementNest.Set](#) attribute), [174](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementText.Del](#) attribute), [175](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementText.Get](#) attribute), [175](#)
[tag_name](#) ([virttest.libvirt_xml.accessors.XMLElementText.Set](#) attribute), [175](#)
[Tail](#) (class in [virttest.aexpect](#)), [287](#)
[tail\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) method), [453](#)
[TAPBringDownError](#), [484](#)
[TAPBringUpError](#), [484](#)
[TAPCreationError](#), [484](#)
[tapfd_ids](#) ([virttest.utils_net.QemuIface](#) attribute), [483](#)
[tapfds](#) ([virttest.utils_net.QemuIface](#) attribute), [483](#)
[TAPModuleError](#), [484](#)
[TAPNotExistError](#), [484](#)
[tar_in\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) method), [453](#)
[tar_in\(\)](#) ([virttest.utils_test.libguestfs.VirtTools](#) method), [260](#)
[tar_in_opts\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) method), [453](#)
[tar_out\(\)](#) ([virttest.utils_libguestfs.GuestfishPersistent](#) method), [453](#)
[tar_out\(\)](#) ([virttest.utils_test.libguestfs.VirtTools](#) method), [260](#)
[Target](#) (class in [virttest.utils_v2v](#)), [503](#)
[target](#) ([virttest.libvirt_xml.devices.channel.Channel](#) attribute), [123](#)
[target](#) ([virttest.libvirt_xml.devices.disk.Disk](#) attribute), [128](#)
[target](#) ([virttest.libvirt_xml.devices.interface.Interface](#) attribute), [133](#)
[target](#) ([virttest.libvirt_xml.devices.memory.Memory](#) attribute), [135](#)
[target](#) ([virttest.libvirt_xml.secret_xml.SecretXMLBase](#) attribute), [195](#)
[target](#) ([virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML](#) attribute), [195](#)

attribute), 196

target_path (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193

target_port (virttest.libvirt_xml.devices.console.Console attribute), 124

target_port (virttest.libvirt_xml.devices.serial.Serial attribute), 137

target_type (virttest.libvirt_xml.devices.console.Console attribute), 124

target_type (virttest.libvirt_xml.devices.serial.Serial attribute), 137

targets (virttest.libvirt_xml.devices.character.CharacterBase attribute), 123

Tcp (class in virttest.libvirt_xml.nwfilter_protocols.tcp), 160

Tcp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.tcp), 160

Tcp_ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6), 161

Tcp_ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6), 162

tcp_port (virttest.utils_conn.TCPConnection attribute), 418

TCPConnection (class in virttest.utils_conn), 417

tearDown() (virttest.iscsi_unittest.iscsi_test method), 320

tearDown() (virttest.libvirt_storage_unittest.NewPoolTest method), 323

tearDown() (virttest.libvirt_xml_unittest.LibvirtXMLTestBase method), 332

tearDown() (virttest.nfs_unittest.nfs_test method), 343

tearDown() (virttest.qemu_devices_unittest.Container method), 359

tearDown() (virttest.qemu_qtree_unittest.QtreeDiskContainer method), 376

tearDown() (virttest.utils_env_unittest.TestEnv method), 424

tearDown() (virttest.utils_misc_unittest.TestNumaNode method), 479

tearDown() (virttest.utils_net_unittest.TestBridge method), 492

tearDown() (virttest.versionable_class_unittest.TestVersionableClass method), 507

tearDown() (virttest.virsh_unittest.VirshPersistentClassHasHelpCommandTest method), 553

tearDown() (virttest.xml_utils_unittest.xml_test_data method), 565

TemplateXML (class in virttest.xml_utils), 562

TemplateXMLTreeBuilder (class in virttest.xml_utils), 563

TempXMLFile (class in virttest.xml_utils), 562

terminate_auto_content_server_thread() (in module virttest.tests.unattended_install), 258

terminate_syslog_server_thread() (in module virttest.tests.unattended_install), 258

terminate_unattended_server_thread() (in module virttest.tests.unattended_install), 258

Test (class in virttest.standalone_test), 401

test() (virttest.staging.utils_cgroup.Cgroup method), 250

test_01_Params() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_02_db() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_03_db() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_04_VirtNet() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_05_VirtNet() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_06_VirtNet() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_07_VirtNet() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_08_ifname() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_99_ifname() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494

test_accessors() (virttest.utils_config_unittest.LibvirtConfigTest method), 415

test_accessors() (virttest.utils_config_unittest.SectionlessConfigTest method), 415

test_accessor_base() (virttest.libvirt_xml_unittest.AccessorsTest method), 332

test_all_command() (virttest.service_unittest.SystemdGeneratorTest method), 400

test_all_command() (virttest.service_unittest.SysVInitGeneratorTest method), 400

test_AllForbidden() (virttest.libvirt_xml_unittest.AccessorsTest method), 332

test_apendex_set() (virttest.utils_net_unittest.TestVirtIface method), 493

test_arbitrart_attributes() (virttest.libvirt_xml_unittest.testCharacterXML method), 333

test_args() (virttest.virsh_unittest.TestVirshClosure method), 552

test_args_and_dargs() (virttest.virsh_unittest.TestVirshClosure method), 552

test_args_dargs_subclass() (virttest.virsh_unittest.TestVirshClosure method), 552

test_backup_backup_and_remove() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565

test_backup_file() (virttest.xml_utils_unittest.test_XMLBackup method), 564

test_backup_filename() (virttest.xml_utils_unittest.test_XMLBackup method), 564

- test_bad_names() (virttest.libvirt_xml_unittest.testLibrarian method), 323
- test_bad_qtree() (virttest.qemu_qtree_unittest.QtreeContainerTest method), 358
- test_bitlist_to_string() (virttest.utils_misc_unittest.TestNumaNode method), 479
- test_bundled_elementtree() (virttest.xml_utils_unittest.test_ElementTree method), 564
- test_cache_command() (virttest.virsh_unittest.VirshHelpCommandTest method), 553
- test_capxmlbase() (virttest.libvirt_xml_unittest.testCAPXML method), 333
- test_channels() (virttest.libvirt_xml_unittest.testVMXMLDevices method), 334
- test_check_params() (virttest.qemu_qtree_unittest.QtreeDiskContainerTest method), 376
- test_check_params_bad() (virttest.qemu_qtree_unittest.QtreeDiskContainerTest method), 376
- test_class_bb (virttest.versionable_class_unittest.BB attribute), 506
- test_class_vml (virttest.versionable_class_unittest.VM attribute), 507
- test_cmp_Virtnet() (virttest.utils_net_unittest.TestVmNetSubclass method), 494
- test_compare() (virttest.propcan_unittest.TestPropCan method), 358
- test_complicated_multiple_create_params() (virttest.versionable_class_unittest.TestVersionableClass method), 507
- test_complicated_versioning() (virttest.versionable_class_unittest.TestVersionableClass method), 507
- test_cpu_vendor_amd() (virttest.utils_misc_unittest.TestUtilsMisc method), 479
- test_cpu_vendor_intel() (virttest.utils_misc_unittest.TestUtilsMisc method), 479
- test_create_by_xpath() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
- test_create_by_xpath() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565
- test_dargs() (virttest.virsh_unittest.TestVirshClosure method), 552
- test_default_default() (virttest.utils_net_unittest.TestVmNetStyle method), 493
- test_del_VirshPersistent() (virttest.virsh_unittest.SessionManagerTest method), 552
- test_dict_methods_1() (virttest.propcan_unittest.TestPropCanBase method), 358
- test_dict_methods_2() (virttest.propcan_unittest.TestPropCanBase method), 358
- test_dir_pool() (virttest.libvirt_storage_unittest.NewPoolTest method), 323
- test_double_init() (virttest.propcan_unittest.TestPropCanBase method), 358
- test_ElementTree (class in virttest.xml_utils_unittest), 564
- test_empty_init() (virttest.propcan_unittest.TestPropCanBase method), 358
- test_empty_params_init() (virttest.propcan_unittest.TestPropCanBase method), 358
- test_enable() (virttest.service_unittest.TestSysVInitServiceManager method), 401
- test_exist_pool() (virttest.libvirt_storage_unittest.ExistPoolTest method), 323
- test_extraneous_init() (virttest.propcan_unittest.TestPropCan method), 358
- test_fake_virsh() (virttest.virsh_unittest.TestVirshClosure method), 552
- test_false_command() (virttest.virsh_unittest.VirshHasHelpCommandTest method), 553
- test_file_path (virttest.remote_unittest.RemoteFileTest attribute), 398
- test_free_cpu() (virttest.utils_misc_unittest.TestNumaNode method), 479
- test_from_element() (virttest.libvirt_xml_unittest.testSerialXML method), 334
- test_full_set() (virttest.utils_net_unittest.TestVirtIface method), 493
- test_get_all_vms() (virttest.utils_env_unittest.TestEnv method), 424
- test_get_archive_tarball_name() (virttest.utils_misc_unittest.TestUtilsMisc method), 479
- test_get_archive_tarball_name_absolute() (virttest.utils_misc_unittest.TestUtilsMisc method), 479
- test_get_archive_tarball_name_from_dir() (virttest.utils_misc_unittest.TestUtilsMisc method), 479
- test_get_cgroup_mountpoint() (virttest.utils_cgroup_unittest.CgroupTest method), 412
- test_get_key2filename_dict() (virttest.libvirt_xml_unittest.testPCIXML method), 333
- test_get_key2syspath_dict() (virttest.libvirt_xml_unittest.testNodedevXML method), 333
- test_get_key2value_dict() (virttest.libvirt_xml_unittest.testNodedevXML method), 333
- test_get_key2value_dict() (virttest.libvirt_xml_unittest.testPCIXML method), 333

[test_get_node_cpus\(\)](#) (virttest.utils_misc_unittest.TestNumaNode method), [462](#)
[test_get_path\(\)](#) (virttest.libvirt_xml_unittest.testPCIXML method), [333](#)
[test_get_set_del\(\)](#) (virttest.libvirt_xml_unittest.testVMCPUXML method), [334](#)
[test_get_xpath\(\)](#) (virttest.xml_utils_unittest.test_XMLTreeFile method), [565](#)
[test_getstructure\(\)](#) (virttest.utils_net_unittest.TestBridge method), [492](#)
[test_getter\(\)](#) (virttest.libvirt_xml_unittest.testNodedevXMLBase method), [333](#)
[test_getters\(\)](#) (virttest.libvirt_xml_unittest.testNetworkXML method), [333](#)
[test_getters\(\)](#) (virttest.libvirt_xml_unittest.testSerialXML method), [334](#)
[test_getters\(\)](#) (virttest.libvirt_xml_unittest.TestVMXML method), [332](#)
[test_git_repo_param_helper\(\)](#) (virttest.utils_misc_unittest.TestUtilsMisc method), [479](#)
[test_graphics\(\)](#) (virttest.libvirt_xml_unittest.testVMXMLDevices method), [334](#)
[test_groups_in_commands\(\)](#) (virttest.virsh_unittest.VirshHasHelpCommandTest method), [553](#)
[test_guest_capabilities\(\)](#) (virttest.libvirt_xml_unittest.TestLibvirtXML method), [332](#)
[test_Guestfish_slots\(\)](#) (virttest.utils_libguestfs_unittest.SlotsCheckTest method), [462](#)
[test_half_set\(\)](#) (virttest.utils_net_unittest.TestVirtIface method), [493](#)
[test_init\(\)](#) (virttest.virsh_unittest.TestVirshClosure method), [552](#)
[test_init_None_value\(\)](#) (virttest.propcan_unittest.TestPropCan method), [358](#)
[test_init_str\(\)](#) (virttest.xml_utils_unittest.test_XMLTreeFile method), [565](#)
[test_init_xml\(\)](#) (virttest.xml_utils_unittest.test_XMLTreeFile method), [565](#)
[test_ip_getter\(\)](#) (virttest.libvirt_xml_unittest.testNetworkXML method), [333](#)
[test_iscsi_get_device_name\(\)](#) (virttest.iscsi_unittest.iscsi_test method), [320](#)
[test_iscsi_login\(\)](#) (virttest.iscsi_unittest.iscsi_test method), [320](#)
[test_iscsi_target_id\(\)](#) (virttest.iscsi_unittest.iscsi_test method), [320](#)
[test_iscsi_visible\(\)](#) (virttest.iscsi_unittest.iscsi_test method), [320](#)
[test_lgf_cmd\(\)](#) (virttest.utils_libguestfs_unittest.LibguestfsTest method), [462](#)
[test_lgf_cmd_check\(\)](#) (virttest.utils_libguestfs_unittest.LibguestfsTest method), [332](#)
[test_lgf_cmd_check_raises\(\)](#) (virttest.utils_libguestfs_unittest.LibguestfsTest method), [462](#)
[test_LibguestfsBase_default_slots\(\)](#) (virttest.utils_libguestfs_unittest.SlotsCheckTest method), [462](#)
[test_LibguestfsBase_update_slots\(\)](#) (virttest.utils_libguestfs_unittest.SlotsCheckTest method), [462](#)
[test_libvirt\(\)](#) (virttest.utils_net_unittest.TestVmNetStyle method), [493](#)
[test_list\(\)](#) (virttest.service_unittest.SystemdGeneratorTest method), [400](#)
[test_list\(\)](#) (virttest.service_unittest.TestSystemdServiceManager method), [401](#)
[test_list\(\)](#) (virttest.service_unittest.TestSysVInitServiceManager method), [401](#)
[test_locking\(\)](#) (virttest.utils_env_unittest.TestEnv method), [424](#)
[test_mac_completer\(\)](#) (virttest.utils_net_unittest.TestVirtIface method), [493](#)
[test_make_installer\(\)](#) (virttest.installer_unittest.installer_test method), [318](#)
[test_MappingTreeBuilder_standalone\(\)](#) (virttest.xml_utils_unittest.test_templatized_xml method), [565](#)
[test_mixed_init\(\)](#) (virttest.propcan_unittest.TestPropCanBase method), [358](#)
[test_ModuleLoad\(\)](#) (virttest.service_unittest.ConstantsTest method), [400](#)
[test_ModuleLoad\(\)](#) (virttest.virsh_unittest.ConstantsTest method), [551](#)
[test_multi_inst\(\)](#) (virttest.virsh_unittest.TestVirshClosure method), [552](#)
[test_new_from_dumpxml\(\)](#) (virttest.libvirt_xml_unittest.testNodedevXML method), [333](#)
[test_new_from_dumpxml\(\)](#) (virttest.libvirt_xml_unittest.TestVMXML method), [332](#)
[test_nfs_setup\(\)](#) (virttest.nfs_unittest.nfs_test method), [343](#)
[test_no_cache\(\)](#) (virttest.virsh_unittest.VirshHasHelpCommandTest method), [553](#)
[test_no_module\(\)](#) (virttest.libvirt_xml_unittest.testLibrarian method), [333](#)
[test_no_path\(\)](#) (virttest.utils_config_unittest.LibvirtConfigCommonTest method), [414](#)
[test_normalize_data_size\(\)](#) (virttest.utils_misc_unittest.TestUtilsMisc method), [479](#)
[test_not_enuf_dargs\(\)](#) (virttest.libvirt_xml_unittest.AccessorsTest method), [332](#)

test_not_exist_pool() (virttest.libvirt_storage_unittest.NotExpectedPoolTest.method), 324	test_test_installer_unittest.installer_test (virttest.installer_unittest.installer_test.method), 318
test_odd_values() (virttest.propcan_unittest.TestPropCan.method), 358	test_register_syncserver() (virttest.utils_env_unittest.TestEnv.method), 425
test_pci() (virttest.qemu_devices_unittest.Container.method), 359	test_register_vm() (virttest.utils_env_unittest.TestEnv.method), 425
test_pickleing() (virttest.versionable_class_unittest.TestVersionableClass.method), 507	test_remove_backup_file() (virttest.xml_utils_unittest.test_XMLBackup.method), 564
test_pin_cpu() (virttest.utils_misc_unittest.TestNumaNode.method), 479	test_required_slots() (virttest.libvirt_xml_unittest.AccessorsTest.method), 332
test_prefix_suffix() (virttest.xml_utils_unittest.test_TempXMLFile.method), 564	test_restore() (virttest.libvirt_xml_unittest.TestVMXML.method), 332
test_printables() (virttest.propcan_unittest.TestPropCan.method), 358	test_restore() (virttest.utils_config_unittest.SectionlessConfigTest.method), 415
test_q_base_device() (virttest.qemu_devices_unittest.Devices.method), 359	test_restore_fails() (virttest.xml_utils_unittest.test_templatized_xml.method), 565
test_q_device() (virttest.qemu_devices_unittest.Devices.method), 359	test_restore_file() (virttest.xml_utils_unittest.test_XMLBackup.method), 564
test_q_pci_bus() (virttest.qemu_devices_unittest.Buses.method), 358	test_restore_from_file() (virttest.xml_utils_unittest.test_XMLTreeFile.method), 565
test_q_pci_bus_strict() (virttest.qemu_devices_unittest.Buses.method), 358	test_restore_from_string() (virttest.xml_utils_unittest.test_XMLTreeFile.method), 565
test_q_sparse_bus() (virttest.qemu_devices_unittest.Buses.method), 358	test_runlevels() (virttest.service_unittest.TestSysVInitServiceManager.method), 401
test_q_string_device() (virttest.qemu_devices_unittest.Devices.method), 359	test_save() (virttest.utils_env_unittest.TestEnv.method), 425
test_qdev_equal() (virttest.qemu_devices_unittest.Container.method), 359	test_seclabel() (virttest.libvirt_xml_unittest.TestVMXML.method), 332
test_qdev_functional() (virttest.qemu_devices_unittest.Container.method), 359	test_serial_class() (virttest.libvirt_xml_unittest.testLibrarian.method), 333
test_qdev_hotplug() (virttest.qemu_devices_unittest.Container.method), 359	test_set_target() (virttest.service_unittest.SystemdGeneratorTest.method), 401
test_qdev_low_level() (virttest.qemu_devices_unittest.Container.method), 359	test_set_target() (virttest.service_unittest.SysVInitGeneratorTest.method), 400
test_qtree() (virttest.qemu_qtree_unittest.QtreeContainerTest.method), 376	test_sharing_data_in_same_version() (virttest.versionable_class_unittest.TestVersionableClass.method), 507
test_qtree_bus_bus() (virttest.qemu_qtree_unittest.KvmQtreeClassTest.method), 376	test_simple_create_by_params_v0() (virttest.versionable_class_unittest.TestVersionableClass.method), 507
test_qtree_dev_dev() (virttest.qemu_qtree_unittest.KvmQtreeClassTest.method), 376	test_simple_create_by_params_v1() (virttest.versionable_class_unittest.TestVersionableClass.method), 507
test_qtree_disk_missing_filename() (virttest.qemu_qtree_unittest.KvmQtreeClassTest.method), 376	test_simple_help_creating() (virttest.versionable_class_unittest.TestVersionableClass.method), 507
test_read_other_changed() (virttest.xml_utils_unittest.test_XMLTreeFile.method), 565	test_single_init() (virttest.propcan_unittest.TestPropCanBase.method), 358
test_rebackup_file() (virttest.xml_utils_unittest.test_XMLBackup.method), 564	test_slots_restrict() (virttest.propcan_unittest.TestPropCanBase.method), 358
test_recycle_session() (virttest.virsh_unittest.VirshPersistentClassHelpCreatingTest.method), 553	
test_register_get_installer() (virttest.installer_unittest.installer_test.method), 318	
test_register_get_installer_default()	

test_sourcebackupfile_closed_file() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565	virttest.xml_utils_unittest), 565
test_sourcebackupfile_closed_string() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565	test_TempXMLBackup_exception_exit() (virttest.xml_utils_unittest.test_XMLBackup method), 564
test_specific_accessors() (virttest.utils_config_unittest.SectionlessConfigTest) method), 415	test_TempXMLBackup_implicit() (virttest.xml_utils_unittest.test_XMLBackup method), 564
test_start() (virttest.service_unittest.TestSystemdServiceManager method), 401	test_TempXMLBackup_unexception_exit() (virttest.xml_utils_unittest.test_XMLBackup method), 564
test_static() (virttest.libvirt_xml_unittest.testNodedevXMLBase_TempXMLFile (class in virttest.xml_utils_unittest), method), 333	test_TempXMLFile_explicit() (virttest.xml_utils_unittest.test_TempXMLFile method), 564
test_static() (virttest.libvirt_xml_unittest.testPCIXML method), 333	test_TempXMLFile_implicit() (virttest.xml_utils_unittest.test_TempXMLFile method), 564
test_string_container() (virttest.utils_net_unittest.TestVmNet method), 493	test_test_TempXMLFile_canread() (virttest.xml_utils_unittest.test_TempXMLFile method), 564
test_string_to_bitlist() (virttest.utils_misc_unittest.TestNumaNode method), 479	test_too_many_dargs() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
test_stringify() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565	test_true_command() (virttest.virsh_unittest.VirshHasHelpCommandTest method), 553
test_sub() (virttest.xml_utils_unittest.test_templatized_xml method), 565	test_type_check() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
test_subclass_no_mask_attributeerror() (virttest.propcan_unittest.TestPropCanBase method), 358	test_typed_device_stub() (virttest.libvirt_xml_unittest.testStubXML method), 334
test_subclass_single_init_delter() (virttest.propcan_unittest.TestPropCanBase method), 358	test_undefined_type() (virttest.utils_config_unittest.LibvirtConfigCommonT method), 414
test_subclass_single_init_getter() (virttest.propcan_unittest.TestPropCanBase method), 358	test_unimplemented() (virttest.utils_config_unittest.LibvirtConfigCommonT method), 414
test_subclass_single_init_setter() (virttest.propcan_unittest.TestPropCanBase method), 358	test_unknown_runlevel() (virttest.service_unittest.TestSysVInitServiceManager method), 401
test_subcommand_help() (virttest.virsh_unittest.VirshHasHelpCommandTest method), 553	test_unregister_syncserver() (virttest.utils_env_unittest.TestEnv method), 425
test_sync_and_state_dict() (virttest.libvirt_network_unittest.NetworkXMLTest method), 320	test_unregister_vm() (virttest.utils_env_unittest.TestEnv method), 425
test_sync_file() (virttest.utils_config_unittest.SectionlessConfigTest method), 415	test_untyped_device_stub() (virttest.libvirt_xml_unittest.testStubXML method), 334
test_systemd_result_parser() (virttest.service_unittest.ResultParserTest method), 400	test_update() (virttest.propcan_unittest.TestPropCanBase method), 358
test_sysvinit_result_parser() (virttest.service_unittest.ResultParserTest method), 400	test_update_args_dargs_subclass() (virttest.virsh_unittest.TestVirshClosure method), 552
test_TemplateXML() (virttest.xml_utils_unittest.test_templatized_xml method), 565	test_usb_bus() (virttest.qemu_devices_unittest.Buses method), 359
test_TemplateXMLTreeBuilder_nosub() (virttest.xml_utils_unittest.test_templatized_xml method), 565	test_uuid() (virttest.libvirt_xml_unittest.TestLibvirtXML method), 332
test_templatized_xml (class in	

test_valid_xml() (virttest.libvirt_xml_unittest.testNetworkXML method), 397
 method), 333
 test_valid_xml() (virttest.libvirt_xml_unittest.TestVMXML method), 332
 test_vendor_unknown() (virttest.utils_misc_unittest.TestUtilMisc method), 479
 test_Virsh() (virttest.virsh_unittest.ConstructorsTest method), 551
 test_VirshBase() (virttest.virsh_unittest.ConstructorsTest method), 551
 test_VirshPersistent() (virttest.virsh_unittest.ConstructorsTest method), 551
 test_VirshPersistent() (virttest.virsh_unittest.SessionManagerTest method), 552
 test_VirshSession() (virttest.virsh_unittest.SessionManagerTest method), 552
 test_VirtIface_container() (virttest.utils_net_unittest.TestVmNet method), 493
 test_vm_get() (virttest.libvirt_xml_unittest.testDiskXML method), 333
 test_vm_get_by_class() (virttest.libvirt_xml_unittest.testDiskXML method), 333
 test_vm_get_by_class() (virttest.libvirt_xml_unittest.testSerialXML method), 334
 test_vm_get_modify() (virttest.libvirt_xml_unittest.testSerialXML method), 334
 test_write_default() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565
 test_write_other() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565
 test_write_other_changed() (virttest.xml_utils_unittest.test_XMLTreeFile method), 565
 test_XMLBackup (class in virttest.xml_utils_unittest), 564
 test_XMLElementBool_deep() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLElementBool_simple() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLElementInt() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLElementList() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLElementList_Text() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLElementNest() (virttest.libvirt_xml_unittest.AccessorsTest method), 332
 test_XMLTreeFile (class in virttest.xml_utils_unittest), 564
 testAdd() (virttest.remote_unittest.RemoteFileTest method), 397
 testAddressXML (class in virttest.libvirt_xml_unittest), 332
 TestBaseSandboxes (class in virttest.lvsb), 338
 TestBridge (class in virttest.utils_net_unittest), 492
 TestBridge.FakeCmd (class in virttest.utils_net_unittest), 492
 testCAPXML (class in virttest.libvirt_xml_unittest), 333
 testCharacterXML (class in virttest.libvirt_xml_unittest), 333
 TestComplexSandboxes (class in virttest.lvsb), 338
 testComplicatedFilter() (virttest.cartesian_config_unittest.CartesianConfigTest method), 307
 testCondition() (virttest.cartesian_config_unittest.CartesianConfigTest method), 307
 testDefaults() (virttest.cartesian_config_unittest.CartesianConfigTest method), 307
 testDel() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testDiskXML (class in virttest.libvirt_xml_unittest), 333
 TestEnv (class in virttest.utils_env_unittest), 424
 testError() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testFilterMixing() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testItem() (virttest.utils_params_unittest.TestParams method), 496
 testFileItemMissing() (virttest.utils_params_unittest.TestParams method), 496
 testFileHugeTest1() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testLibrarian (class in virttest.libvirt_xml_unittest), 333
 TestLibvirtIface (class in virttest.utils_net_unittest), 492
 TestLibvirtXML (class in virttest.libvirt_xml_unittest), 332
 testMissingInclude() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testNameVariant() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testNegativeCondition() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
 testNetworkXML (class in virttest.libvirt_xml_unittest), 333
 testNodedevXML (class in virttest.libvirt_xml_unittest), 333
 testNodedevXMLBase (class in virttest.libvirt_xml_unittest), 333
 TestNumaNode (class in virttest.utils_misc_unittest), 479
 testObjects() (virttest.utils_params_unittest.TestParams method), 496
 testObjectsParams() (virttest.utils_params_unittest.TestParams method), 496
 TestParams (class in virttest.utils_params_unittest), 496
 testParseBlocks() (virttest.qemu_monitor_unittest.InfoBlocks

- method), 373
- testPCIXML (class in virttest.libvirt_xml_unittest), 333
- TestPropCan (class in virttest.propcan_unittest), 358
- TestPropCanBase (class in virttest.propcan_unittest), 358
- TestQemuiface (class in virttest.utils_net_unittest), 492
- testRemove() (virttest.remote_unittest.RemoteFileTest method), 397
- TestSandboxes (class in virttest.lvsb_base), 341
- testSEEA() (virttest.remote_unittest.RemoteFileTest method), 397
- testSerialXML (class in virttest.libvirt_xml_unittest), 333
- TestServiceManager (class in virttest.service_unittest), 401
- TestSimpleSandboxes (class in virttest.lvsb), 338
- testSimpleVariant() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
- testStubXML (class in virttest.libvirt_xml_unittest), 334
- testSub() (virttest.remote_unittest.RemoteFileTest method), 398
- testSyntaxErrors() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
- TestSystemdServiceManager (class in virttest.service_unittest), 401
- TestSysVInitServiceManager (class in virttest.service_unittest), 401
- testTwoNodes() (virttest.qemu_monitor_unittest.InfoNumaTest method), 373
- TestUtilsMisc (class in virttest.utils_misc_unittest), 479
- testVariableAssignment() (virttest.cartesian_config_unittest.CartesianConfigTest method), 308
- TestVersionableClass (class in virttest.versionable_class_unittest), 507
- TestVirshClosure (class in virttest.virsh_unittest), 552
- TestVirshClosure() (virttest.virsh_unittest.ConstructorsTest method), 551
- TestVirshClosure.SomeClass (class in virttest.virsh_unittest), 552
- TestVirtIface (class in virttest.utils_net_unittest), 492
- TestVirtIface.VirtIface (class in virttest.utils_net_unittest), 492
- testVMCPUTuneXML (class in virttest.libvirt_xml_unittest), 334
- TestVmNet (class in virttest.utils_net_unittest), 493
- TestVmNetStyle (class in virttest.utils_net_unittest), 493
- TestVmNetSubclasses (class in virttest.utils_net_unittest), 493
- TestVMXML (class in virttest.libvirt_xml_unittest), 332
- testVMXMLDevices (class in virttest.libvirt_xml_unittest), 334
- testZeroNodes() (virttest.qemu_monitor_unittest.InfoNumaTest method), 373
- tftp (virttest.utils_net.Qemuiface attribute), 483
- tftp_root (virttest.libvirt_xml.network_xml.IPXML attribute), 181
- THPError, 411
- THPKhugepagedError, 411
- THPNotSupportedError, 411
- THPWriteConfigError, 411
- thread_func_migration() (virttest.utils_test.libvirt.MigrationTest method), 262
- ThRecv (class in virttest.qemu_virtio_port), 380
- ThRecvCheck (class in virttest.qemu_virtio_port), 380
- ThSend (class in virttest.qemu_virtio_port), 380
- ThSendCheck (class in virttest.qemu_virtio_port), 380
- tickpolicy (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199
- time() (virttest.utils_libguestfs.GuestfishPersistentTest method), 453
- timeout (virttest.utils_libguestfs.LibguestfsBase attribute), 458
- timers (virttest.libvirt_xml.vm_xml.VMClockXML attribute), 200
- timezone (virttest.libvirt_xml.vm_xml.VMClockXML attribute), 200
- tls_allowed_dn_list (virttest.utils_conn.TLSConnection attribute), 419
- tls_port (virttest.utils_conn.TLSConnection attribute), 419
- tls_sanity_cert (virttest.utils_conn.TLSConnection attribute), 419
- tls_verify_cert (virttest.utils_conn.TLSConnection attribute), 419
- TLSConnection (class in virttest.utils_conn), 418
- tlsPort (virttest.libvirt_xml.devices.graphics.Graphics attribute), 130
- tmp_dir (virttest.remote_unittest.RemoteFileTest attribute), 398
- tmp_dir (virttest.utils_conn.ConnectionBase attribute), 417
- to_text() (virttest.staging.utils_koji.KojiPkgSpec method), 254
- Token (class in virttest.cartesian_config), 307
- topology (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 199
- TopologyXML (class in virttest.libvirt_xml.capability_xml), 179
- tostring() (in module virttest.element_tree), 310
- total_bytes_sec (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), 127
- total_iops_sec (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), 127
- touch() (virttest.utils_libguestfs.GuestfishPersistent method), 454
- Track (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML attribute), 199
- transient (virttest.libvirt_xml.devices.disk.Disk attribute), 128

transient (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196

translate_path() (virttest.http_server.HTTPRequestHandler method), 317

transmogrify_sub_dirs() (in module virttest.utils_selinux), 500

transmogrify_usr_local() (in module virttest.utils_selinux), 500

TransparentHugePageConfig (class in virttest.test_setup), 412

TreeBuilder (class in virttest.element_tree), 310

truncate() (virttest.remote.RemoteFile method), 391

ttyconsole() (in module virttest.virsh), 546

tune2fs() (virttest.utils_libguestfs.GuestfishPersistent method), 454

tune2fs_l() (virttest.utils_libguestfs.GuestfishPersistent method), 454

txt (virttest.libvirt_xml.network_xml.DNSXML attribute), 180

type (virttest.libvirt_xml.devices.controller.Controller attribute), 124

type (virttest.libvirt_xml.nwfilter_protocols.icmp.Icmp.Attr attribute), 149

type (virttest.libvirt_xml.nwfilter_protocols.icmpv6.Icmpv6.Attr attribute), 150

type (virttest.libvirt_xml.nwfilter_protocols.stp.Stp.Attr attribute), 160

type (virttest.libvirt_xml.vm_xml.VMOSXML attribute), 202

type_check() (in module virttest.libvirt_xml.accessors), 175

type_name (virttest.libvirt_xml.devices.base.TypedDeviceBase attribute), 122

type_name (virttest.libvirt_xml.nwfilter_protocols.base.TypedDeviceBase attribute), 145

TypedDeviceBase (class in virttest.libvirt_xml.devices.base), 122

TypedDeviceBase (class in virttest.libvirt_xml.nwfilter_protocols.base), 145

TypedFoobar (virttest.libvirt_xml_unittest.testStubXML attribute), 334

U

Udp (class in virttest.libvirt_xml.nwfilter_protocols.udp), 163

Udp.Attr (class in virttest.libvirt_xml.nwfilter_protocols.udp), 163

udp_copy_between_remotes() (in module virttest.remote), 396

Udp_ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.udp_ipv6), 164

Udp_ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.udp_ipv6), 164

Udplite (class in virttest.libvirt_xml.nwfilter_protocols.udplite), 165

Udplite.Attr (class in virttest.libvirt_xml.nwfilter_protocols.udplite), 165

Udplite_ipv6 (class in virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6), 166

Udplite_ipv6.Attr (class in virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6), 167

umask() (virttest.utils_libguestfs.GuestfishPersistent method), 454

umount() (in module virttest.utils_disk), 422

umount() (in module virttest.utils_misc), 477

umount() (virttest.lvm.Volume method), 337

umount() (virttest.nfs.Nfs method), 343

umount() (virttest.utils_libguestfs.GuestfishPersistent method), 454

umount_all() (virttest.utils_disk.GuestFSModiDisk method), 421

umount_all() (virttest.utils_libguestfs.GuestfishPersistent method), 454

UnattendedInstallConfig (class in virttest.tests.unattended_install), 257

unbind_device_driver() (in module virttest.utils_misc), 477

uncompress_asset() (in module virttest.asset), 291

undefine() (in module virttest.virsh), 546

undefine() (virttest.libvirt_vm.VM method), 331

undefine() (virttest.libvirt_xml.network_xml.NetworkXML method), 182

undefine() (virttest.libvirt_xml.vm_xml.VMXML method), 207

unexport() (virttest.nfs.Exportfs method), 342

uninstall() (virttest.base_installer.BaseInstaller method), 291

unique() (in module virttest.utils_misc), 477

unit (virttest.libvirt_xml.vm_xml.VMHugepagesXML.PageXML attribute), 200

unix_sock_dir (virttest.utils_conn.UNIXConnection attribute), 420

unix_sock_group (virttest.utils_conn.UNIXConnection attribute), 420

unix_sock_ro_perms (virttest.utils_conn.UNIXConnection attribute), 420

unix_sock_rw_perms (virttest.utils_conn.UNIXConnection attribute), 420

UNIXConnection (class in virttest.utils_conn), 419

UnknownBackendError, 308

unlink() (virttest.xml_utils.TempXMLFile method), 562

unload_ksm_module() (virttest.utils_misc.KSMController method), 464

unload_modules() (virttest.base_installer.BaseInstaller method), 291

- method), 291
- unload_stress() (in module virttest.utils_test), 279
- unload_stress() (virttest.utils_test.HostStress method), 275
- unload_stress() (virttest.utils_test.VMStress method), 276
- unlock_db() (virttest.utils_net.DbNet method), 480
- unlock_file() (in module virttest.utils_misc), 478
- unplug() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- unplug_hmp() (virttest.qemu_devices.qdevices.QBaseDevice method), 220
- unplug_hmp() (virttest.qemu_devices.qdevices.QDevice method), 222
- unplug_hmp() (virttest.qemu_devices.qdevices.QHPDrive method), 222
- unplug_hmp() (virttest.qemu_devices.qdevices.QRHPDrive method), 223
- unplug_hook() (virttest.qemu_devices.qdevices.QBaseDevice method), 221
- unplug_hook() (virttest.qemu_devices.qdevices.QHPDrive method), 222
- unplug_hook() (virttest.qemu_devices.qdevices.QRHPDrive method), 223
- unplug_qmp() (virttest.qemu_devices.qdevices.QBaseDevice method), 221
- unplug_qmp() (virttest.qemu_devices.qdevices.QDevice method), 222
- unplug_qmp() (virttest.qemu_devices.qdevices.QRHPDrive method), 223
- unplug_unhook() (virttest.qemu_devices.qdevices.QBaseDevice method), 221
- unplug_unhook() (virttest.qemu_devices.qdevices.QHPDrive method), 222
- unplug_unhook() (virttest.qemu_devices.qdevices.QRHPDrive method), 223
- unprivileged_user (virttest.virsh.VirshPersistent attribute), 510
- unregister() (in module virttest.funcatexit), 313
- unregister() (virttest.lvm.LVM method), 336
- unregister_lvmdev() (virttest.utils_env.Env method), 423
- unregister_syncserver() (virttest.utils_env.Env method), 424
- unregister_vm() (virttest.utils_env.Env method), 424
- unsetenv() (virttest.utils_libguestfs.GuestfishPersistent method), 454
- UnsupportedCPU, 466
- untyped_address (virttest.libvirt_xml.devices.hostdev.HostDev attribute), 130
- UntypedDeviceBase (class in virttest.libvirt_xml.devices.base), 122
- UntypedDeviceBase (class in virttest.libvirt_xml.nwfilter_protocols.base), 145
- UntypedFoobar (virttest.libvirt_xml_unittest.testStubXML attribute), 334
- up() (virttest.utils_net.Interface method), 482
- update() (virttest.libvirt_xml.vm_xml.VMClockXML.TimerXML method), 199
- update() (virttest.libvirt_xml.vm_xml.VMHugepagesXML.PageXML method), 200
- update() (virttest.propcan.PropCanBase method), 357
- update() (virttest.remote_commander.remote_interface.BaseCmd method), 226
- update() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
- update() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- update() (virttest.staging.backports.simplejson.OrderedDict method), 242
- update_block_prop() (virttest.qemu_qtree.QtreeDisk method), 374
- update_cmd_hash() (virttest.remote_commander.remote_interface.BaseCmd method), 226
- update_db() (virttest.utils_net.DbNet method), 480
- update_device() (in module virttest.virsh), 547
- update_driver_hardware_id() (virttest.tests.unattended_install.UnattendedInstallConfig method), 258
- update_instance() (virttest.ovirt.VMManager method), 349
- update_mac_ip_address() (in module virttest.utils_net), 491
- update_params() (virttest.qemu_qtree.QtreeNode method), 375
- update_polkit_rule() (in module virttest.utils_test.libvirt), 268
- update_qtree_prop() (virttest.qemu_qtree.QtreeNode method), 375
- update_source() (virttest.libvirt_xml.devices.character.CharacterBase method), 123
- update_system_dependent_devs() (virttest.qemu_vm.VM method), 389
- update_target() (virttest.libvirt_xml.devices.character.CharacterBase method), 123
- update_vga_global_default() (virttest.qemu_vm.VM method), 389
- update_vm_disk() (virttest.utils_test.libguestfs.VirtTools method), 260
- update_vm_disk_source() (in module virttest.utils_test.libvirt), 269
- update_source_addresses() (virttest.virt_vm.BaseVM method), 558
- upload() (in module virttest.rss_client), 400
- upload() (virttest.rss_client.FileUploadClient method), 399
- upload() (virttest.utils_libguestfs.GuestfishPersistent method), 454
- upload_offset() (virttest.utils_libguestfs.GuestfishPersistent

- method), 454
- Uri (class in virttest.utils_v2v), 503
- uri (virttest.lvsbs.SandboxService attribute), 342
- uri (virttest.utils_libguestfs.LibguestfsBase attribute), 458
- uri (virttest.virsh.VirshBase attribute), 509
- uri (virttest.virsh.VirshPersistent attribute), 510
- usage (virttest.libvirt_xml.secret_xml.SecretXMLBase attribute), 195
- usage_name (virttest.libvirt_xml.secret_xml.SecretXMLBase attribute), 195
- usb_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 219
- usb_by_variables() (virttest.qemu_devices.qcontainer.DevContainer method), 219
- usbc_by_params() (virttest.qemu_devices.qcontainer.DevContainer method), 219
- usbc_by_variables() (virttest.qemu_devices.qcontainer.DevContainer method), 219
- used (virttest.lvsb_base.SandboxSession attribute), 341
- utils_run() (in module virttest.utils_misc_unittest), 479
- utimens() (virttest.utils_libguestfs.GuestfishPersistent method), 454
- utsname() (virttest.utils_libguestfs.GuestfishPersistent method), 454
- uuid (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 178
- uuid (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- uuid (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189
- uuid (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 191
- uuid (virttest.libvirt_xml.pool_xml.PoolXMLBase attribute), 193
- uuid (virttest.libvirt_xml.secret_xml.SecretXMLBase attribute), 195
- uuid (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- validates (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189
- validates (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 191
- value (virttest.cartesian_config.LOperators attribute), 303
- value_listed() (virttest.test_setup.TransparentHugePageConfig method), 412
- valued_option_dict() (in module virttest.utils_misc), 478
- values() (virttest.propcan.PropCan method), 357
- values() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
- values() (virttest.staging.backports.simplejson.ordered_dict.OrderedDict method), 236
- values() (virttest.staging.backports.simplejson.OrderedDict method), 242
- var_name (virttest.cartesian_config.Label attribute), 304
- Container (virttest.cartesian_config.Node attribute), 306
- vcpu (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- vcpucount() (in module virttest.virsh), 547
- vcpuinfo() (in module virttest.virsh), 547
- vcpuinfo() (virttest.libvirt_vm.VM method), 331
- vcupin() (in module virttest.virsh), 547
- vcupin() (virttest.libvirt_vm.VM method), 331
- vcupins (virttest.libvirt_xml.vm_xml.VMCPUTuneXML attribute), 198
- vectors (virttest.libvirt_xml.devices.controller.Controller attribute), 125
- vectors (virttest.utils_net.Qemuiface attribute), 484
- vendor (virttest.libvirt_xml.capability_xml.CapabilityXML attribute), 178
- vendor (virttest.libvirt_xml.devices.disk.Disk attribute), 128
- vendor (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196
- vendor (virttest.libvirt_xml.vm_xml.VMCPUXML attribute), 199
- vendor_id (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187
- verify() (virttest.qemu_qtree.QtreeNode method), 375
- verify_alive() (virttest.libvirt_vm.VM method), 331
- verify_alive() (virttest.qemu_vm.VM method), 389
- verify_alive() (virttest.virt_vm.BaseVM method), 558
- verify_background_errors() (virttest.standalone_test.Test method), 401
- verify_bsod() (virttest.virt_vm.BaseVM method), 559
- verify_defcon() (in module virttest.utils_selinux), 500
- verify_disk_image_bootable() (virttest.qemu_vm.VM method), 389
- verify_established() (in module virttest.utils_spice), 501
- verify_fsfreeze_status() (virttest.guest_agent.QemuAgent method), 316
- verify_guest_down() (virttest.utils_test.qemu.GuestSuspend method), 269

V

- v2v_cmd() (in module virttest.utils_v2v), 504
- VAgentCmdError, 316
- VAgentConnectError, 316
- VAgentError, 316
- VAgentFreezeStatusError, 316
- VAgentLockError, 316
- VAgentNotSupportedError, 316
- VAgentProtocolError, 316
- VAgentSocketError, 316
- VAgentSuspendError, 316
- VAgentSuspendUnknownModeError, 316
- VAgentSyncError, 316
- validates (virttest.libvirt_xml.base.LibvirtXMLBase attribute), 176

[verify_guest_support_suspend\(\)](#) (virttest.utils_test.qemu.GuestSuspend method), 270
[verify_guest_up\(\)](#) (virttest.utils_test.qemu.GuestSuspend method), 270
[verify_host_dmesg\(\)](#) (in module virttest.utils_misc), 478
[verify_hotplug\(\)](#) (virttest.qemu_devices.qdevices.QBaseDevice method), 221
[verify_hotplug\(\)](#) (virttest.qemu_devices.qdevices.QDevice method), 222
[verify_hotplug\(\)](#) (virttest.qemu_devices.qdevices.QHPDrive method), 222
[verify_illegal_instruction\(\)](#) (virttest.virt_vm.BaseVM method), 559
[verify_ip_address_ownership\(\)](#) (in module virttest.utils_net), 491
[verify_kernel_crash\(\)](#) (virttest.virt_vm.BaseVM method), 559
[verify_kvm_internal_error\(\)](#) (virttest.qemu_vm.VM method), 390
[verify_mandatory_programs\(\)](#) (in module virttest.bootstrap), 294
[verify_recommended_programs\(\)](#) (in module virttest.bootstrap), 294
[verify_responsive\(\)](#) (virttest.guest_agent.QemuAgent method), 316
[verify_responsive\(\)](#) (virttest.qemu_monitor.HumanMonitor method), 365
[verify_responsive\(\)](#) (virttest.qemu_monitor.QMPMonitor method), 372
[verify_running_as_root\(\)](#) (in module virttest.utils_misc), 478
[verify_selinux\(\)](#) (in module virttest.bootstrap), 294
[verify_status\(\)](#) (virttest.qemu_monitor.HumanMonitor method), 365
[verify_status\(\)](#) (virttest.qemu_monitor.QMPMonitor method), 372
[verify_status\(\)](#) (virttest.qemu_vm.VM method), 390
[verify_supported_cmd\(\)](#) (virttest.qemu_monitor.Monitor method), 366
[verify_supported_hmp_cmd\(\)](#) (virttest.qemu_monitor.QMPMonitor method), 373
[verify_unplug\(\)](#) (virttest.qemu_devices.qdevices.QBaseDevice method), 221
[verify_unplug\(\)](#) (virttest.qemu_devices.qdevices.QDevice method), 222
[verify_unplug\(\)](#) (virttest.qemu_devices.qdevices.QHPDrive method), 222
[verify_userspace_crash\(\)](#) (virttest.qemu_vm.VM method), 390
[verify_userspace_crash\(\)](#) (virttest.virt_vm.BaseVM method), 559
[verify_vdagent\(\)](#) (in module virttest.utils_spice), 501
[verify_virsh_console\(\)](#) (in module virttest.utils_test.libvirt), 269
[verify_virtio\(\)](#) (in module virttest.utils_spice), 502
[version\(\)](#) (in module virttest.virsh), 547
[version\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 454
[VersionableClass](#) (class in virttest.versionable_class), 506
[VFIOController](#) (class in virttest.utils_misc), 466
[VFIOError](#), 467
[vfs_label\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vfs_type\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vfs_uuid\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vg_activate\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vg_activate_all\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vg_check\(\)](#) (in module virttest.staging.lv_utils), 244
[vg_list\(\)](#) (in module virttest.staging.lv_utils), 244
[vg_name](#) (virttest.libvirt_xml.pool_xml.SourceXML attribute), 194
[vg_ramdisk_cleanup\(\)](#) (in module virttest.staging.lv_utils), 244
[vgcreate\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vglvuuids\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgpvuuuids\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgremove\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgrename\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgs\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgs_full\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 455
[vgscan\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 456
[vguuid\(\)](#) (virttest.utils_libguestfs.GuestfishPersistent method), 456
[vhostfds](#) (virttest.utils_net.QemuIFace attribute), 484
[Video](#) (class in virttest.libvirt_xml.devices.video), 138
[video_maker\(\)](#) (in module virttest.video_maker), 508
[viewitems\(\)](#) (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
[viewitems\(\)](#) (virttest.staging.backports.simplejson.OrderedDict method), 242
[viewkeys\(\)](#) (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232
[viewkeys\(\)](#) (virttest.staging.backports.simplejson.OrderedDict method), 242

viewvalues() (virttest.staging.backports.collections.OrderedDict.OrderedDict method), 232

viewvalues() (virttest.staging.backports.simplejson.OrderedDict method), 242

Virsh (class in virttest.virsh), 509

virsh (virttest.libvirt_xml.base.LibvirtXMLBase attribute), 176

virsh (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189

virsh (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 191

virsh (virttest.virsh_unittest.ModuleLoad attribute), 552

virsh (virttest.virsh_unittest.ModuleLoadCheckVirsh attribute), 552

virsh_exec (virttest.virsh.VirshBase attribute), 509

VirshBase (class in virttest.virsh), 509

VirshClassHasHelpCommandTest (class in virttest.virsh_unittest), 552

VirshClosure (class in virttest.virsh), 509

VirshConnectBack (class in virttest.virsh), 509

VirshHasHelpCommandTest (class in virttest.virsh_unittest), 552

VirshHelpCommandTest (class in virttest.virsh_unittest), 553

VirshPersistent (class in virttest.virsh), 509

VirshPersistentClassHasHelpCommandTest (class in virttest.virsh_unittest), 553

VirshSession (class in virttest.virsh), 510

VirshSessionSASL (class in virttest.utils_sasl), 497

virt_cat_cmd() (in module virttest.utils_libguestfs), 459

virt_clone_cmd() (in module virttest.utils_libguestfs), 459

virt_cmd_contain_opt() (in module virttest.utils_libguestfs), 459

virt_copy_in() (in module virttest.utils_libguestfs), 459

virt_copy_out() (in module virttest.utils_libguestfs), 459

virt_df() (in module virttest.utils_libguestfs), 459

virt_edit_cmd() (in module virttest.utils_libguestfs), 459

virt_filesystems() (in module virttest.utils_libguestfs), 460

virt_format() (in module virttest.utils_libguestfs), 460

virt_functions (virttest.libvirt_xml.nodedev_xml.PCIXML attribute), 187

virt_inspector() (in module virttest.utils_libguestfs), 460

virt_list_filesystems() (in module virttest.utils_libguestfs), 460

virt_list_partitions() (in module virttest.utils_libguestfs), 460

virt_list_partitions_cmd() (in module virttest.utils_libguestfs), 460

virt_ls_cmd() (in module virttest.utils_libguestfs), 461

virt_resize_cmd() (in module virttest.utils_libguestfs), 461

virt_sparsify_cmd() (in module virttest.utils_libguestfs), 461

virt_sysprep_cmd() (in module virttest.utils_libguestfs), 461

virt_sysprep_operations() (in module virttest.utils_libguestfs), 461

virt_tar_in() (in module virttest.utils_libguestfs), 461

virt_tar_out() (in module virttest.utils_libguestfs), 461

virt_xml_validate() (virttest.libvirt_xml.base.LibvirtXMLBase static method), 177

VirtIface (class in virttest.utils_net), 485

VirtInstallException, 292

VirtInstallFailed, 293

VirtInstallNotInstalled, 293

VirtioConsole (class in virttest.qemu_virtio_port), 381

VirtioPortException, 381

VirtioPortFatalException, 381

VirtioPortTest (class in virttest.utils_virtio_port), 504

VirtioSerial (class in virttest.qemu_virtio_port), 381

VirtLoggingConfig (class in virttest.utils_misc), 467

VirtNet (class in virttest.utils_net), 485

virttest (module), 566

virttest.aexpect (module), 280

virttest.arch (module), 289

virttest.asset (module), 289

virttest.base_installer (module), 291

virttest.bootstrap (module), 293

virttest.build_helper (module), 294

virttest.cartesian_config (module), 298

virttest.cartesian_config_unittest (module), 307

virttest.ceph (module), 308

virttest.common (module), 308

virttest.data_dir (module), 308

virttest.defaults (module), 309

virttest.element_path (module), 309

virttest.element_tree (module), 309

virttest.env_process (module), 310

virttest.functexit (module), 312

virttest.gluster (module), 313

virttest.guest_agent (module), 313

virttest.http_server (module), 316

virttest.installer (module), 317

virttest.installer_unittest (module), 318

virttest.iscsi (module), 318

virttest.iscsi_unittest (module), 320

virttest.libvirt_network_unittest (module), 320

virttest.libvirt_storage (module), 321

virttest.libvirt_storage_unittest (module), 323

virttest.libvirt_vm (module), 324

virttest.libvirt_xml (module), 212

virttest.libvirt_xml.accessors (module), 169

virttest.libvirt_xml.base (module), 175

virttest.libvirt_xml.capability_xml (module), 177

virttest.libvirt_xml.devices (module), 138

virttest.libvirt_xml.devices.address (module), 121

virttest.libvirt_xml.devices.base (module), 122
virttest.libvirt_xml.devices.channel (module), 122
virttest.libvirt_xml.devices.character (module), 123
virttest.libvirt_xml.devices.console (module), 124
virttest.libvirt_xml.devices.controller (module), 124
virttest.libvirt_xml.devices.disk (module), 125
virttest.libvirt_xml.devices.emulator (module), 128
virttest.libvirt_xml.devices.filesystem (module), 128
virttest.libvirt_xml.devices.graphics (module), 128
virttest.libvirt_xml.devices.hostdev (module), 130
virttest.libvirt_xml.devices.hub (module), 130
virttest.libvirt_xml.devices.input (module), 131
virttest.libvirt_xml.devices.interface (module), 131
virttest.libvirt_xml.devices.lease (module), 133
virttest.libvirt_xml.devices.librarian (module), 133
virttest.libvirt_xml.devices.memballoon (module), 134
virttest.libvirt_xml.devices.memory (module), 134
virttest.libvirt_xml.devices.panic (module), 135
virttest.libvirt_xml.devices.parallel (module), 135
virttest.libvirt_xml.devices.redirdev (module), 135
virttest.libvirt_xml.devices.rng (module), 136
virttest.libvirt_xml.devices.seclabel (module), 136
virttest.libvirt_xml.devices.serial (module), 137
virttest.libvirt_xml.devices.smartcard (module), 137
virttest.libvirt_xml.devices.sound (module), 138
virttest.libvirt_xml.devices.video (module), 138
virttest.libvirt_xml.devices.watchdog (module), 138
virttest.libvirt_xml.network_xml (module), 179
virttest.libvirt_xml.nodedev_xml (module), 185
virttest.libvirt_xml.nwfilter_protocols (module), 169
virttest.libvirt_xml.nwfilter_protocols.ah (module), 139
virttest.libvirt_xml.nwfilter_protocols.ah_ipv6 (module), 140
virttest.libvirt_xml.nwfilter_protocols.all (module), 141
virttest.libvirt_xml.nwfilter_protocols.all_ipv6 (module), 142
virttest.libvirt_xml.nwfilter_protocols.arp (module), 143
virttest.libvirt_xml.nwfilter_protocols.base (module), 145
virttest.libvirt_xml.nwfilter_protocols.esp (module), 145
virttest.libvirt_xml.nwfilter_protocols.esp_ipv6 (module), 147
virttest.libvirt_xml.nwfilter_protocols.icmp (module), 148
virttest.libvirt_xml.nwfilter_protocols.icmpv6 (module), 149
virttest.libvirt_xml.nwfilter_protocols.igmp (module), 150
virttest.libvirt_xml.nwfilter_protocols.ip (module), 151
virttest.libvirt_xml.nwfilter_protocols.ipv6 (module), 153
virttest.libvirt_xml.nwfilter_protocols.librarian (module), 154
virttest.libvirt_xml.nwfilter_protocols.mac (module), 154
virttest.libvirt_xml.nwfilter_protocols.rarp (module), 155
virttest.libvirt_xml.nwfilter_protocols.sctp (module), 156
virttest.libvirt_xml.nwfilter_protocols.sctp_ipv6 (module), 157
virttest.libvirt_xml.nwfilter_protocols.stp (module), 158
virttest.libvirt_xml.nwfilter_protocols.tcp (module), 160
virttest.libvirt_xml.nwfilter_protocols.tcp_ipv6 (module), 161
virttest.libvirt_xml.nwfilter_protocols.udp (module), 163
virttest.libvirt_xml.nwfilter_protocols.udp_ipv6 (module), 164
virttest.libvirt_xml.nwfilter_protocols.udplite (module), 165
virttest.libvirt_xml.nwfilter_protocols.udplite_ipv6 (module), 166
virttest.libvirt_xml.nwfilter_protocols.vlan (module), 168
virttest.libvirt_xml.nwfilter_xml (module), 188
virttest.libvirt_xml.pool_xml (module), 191
virttest.libvirt_xml.secret_xml (module), 194
virttest.libvirt_xml.snapshot_xml (module), 195
virttest.libvirt_xml.sysinfo_xml (module), 197
virttest.libvirt_xml.vm_xml (module), 197
virttest.libvirt_xml.vol_xml (module), 210
virttest.libvirt_xml.xcepts (module), 212
virttest.libvirt_xml_unittest (module), 332
virttest.lvm (module), 334
virttest.lvsb (module), 338
virttest.lvsb_base (module), 339
virttest.lvsbs (module), 341
virttest.nfs (module), 342
virttest.nfs_unittest (module), 343
virttest.openvswitch (module), 344
virttest.ovirt (module), 346
virttest.ovs_utils (module), 350
virttest.passfd_setup (module), 351
virttest.postprocess_iozone (module), 351
virttest.ppm_utils (module), 353
virttest.propcan (module), 356
virttest.propcan_unittest (module), 358
virttest.qemu_devices (module), 224
virttest.qemu_devices.qbuses (module), 213
virttest.qemu_devices.qcontainer (module), 215
virttest.qemu_devices.qdevices (module), 219
virttest.qemu_devices.utils (module), 223
virttest.qemu_devices_unittest (module), 358
virttest.qemu_installer (module), 359
virttest.qemu_io (module), 360
virttest.qemu_monitor (module), 361
virttest.qemu_monitor_unittest (module), 373
virttest.qemu_qtree (module), 374
virttest.qemu_qtree_unittest (module), 376
virttest.qemu_storage (module), 377
virttest.qemu_virtio_port (module), 379
virttest.qemu_vm (module), 381
virttest.remote (module), 391
virttest.remote_build (module), 397

- virttest.remote_commander (module), 231
- virttest.remote_commander.messenger (module), 224
- virttest.remote_commander.remote_interface (module), 226
- virttest.remote_commander.remote_master (module), 227
- virttest.remote_commander.remote_runner (module), 229
- virttest.remote_unittest (module), 397
- virttest.RFBDes (module), 279
- virttest.rss_client (module), 398
- virttest.scheduler (module), 400
- virttest.service_unittest (module), 400
- virttest.staging (module), 257
- virttest.staging.backports (module), 243
- virttest.staging.backports.collections (module), 233
- virttest.staging.backports.collections.defaultdict (module), 232
- virttest.staging.backports.collections.namedtuple (module), 232
- virttest.staging.backports.collections.OrderedDict (module), 231
- virttest.staging.backports.simplejson (module), 236
- virttest.staging.backports.simplejson.decoder (module), 233
- virttest.staging.backports.simplejson.encoder (module), 234
- virttest.staging.backports.simplejson.ordered_dict (module), 235
- virttest.staging.backports.simplejson.scanner (module), 236
- virttest.staging.lv_utils (module), 243
- virttest.staging.service (module), 244
- virttest.staging.utils_cgroup (module), 247
- virttest.staging.utils_koji (module), 251
- virttest.staging.utils_memory (module), 256
- virttest.standalone_test (module), 401
- virttest.storage (module), 403
- virttest.syslog_server (module), 405
- virttest.test_setup (module), 407
- virttest.tests (module), 258
- virttest.tests.unattended_install (module), 257
- virttest.utils_cgroup_unittest (module), 412
- virttest.utils_config (module), 412
- virttest.utils_config_unittest (module), 414
- virttest.utils_conn (module), 415
- virttest.utils_disk (module), 420
- virttest.utils_env (module), 422
- virttest.utils_env_unittest (module), 424
- virttest.utils_gdb (module), 426
- virttest.utils_libguestfs (module), 427
- virttest.utils_libguestfs_unittest (module), 462
- virttest.utils_libvirtd (module), 462
- virttest.utils_misc (module), 463
- virttest.utils_misc_unittest (module), 479
- virttest.utils_net (module), 479
- virttest.utils_net_unittest (module), 492
- virttest.utils_netperf (module), 494
- virttest.utils_params (module), 496
- virttest.utils_params_unittest (module), 496
- virttest.utils_sasl (module), 496
- virttest.utils_selinux (module), 497
- virttest.utils_spice (module), 500
- virttest.utils_test (module), 274
- virttest.utils_test.libguestfs (module), 259
- virttest.utils_test.libvirt (module), 261
- virttest.utils_test.qemu (module), 269
- virttest.utils_v2v (module), 502
- virttest.utils_virtio_port (module), 504
- virttest.version (module), 505
- virttest.versionable_class (module), 505
- virttest.versionable_class_unittest (module), 506
- virttest.video_maker (module), 508
- virttest.virsh (module), 508
- virttest.virsh_unittest (module), 551
- virttest.virt_vm (module), 553
- virttest.xml_utils (module), 562
- virttest.xml_utils_unittest (module), 564
- virttest.yumrepo (module), 565
- VirtTools (class in virttest.utils_test.libguestfs), 259
- virtualport_type (virttest.libvirt_xml.devices.interface.Interface attribute), 133
- virtualport_type (virttest.libvirt_xml.network_xml.NetworkXMLBase attribute), 184
- virtualport_type (virttest.libvirt_xml.network_xml.PortgroupXML attribute), 185
- Vlan (class in virttest.libvirt_xml.nwfilter_protocols.vlan), 168
- vlan (virttest.utils_net.QemuIface attribute), 484
- Vlan.Attr (class in virttest.libvirt_xml.nwfilter_protocols.vlan), 168
- vlan_tag (virttest.libvirt_xml.network_xml.PortgroupXML attribute), 185
- VlanError, 486
- vlanid (virttest.libvirt_xml.nwfilter_protocols.vlan.Vlan.Attr attribute), 168
- VM (class in virttest.libvirt_vm), 324
- VM (class in virttest.qemu_vm), 381
- VM (class in virttest.versionable_class_unittest), 507
- VM1 (class in virttest.versionable_class_unittest), 507
- VM_container (class in virttest.versionable_class_unittest), 507
- vm_name (virttest.libvirt_xml.vm_xml.VMXMLBase attribute), 210
- vm_rename() (virttest.libvirt_xml.vm_xml.VMXML static method), 207
- VMAddNetDevError, 559
- VMAddNicError, 559
- VMAddressError, 560
- VMAddressVerificationError, 560

VMBadPATypeError, 560
VMCheck (class in virttest.utils_v2v), 503
VMClockXML (class in virttest.libvirt_xml.vm_xml), 199
VMClockXML.TimerXML (class in virttest.libvirt_xml.vm_xml), 199
VMConfigMissingError, 560
VMCPUTuneXML (class in virttest.libvirt_xml.vm_xml), 197
VMCPUXML (class in virttest.libvirt_xml.vm_xml), 198
VMCreateError, 560
VMDeadError, 560
VMDeadKernelCrashError, 560
VMDelNetDevError, 560
VMDelNicError, 560
VMDeviceError, 560
VMDeviceNotSupportedError, 560
VMError, 560
VMFeaturesXML (class in virttest.libvirt_xml.vm_xml), 200
VMHashMismatchError, 560
VMHugePageError, 560
VMHugepagesXML (class in virttest.libvirt_xml.vm_xml), 200
VMHugepagesXML.PageXML (class in virttest.libvirt_xml.vm_xml), 200
VMImageCheckError, 560
VMImageMissingError, 560
VMInterfaceIndexError, 560
VMInvalidInstructionCode, 560
VMIPAddressMissingError, 560
VMIPv6AddressError, 484
VMIPv6NeighNotFound, 484
VMKVMInitError, 560
VMMACAddressMissingError, 560
VMManager (class in virttest.ovirt), 347
VMMemBackingXML (class in virttest.libvirt_xml.vm_xml), 201
VMMemTuneXML (class in virttest.libvirt_xml.vm_xml), 201
VMMigrateCancelError, 561
VMMigrateError, 561
VMMigrateFailedError, 561
VMMigrateProtoUnknownError, 561
VMMigrateProtoUnsupportedError, 390
VMMigrateStateMismatchError, 561
VMMigrateTimeoutError, 561
VMNet (class in virttest.utils_net), 484
VMNet_Style_Map (virttest.utils_net.VMNetStyle attribute), 484
VMNetError, 484
VMNetStyle (class in virttest.utils_net), 484
VMOSXML (class in virttest.libvirt_xml.vm_xml), 201
VMPEAError, 561
VMPCIDeviceError, 561
VMPCIOOutOfRangeError, 561
VMPCISlotInUseError, 561
VMPMXML (class in virttest.libvirt_xml.vm_xml), 202
VMPortNotRedirectedError, 561
VMPostCreateError, 561
VMRebootError, 561
VMRemoveError, 561
VMScreenInactiveError, 561
VMStartError, 561
VMStatusError, 561
VMStress (class in virttest.utils_test), 275
VMUnknownNetTypeError, 562
VMUSBControllerError, 561
VMUSBControllerMissingError, 561
VMUSBControllerPortFullError, 561
VMUSBError, 561
VMUSBPortInUseError, 562
VMXML (class in virttest.libvirt_xml.vm_xml), 202
VMXMLBase (class in virttest.libvirt_xml.vm_xml), 207
VMXMLDevices (class in virttest.libvirt_xml.vm_xml), 210
vncdisplay() (in module virttest.virsh), 548
vnet_hdr_probe() (in module virttest.utils_net), 492
vnet_mq_probe() (in module virttest.utils_net), 492
vol_clone() (in module virttest.virsh), 548
vol_create() (in module virttest.virsh), 548
vol_create_as() (in module virttest.virsh), 548
vol_create_from() (in module virttest.virsh), 549
vol_delete() (in module virttest.virsh), 549
vol_download() (in module virttest.virsh), 549
vol_dumpxml() (in module virttest.virsh), 549
vol_info() (in module virttest.virsh), 549
vol_key() (in module virttest.virsh), 550
vol_list() (in module virttest.virsh), 550
vol_name() (in module virttest.virsh), 550
vol_path() (in module virttest.virsh), 550
vol_pool() (in module virttest.virsh), 550
vol_resize() (in module virttest.virsh), 551
vol_upload() (in module virttest.virsh), 551
vol_wipe() (in module virttest.virsh), 551
Volume (class in virttest.lvm), 337
volume (virttest.libvirt_xml.secret_xml.SecretXMLBase attribute), 195
volume_exists() (virttest.libvirt_storage.PoolVolume method), 321
volume_info() (virttest.libvirt_storage.PoolVolume method), 321
VolumeGroup (class in virttest.lvm), 337
VolXML (class in virttest.libvirt_xml.vol_xml), 210
VolXML.Encryption (class in virttest.libvirt_xml.vol_xml), 210
VolXMLBase (class in virttest.libvirt_xml.vol_xml), 211
VTAttachError, 259

VTError, 259
 VTMountError, 259
 VTXMLParseError, 259

W

wait() (virttest.remote_commander.remote_master.CmdMaster method), 228
 wait() (virttest.remote_commander.remote_master.CommanderMaster method), 228
 wait_for() (in module virttest.utils_misc), 478
 wait_for_create_monitor() (in module virttest.qemu_monitor), 373
 wait_for_get_address() (virttest.virt_vm.BaseVM method), 559
 wait_for_login() (in module virttest.remote), 397
 wait_for_login() (virttest.libvirt_vm.VM method), 331
 wait_for_login() (virttest.virt_vm.BaseVM method), 559
 wait_for_match() (virttest.utils_v2v.WindowsVMCheck method), 504
 wait_for_migration() (virttest.qemu_vm.VM method), 390
 wait_for_serial_login() (virttest.virt_vm.BaseVM method), 559
 wait_for_shutdown() (virttest.libvirt_vm.VM method), 331
 wait_for_shutdown() (virttest.qemu_vm.VM method), 390
 wait_for_start() (virttest.utils_gdb.GDB method), 426
 wait_for_status() (virttest.qemu_vm.VM method), 390
 wait_for_stop() (virttest.utils_gdb.GDB method), 426
 wait_for_stop() (virttest.utils_libvirtd.LibvirtdSession method), 463
 wait_for_termination() (virttest.utils_gdb.GDB method), 426
 wait_for_termination() (virttest.utils_libvirtd.LibvirtdSession method), 463
 wait_for_working() (virttest.utils_libvirtd.LibvirtdSession method), 463
 wait_response() (virttest.remote_commander.remote_master.CmdMaster method), 228
 wait_response() (virttest.remote_commander.remote_master.CommanderMaster method), 229
 wait_timeout() (in module virttest.remote_commander.remote_master), 229
 wait_timeout() (in module virttest.utils_spice), 502
 wait_until_dead() (virttest.qemu_vm.VM method), 390
 wait_until_paused() (virttest.qemu_vm.VM method), 390
 WaitHostStateTimeoutError, 349
 WaitStateTimeoutError, 349
 WaitVMStateTimeoutError, 349
 warning_issued (virttest.libvirt_xml.devices.base.StubDeviceMeta attribute), 122
 warp_init_del() (in module virttest.utils_net), 492
 wash_the_device_out() (virttest.qemu_devices.qcontainer.DevContainer method), 219
 Watchdog (class in virttest.libvirt_xml.devices.watchdog), 138
 WindowsVMCheck (class in virttest.utils_v2v), 503
 work() (virttest.remote_commander.remote_runner.CmdSlave method), 229
 work() (virttest.scheduler.scheduler method), 400
 write() (virttest.element_tree.ElementTree method), 310
 write() (virttest.remote.AexpectIOWrapperOut method), 391
 write() (virttest.remote_commander.messenger.IOWrapper method), 225
 write() (virttest.remote_commander.messenger.StdIOWrapperOut method), 226
 write() (virttest.utils_libguestfs.GuestfishPersistent method), 456
 write() (virttest.xml_utils.XMLTreeFile method), 564
 write_append() (virttest.utils_libguestfs.GuestfishPersistent method), 456
 write_bytes_sec (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), 127
 write_file() (virttest.utils_test.libguestfs.GuestfishTools method), 259
 write_file_with_guestmount() (virttest.utils_test.libguestfs.VirtTools method), 260
 write_iops_sec (virttest.libvirt_xml.devices.disk.Disk.IOTune attribute), 127
 write_msg() (virttest.remote_commander.messenger.Messenger method), 225
 write_pid() (in module virttest.utils_misc), 478
 write_subtests_files() (in module virttest.bootstrap), 294
 write_test_keyval() (virttest.standalone_test.Test method), 401
 write_to_image_file() (virttest.utils_disk.GuestFSModiDisk method), 421
 write_version_keyval() (virttest.base_installer.BaseInstaller method), 291
 wwn (virttest.libvirt_xml.devices.disk.Disk attribute), 128
 wwn (virttest.libvirt_xml.snapshot_xml.SnapshotXML.SnapDiskXML attribute), 196
 wwnn (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 187
 wwpm (virttest.libvirt_xml.nodedev_xml.NodedevXMLBase attribute), 187
X
 xml (virttest.libvirt_xml.base.LibvirtXMLBase attribute), 177
 xml (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189

- xml (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 191
- XML (virttest.libvirt_xml_unittest.testSerialXML attribute), 334
- XML() (in module virttest.element_tree), 310
- xml_test_data (class in virttest.xml_utils_unittest), 565
- XMLAttribute (class in virttest.libvirt_xml.accessors), 170
- XMLAttribute.Delter (class in virttest.libvirt_xml.accessors), 170
- XMLAttribute.Getter (class in virttest.libvirt_xml.accessors), 170
- XMLAttribute.Setter (class in virttest.libvirt_xml.accessors), 170
- XMLBackup (class in virttest.xml_utils), 563
- XMLElementBool (class in virttest.libvirt_xml.accessors), 170
- XMLElementBool.Delter (class in virttest.libvirt_xml.accessors), 171
- XMLElementBool.Getter (class in virttest.libvirt_xml.accessors), 171
- XMLElementBool.Setter (class in virttest.libvirt_xml.accessors), 171
- XMLElementDict (class in virttest.libvirt_xml.accessors), 171
- XMLElementDict.Delter (class in virttest.libvirt_xml.accessors), 171
- XMLElementDict.Getter (class in virttest.libvirt_xml.accessors), 172
- XMLElementDict.Setter (class in virttest.libvirt_xml.accessors), 172
- XMLElementInt (class in virttest.libvirt_xml.accessors), 172
- XMLElementInt.Delter (class in virttest.libvirt_xml.accessors), 172
- XMLElementInt.Getter (class in virttest.libvirt_xml.accessors), 172
- XMLElementInt.Setter (class in virttest.libvirt_xml.accessors), 173
- XMLElementList (class in virttest.libvirt_xml.accessors), 173
- XMLElementList.Delter (class in virttest.libvirt_xml.accessors), 173
- XMLElementList.Getter (class in virttest.libvirt_xml.accessors), 173
- XMLElementList.Setter (class in virttest.libvirt_xml.accessors), 173
- XMLElementNest (class in virttest.libvirt_xml.accessors), 174
- XMLElementNest.Delter (class in virttest.libvirt_xml.accessors), 174
- XMLElementNest.Getter (class in virttest.libvirt_xml.accessors), 174
- XMLElementNest.Setter (class in virttest.libvirt_xml.accessors), 174
- XMLElementText (class in virttest.libvirt_xml.accessors), 175
- XMLElementText.Delter (class in virttest.libvirt_xml.accessors), 175
- XMLElementText.Getter (class in virttest.libvirt_xml.accessors), 175
- XMLElementText.Setter (class in virttest.libvirt_xml.accessors), 175
- XMLParser (in module virttest.element_tree), 310
- xmlstr (virttest.lvsbs.SandboxService attribute), 342
- XMLTreeBuilder (class in virttest.element_tree), 310
- XMLTreeFile (class in virttest.xml_utils), 563
- xmltreefile (virttest.libvirt_xml.base.LibvirtXMLBase attribute), 177
- in xmltreefile (virttest.libvirt_xml.nwfilter_xml.NwfilterXML attribute), 189
- in xmltreefile (virttest.libvirt_xml.nwfilter_xml.NwfilterXMLBase attribute), 191
- in xmltreefile() (virttest.libvirt_xml.accessors.AccessorBase method), 169
- in xpath_descendant_or_self (class in virttest.element_path), 309
- in xpath_tokenizer() (in module virttest.element_path), 309
- Y**
- yum_install() (in module virttest.utils_misc), 478
- in yum_install() (in module virttest.utils_test.libvirt), 269
- YumInstaller (class in virttest.base_installer), 293
- in YumRepo (class in virttest.yumrepo), 565
- Z**
- in zegrep() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zegrepi() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zero() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- zero_counter() (virttest.utils_net_unittest.TestVmNetSubclasses method), 494
- in zero_device() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zfgrep() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zfgrepi() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zgrep() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zgrepi() (virttest.utils_libguestfs.GuestfishPersistent method), 456
- in zlib_compression (virttest.libvirt_xml.devices.graphics.Graphics attribute), 130