Red Hat Virtualization 4.0

Python SDK Guide

Using the Red Hat Virtualization Python SDK

Last Updated: 2018-03-29

Red Hat Virtualization 4.0 Python SDK Guide

Using the Red Hat Virtualization Python SDK

Red Hat Virtualization Documentation Team
Red Hat Customer Content Services
rhev-docs@redhat.com

Legal Notice

Abstract

This guide describes how to install and work with version 3 and version 4 of the Red Hat Virtualization Python software development kit.

Table of Contents

PART I. THE PYTHON SOFWARE DEVELOPMENT KIT

CHAPTER 1. OVERVIEW

The Python software development kit is a collection of classes that allows you to interact with the Red Hat Virtualization Manager in Python-based projects. By downloading these classes and adding them to your project, you can access a range of functionality for high-level automation of administrative tasks.

Red Hat Virtualization provides two versions of the Python software development kit:

`Version 3`

The V3 Python software development kit provides backwards compatibility with the class and method structure provided in the Python software development kit as of the latest release of Red Hat Enterprise Virtualization 3.6. Applications written using using the Python software development kit from Red Hat Enterprise Virtualization 3.6 can be used with this version without modification.

`Version 4`

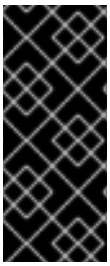
The V4 Python software development kit provides an updated set of class and method names and signatures. Applications written using the Python software development kit from Red Hat Enterprise Virtualization 3.6 must be updated before they can be used with this version.

Either version of the Python software development kit can be used in a Red Hat Virtualization environment as required by installing the corresponding package and adding the required libraries to your Python project.

1.1. PREREQUISITES

To install the Python software development kit, you must have:

- A system where Red Hat Enterprise Linux 7 is installed. Both the Server and Workstation variants are supported.
- A subscription to Red Hat Virtualization entitlements.



IMPORTANT

The software development kit is an interface for the Red Hat Virtualization REST API. As such, you must use the version of the software development kit that corresponds to the version of your Red Hat Virtualization environment. For example, if you are using Red Hat Virtualization 4.0, you must use the version of the software development kit designed for 4.0.

1.2. INSTALLING THE PYTHON SOFTWARE DEVELOPMENT KIT

Install the Python software development kit.

Procedure 1.1. Installing the Python Software Development Kit

1. Enable the required channels:

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-rhv-4.0-rpms
```

2. Install the required packages:

- For V3:

```
# yum install ovirt-engine-sdk-python
```

- For V4:

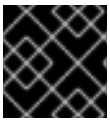
```
# yum install python-ovirt-engine-sdk4
```

The Python software development kit and accompanying documentation are downloaded to the **/usr/lib/python2.7/site-packages/ovirtsdk/** directory, and can now be added to Python projects.

CHAPTER 2. PYTHON QUICK START EXAMPLE

2.1. PYTHON QUICK START INTRODUCTION

This chapter provides a series of examples demonstrating the steps to create a virtual machine within a basic Red Hat Virtualization environment, using the Python SDK.



IMPORTANT

The examples in this chapter are designed to work with V3 of the Python SDK.

These examples use the ovirtsdk Python library provided by the ovirt-engine-sdk-python package. This package is available to systems subscribed to a **`Red Hat Virtualization`** entitlement pool in Red Hat Subscription Manager. See Section 1.2, “Installing the Python Software Development Kit” for more information on subscribing your system(s) to download the software.

You will also need:

- A networked installation of Red Hat Virtualization Manager.
- A networked and configured Red Hat Virtualization Host.
- An ISO image file containing an operating system for installation on a virtual machine.
- A working understanding of both the logical and physical objects that make up a Red Hat Virtualization environment.
- A working understanding of the Python programming language.

IMPORTANT

All Python examples include placeholders for authentication details (*USER* for user name, and *PASS* for password). Ensure all requests performed with Python fulfill the authentication requirements of your environment.

NOTE

Red Hat Virtualization Manager generates a globally unique identifier (GUID) for the **`id`** attribute for each resource. Identifier codes in these examples might appear different to the identifier codes in your Red Hat Virtualization environment.

NOTE

These Python examples contain only basic exception and error handling logic. For more information on the exception handling specific to the SDK, refer to the pydoc for the **`ovirtsdk.infrastructure.errors`** module.

```
$ pydoc ovirtsdk.infrastructure.errors
```

2.2. EXAMPLE: ACCESSING THE API ENTRY POINT USING PYTHON

The ovirtsdk Python library provides the **API** class, which acts as the entry point for the API.

Example 2.1. Accessing the API entry point using Python

This python example connects to an instance of the REST API provided by the Red Hat Virtualization Manager at **rhevm.demo.redhat.com**. To connect the example creates an instance of the **API** class If connection was successful a message is printed. Finally the **disconnect()** method of the **API** class is called to close the connection.

The parameters provided to the constructor for the **API** class in this example are:

- The **url** of the Manager to which to connect.
- The **username** of the user by which to authenticate.
- The **password** of the user by which to authenticate.
- The **ca_file**, which is the path to a certificate. The certificate is expected to be a copy of the one for the Manager's Certificate Authority. It can be obtained from **https://*[engine-fqdn]*ovirt-engine/services/pki-resource?resource=ca-certificate&format=*X509-PEM-CA***.

The constructor for the **API** class supports other parameters. Only mandatory parameters are specified in this example.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    print "Connected to %s successfully!" % api.get_product_info().name

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the connection attempt was successful, the example outputs the text:

```
Connected to Red Hat Virtualization Manager successfully!
```

2.3. EXAMPLE: LISTING THE DATA CENTER COLLECTION USING PYTHON

The **API** class provides access to a data centers collection, named **datacenters**. This collection contains all data centers in the environment.

Example 2.2. Listing the Data Center Collection using Python

This Python example lists the data centers in the **`datacenters`** collection. It also outputs some basic information about each data center in the collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    dc_list = api.datacenters.list()

    for dc in dc_list:
        print "%s (%s)" % (dc.get_name(), dc.get_id())

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

In an environment where only the **`Default`** data center exists, and it is not activated, the example outputs:

```
Default (d8b74b20-c6e1-11e1-87a3-00163e77e2ed)
```

2.4. EXAMPLE: LISTING THE CLUSTER COLLECTION USING PYTHON

The API class provides a clusters collection, named **`clusters`**. This collection contains all clusters in the environment.

Example 2.3. Listing the clusters collection using Python

This Python example lists the clusters in the **`clusters`** collection. It also outputs some basic information about each cluster in the collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    c_list = api.clusters.list()

    for c in c_list:
        print "%s (%s)" % (c.get_name(), c.get_id())

    api.disconnect()
```

```
except Exception as ex:
    print "Unexpected error: %s" % ex
```

In an environment where only the **Default** cluster exists, the example outputs:

```
Default (99408929-82cf-4dc7-a532-9d998063fa95)
```

2.5. EXAMPLE: LISTING THE LOGICAL NETWORKS COLLECTION USING PYTHON

The **API** class provides access to a logical networks collection, named **networks**. This collection contains all logical networks in the environment.

Example 2.4. Listing the logical networks collection using Python

This Python example lists the logical networks in the **networks** collection. It also outputs some basic information about each network in the collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS",
              ca_file="ca.crt")

    n_list = api.networks.list()

    for n in n_list:
        print "%s (%s)" % (n.get_name(), n.get_id())

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

In an environment where only the default management network exists, the example outputs:

```
ovirtmgmt (00000000-0000-0000-0000-000000000009)
```

2.6. EXAMPLE: LISTING THE HOST COLLECTION USING PYTHON

The **API** class provides access to a hosts collection, named **hosts**. This collection contains all hosts in the environment.

Example 2.5. Listing the host collection using Python

This Python example lists the hosts in the **hosts** collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS",
              ca_file="ca.crt")

    h_list = api.hosts.list()

    for h in h_list:
        print "%s (%s)" % (h.get_name(), h.get_id())

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

In an environment where only one host, named **Atlantic**, has been attached the example outputs:

```
Atlantic (5b333c18-f224-11e1-9bdd-00163e77e2ed)
```

2.7. EXAMPLE: LISTING THE ISO FILES IN AN ISO STORAGE DOMAIN

The **API** class provides access to a storage domain collection, named **storagedomains**. This collection in turn contains a **files** collection that describes the files in a storage domain.

Example 2.6. Listing the ISO Files in an ISO Storage Domain

This Python example prints a list of the ISO files in each ISO storage domain in the Red Hat Virtualization environment:

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    storage_domains = api.storagedomains.list()

    for storage_domain in storage_domains:
        if(storage_domain.get_type() == "iso"):

            print(storage_domain.get_name() + ":\n")

            files = storage_domain.files.list()

            for file in files:
                print("     %s" % file.get_name())
```

```
            print()

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

2.8. EXAMPLE: LISTING THE SIZE OF A VIRTUAL MACHINE

The **API** class provides access to a virtual machine collection, named **vms**. This collection in turn contains a **disks** collection that describes the details of each disk attached to a virtual machine.

Example 2.7. Listing the Size of a Virtual Machine

This Python example prints a list of the virtual machines in the Red Hat Virtualization environment along with their total disk size in bytes:

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    virtual_machines = api.vms.list()

    if len(virtual_machines) > 0:

        print("%-30s  %s" % ("Name","Disk Size"))
        print("================================================")

        for virtual_machine in virtual_machines:

            disks = virtual_machine.disks.list()

            disk_size = 0

            for disk in disks:
                disk_size += disk.get_size()

            print("%-30s: %d" % (virtual_machine.get_name(),
disk_size))

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

2.9. EXAMPLE: CREATING NFS DATA STORAGE USING PYTHON

When a Red Hat Virtualization environment is first being created it is necessary to define at least a data storage domain, and an ISO storage domain. The data storage domain will be used to store virtual disk images while the ISO storage domain will be used to store installation media for guest operating systems.

The **API** class provides access to a storage domains collection, named **storagedomains**. This collection contains all the storage domains in the environment. The **storagedomains** collection can also be used to add and remove storage domains.

NOTE

The code provided in this example assumes that the remote NFS share has been pre-configured for use with Red Hat Virtualization. Refer to the Red Hat Virtualization *Administration Guide* for more information on preparing NFS shares for use.

Example 2.8. Creating NFS data storage using Python

This Python example adds an NFS data domain to the **storagedomains** collection. Adding an NFS storage domain in Python can be broken down into several steps:

1. Identify the data center to which the storage must be attached, using the **get** method of the **datacenters** collection.

```
dc = api.datacenters.get(name="Default")
```

2. Identify the host that must be used to attach the storage, using the **get** method of the **hosts** collection.

```
h = api.hosts.get(name="Atlantic")
```

3. Define the **Storage** parameters for the NFS storage domain. In this example the NFS location **192.0.43.10/storage/data** is being used.

```
s = params.Storage(address="192.0.43.10", path="/storage/data",
type_="nfs")
```

4. Request creation of the storage domain, using the **add** method of the **storagedomains** collection. In addition to the **Storage** parameters it is necessary to pass:
   - A name for the storage domain.
   - The data center object that was retrieved from the **datacenters** collection.
   - The host object that was retrieved from the **hosts** collection.
   - The type of storage domain being added (**data**, **iso**, or **export**).
   - The storage format to use (**v1**, **v2**, or **v3**).

Once these steps are combined, the completed script is:

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    dc = api.datacenters.get(name="Default")
    h = api.hosts.get(name="Atlantic")

    s = params.Storage(address="192.0.43.10", path="/storage/data",
type_="nfs")
    sd_params = params.StorageDomain(name="data1", data_center=dc,
host=h, type_="data", storage_format="v3", storage=s)

    try:
        sd = api.storagedomains.add(sd_params)
        print "Storage Domain '%s' added (%s)." % (sd.get_name())
    except Exception as ex:
        print "Adding storage domain failed: %s" % ex

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** method call is successful then the script will output:

```
Storage Domain 'data1' added (bd954c03-d180-4d16-878c-2aedbdede566).
```

2.10. EXAMPLE: CREATING NFS ISO STORAGE USING PYTHON

To create a virtual machine you must be able to provide installation media for the guest operating system. In a Red Hat Virtualization environment you store the installation media on an ISO storage domain.

NOTE

The code provided in this example assumes that the remote NFS share has been pre-configured for use with Red Hat Virtualization. Refer to the Red Hat Virtualization *Administration Guide* for more information on preparing NFS shares for use.

Example 2.9. Creating NFS ISO storage using Python

This Python example adds an NFS ISO domain to the **`storagedomains`** collection. Adding an NFS storage domain in Python can be broken down into several steps:

1. Identify the data center to which the storage must be attached, using the **`get`** method of the **`datacenters`** collection.

```
dc = api.datacenters.get( name="Default" )
```

2. Identify the host that must be used to attach the storage, using the **get** method of the **hosts** collection.

```
h = api.hosts.get(name="Atlantic")
```

3. Define the **Storage** parameters for the NFS storage domain. In this example the NFS location **192.0.43.10/storage/iso** is being used.

```
s = params.Storage(address="192.0.43.10", path="/storage/iso",
type_="nfs")
```

4. Request creation of the storage domain, using the **add** method of the **storagedomains** collection. In addition to the **Storage** parameters it is necessary to pass:
   - A name for the storage domain.
   - The data center object that was retrieved from the **datacenters** collection.
   - The host object that was retrieved from the **hosts** collection.
   - The type of storage domain being added (**data**, **iso**, or **export**).
   - The storage format to use (**v1**, **v2**, or **v3**).

Once these steps are combined, the completed script is:

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    dc = api.datacenters.get(name="Default")
    h = api.hosts.get(name="Atlantic")

    s = params.Storage(address="192.0.43.10", path="/storage/iso",
type_="nfs")
    sd_params = params.StorageDomain(name="iso1", data_center=dc,
host=h, type_="iso", storage_format="v3", storage=s)

    try:
        sd = api.storagedomains.add(sd_params)
        print "Storage Domain '%s' added (%s)." % (sd.get_name())
    except Exception as ex:
        print "Adding storage domain failed: %s" % ex

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** method call is successful then the script will output:

```
Storage Domain 'iso1' added (789814a7-7b90-4a39-a1fd-f6a98cc915d8).
```

2.11. EXAMPLE: ATTACHING STORAGE DOMAINS TO A DATA CENTER USING PYTHON

Once you have added storage domains to Red Hat Virtualization you must attach them to a data center and activate them before they will be ready for use.

Example 2.10. Attaching storage domains to a data center using Python

This Python example attaches a data storage domain named **data1**, and an ISO storage domain named **iso1** to the **Default** data center. The attach action is facilitated by the **add** method of the data center's **storagedomains** collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    dc = api.datacenters.get(name="Default")

    sd_data = api.storagedomains.get(name="data1")
    sd_iso = api.storagedomains.get(name="iso1")

    try:
        dc_sd = dc.storagedomains.add(sd_data)
        print "Attached data storage domain '%s' to data center '%s'
(Status: %s)." %
        (dc_sd.get_name(), dc.get_name, dc_sd.get_status().get_state())
    except Exception as ex:
        print "Attaching data storage domain to data center failed:
%s." % ex

    try:
        dc_sd = dc.storagedomains.add(sd_iso)
        print "Attached ISO storage domain '%s' to data center '%s'
(Status: %s)." %
        (dc_sd.get_name(), dc.get_name, dc_sd.get_status().get_state())
    except Exception as ex:
        print "Attaching ISO storage domain to data center failed: %s."
% ex

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the calls to the **add** methods are successful then the script will output:

```
Attached data storage domain 'data1' to data center 'Default' (Status:
maintenance).
Attached ISO storage domain 'iso1' to data center 'Default' (Status:
maintenance).
```

Note that the **status** reflects that the storage domains still need to be activated.

2.12. EXAMPLE: ACTIVATING STORAGE DOMAINS USING PYTHON

Once you have added storage domains to Red Hat Virtualization and attached them to a data center you must activate them before they will be ready for use.

Example 2.11. Activating storage domains using Python

This Python example activates a data storage domain named **data1**, and an ISO storage domain named **iso1**. Both storage domains are attached to the **Default** data center. The activate action is facilitated by the **activate** method of the storage domain.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    dc = api.datacenters.get(name="Default")

    sd_data = dc.storagedomains.get(name="data1")
    sd_iso = dc.storagedomains.get(name="iso1")

    try:
        sd_data.activate()
        print "Activated data storage domain '%s' in data center '%s'
(Status: %s)." %
        (sd_data.get_name(), dc.get_name,
sd_data.get_status().get_state())
    except Exception as ex:
        print "Activating data storage domain in data center failed:
%s." % ex

    try:
        sd_iso.activate()
        print "Activated ISO storage domain '%s' in data center '%s'
(Status: %s)." %
        (sd_iso.get_name(), dc.get_name,
sd_iso.get_status().get_state())
    except Exception as ex:
        print "Activating ISO storage domain in data center failed:
```

```
%s." % ex

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **activate** requests are successful then the script will output:

```
Activated data storage domain 'data1' in data center 'Default' (Status:
active).
Activated ISO storage domain 'iso1' in data center 'Default' (Status:
active).
```

Note that the **status** reflects that the storage domains have been activated.

2.13. EXAMPLE: CREATING A VIRTUAL MACHINE USING PYTHON

Virtual machine creation is performed in several steps. The first step, covered here, is to create the virtual machine object itself.

Example 2.12. Creating a virtual machine using Python

This Python example creates a virtual machine named **vm1**. The virtual machine in this example:

- Must have 512 MB of memory, expressed in bytes.

  ```
  vm_memory = 512 * 1024 * 1024
  ```

- Must be attached to the **Default** cluster, and therefore the **Default** data center.

  ```
  vm_cluster = api.clusters.get(name="Default")
  ```

- Must be based on the default **Blank** template.

  ```
  vm_template = api.templates.get(name="Blank")
  ```

- Must boot from the virtual hard disk drive.

  ```
  vm_os = params.OperatingSystem(boot=[params.Boot(dev="hd")])
  ```

These options are combined into a virtual machine parameter object, before using the **add** method of the **vms** collection to create the virtual machine itself.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
```

```
                ca_file="ca.crt")

    vm_name = "vm1"
    vm_memory = 512 * 1024 * 1024
    vm_cluster = api.clusters.get(name="Default")
    vm_template = api.templates.get(name="Blank")
    vm_os = params.OperatingSystem(boot=[params.Boot(dev="hd")])

    vm_params = params.VM(name=vm_name,
                          memory=vm_memory,
                          cluster=vm_cluster,
                          template=vm_template,
                          os=vm_os)

    try:
        api.vms.add(vm=vm_params)
        print "Virtual machine '%s' added." % vm_name
    except Exception as ex:
        print "Adding virtual machine '%s' failed: %s" % (vm_name, ex)

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** request is successful then the script will output:

```
Virtual machine 'vm1' added.
```

2.14. EXAMPLE: CREATING A VIRTUAL MACHINE NIC USING PYTHON

To ensure a newly created virtual machine has network access you must create and attach a virtual NIC.

Example 2.13. Creating a virtual machine NIC using Python

This Python example creates an NIC named **nic1** and attaches it to the virtual machine named **vm1**. The NIC in this example:

- Must be a **virtio** network device.

  ```
  nic_interface = "virtio"
  ```

- Must be linked to the **ovirtmgmt** management network.

  ```
  nic_network = api.networks.get(name="ovirtmgmt")
  ```

These options are combined into an NIC parameter object, before using the **add** method of the virtual machine's **nics** collection to create the NIC.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params
```

```
try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    vm = api.vms.get(name="vm1")

    nic_name = "nic1"
    nic_interface = "virtio"
    nic_network = api.networks.get(name="ovirtmgmt")

    nic_params = params.NIC(name=nic_name, interface=nic_interface,
network=nic_network)

    try:
        nic = vm.nics.add(nic_params)
        print "Network interface '%s' added to '%s'." %
(nic.get_name(), vm.get_name())
    except Exception as ex:
        print "Adding network interface to '%s' failed: %s" %
(vm.get_name(), ex)

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** request is successful then the script will output:

```
Network interface 'nic1' added to 'vm1'.
```

2.15. EXAMPLE: CREATING A VIRTUAL MACHINE STORAGE DISK USING PYTHON

To ensure a newly created virtual machine has access to persistent storage you must create and attach a disk.

Example 2.14. Creating a virtual machine storage disk using Python

This Python example creates an 8 GB **virtio** disk drive and attaches it to the virtual machine named **vm1**. The disk in this example:

- must be stored on the storage domain named **data1**,

```
disk_storage_domain = params.StorageDomains(storage_domain=
[api.storagedomains.get(name="data1")])
```

- must be 8 GB in size,

```
disk_size = 8*1024*1024
```

- must be a **system** type disk (as opposed to **data**),

```
disk_type = "system"
```

- must be **virtio** storage device,

```
disk_interface = "virtio"
```

- must be stored in **cow** format, and

```
disk_format = "cow"
```

- must be marked as a usable boot device.

```
disk_bootable = True
```

These options are combined into a disk parameter object, before using the **add** method of the virtual machine's **disks** collection to create the disk itself.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    vm = api.vms.get(name="vm1")

    sd = params.StorageDomains(storage_domain=
[api.storagedomains.get(name="data1")])
    disk_size = 8*1024*1024
    disk_type = "system"
    disk_interface = "virtio"
    disk_format = "cow"
    disk_bootable = True

    disk_params = params.Disk(storage_domains=sd,
                              size=disk_size,
                              type_=disk_type,
                              interface=disk_interface,
                              format=disk_format,
                              bootable=disk_bootable)

    try:
        d = vm.disks.add(disk_params)
        print "Disk '%s' added to '%s'." % (d.get_name(),
vm.get_name())
    except Exception as ex:
        print "Adding disk to '%s' failed: %s" % (vm.get_name(), ex)

    api.disconnect()
```

```
except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** request is successful then the script will output:

```
Disk 'vm1_Disk1' added to 'vm1'.
```

2.16. EXAMPLE: ATTACHING AN ISO IMAGE TO A VIRTUAL MACHINE USING PYTHON

To begin installing a guest operating system on a newly created virtual machine you must attach an ISO file containing the operating system installation media.

Example 2.15. Identifying ISO images

ISO images are found in the **files** collection attached to the ISO storage domain. This example lists the contents of the **files** collection on an ISO storage domain.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS",
              ca_file="ca.crt")

    sd = api.storagedomains.get(name="iso1")
    iso = sd.files.list()

    for i in iso:
        print "%s" % i.get_name()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If successful the script will output an entry like this for each file found in the **files** collection:

```
RHEL6.3-Server-x86_64-DVD1.iso
```

Note that because files on the ISO domain must be uniquely named the **id** and **name** attributes of the file are shared.

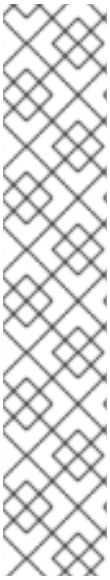
Example 2.16. Attaching an ISO image to a virtual machine using Python

This Python example attaches the **RHEL6.3-Server-x86_64-DVD1.iso** ISO image file to the **vm1** virtual machine. Once identified the image file is attached using the **add** method of the virtual machine's **cdroms** collection.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS,
              ca_file="ca.crt")

    sd = api.storagedomains.get(name="iso1")

    cd_iso = sd.files.get(name="RHEL6.3-Server-x86_64-DVD1.iso")
    cd_vm = api.vms.get(name="vm1")
    cd_params = params.CdRom(file=cd_iso)

    try:
        cd_vm.cdroms.add(cd_params)
        print "Attached CD to '%s'." % cd_vm.get_name()
    except Exception as ex:
        print "Failed to attach CD to '%s': %s" % (cd_vm.get_name(),
ex)

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **add** request is successful then the script will output:

```
Attached CD to 'vm1'.
```



NOTE

This procedure is for attaching an ISO image to virtual machines with a status of **Down**. To attach an ISO to a virtual machine with an **Up** status, amend the second **try** statement to the following:

```
try:
 cdrom=cd_vm.cdroms.get(id="00000000-0000-0000-0000-
000000000000")
 cdrom.set_file(cd_iso)
 cdrom.update(current=True)
 print "Attached CD to '%s'." % cd_vm.get_name()
except:
 print "Failed to attach CD to '%s': %s" % (cd_vm.get_name(),
ex)
```

Example 2.17. Ejecting a cdrom from a Virtual Machine using Python

Eject an ISO from a virtual machine's **cdrom** collection.

```
from ovirtsdk.api import API
```

```
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS,
              ca_file="ca.crt")

    sd = api.storagedomains.get(name="iso1")
    vm = api.vms.get(name="vm1")

    try:
        vm.cdroms.get(id="00000000-0000-0000-0000-
000000000000").delete()
        print "Removed CD from '%s'." % vm.get_name()
    except Exception as ex:
        print "Failed to remove CD from '%s': %s" % (vm.get_name(), ex)

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

2.17. EXAMPLE: DETACHING A DISK USING PYTHON

You can use the Python software development kit to detach a virtual disk from a virtual machine.

Example 2.18. Detaching a disk using Python

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              username="USER@DOMAIN",
              password="PASS",
              ca_file="ca.crt")

    vm = api.vms.get(name="VM_NAME")
    disk = vm.disks.get(name="DISK_NAME")

    detach = params.Action(detach=True)
    disk.delete(action=detach)

    print "Detached disk %s successfully!" % disk

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

2.18. EXAMPLE: STARTING A VIRTUAL MACHINE USING PYTHON

Starting a virtual machine

Example 2.19. Starting a virtual machine using Python

This example starts the virtual machine using the **start** method.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    vm = api.vms.get(name="vm1")

    try:
        vm.start()
        print "Started '%s'." % vm.get_name()
    except Exception as ex:
        print "Unable to start '%s': %s" % (vm.get_name(), ex)

    api.disconnect()

except Exception as ex:
    print "Unexpected error: %s" % ex
```

If the **start** request is successful then the script will output:

```
Started 'vm1'.
```

Note that the **status** reflects that the virtual machine has been started and is now **up**.

2.19. EXAMPLE: STARTING A VIRTUAL MACHINE WITH OVERRIDDEN PARAMETERS USING PYTHON

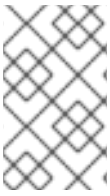
Starting a virtual machine with overridden parameters.

Example 2.20. Starting a virtual machine with overridden parameters using Python

This example boots a virtual machine with a Windows ISO and attaches the **virtio-win_x86.vfd** floppy disk which contains Windows drivers. This action is equivalent to using the Run Once window in the Administration or User Portal to start a virtual machine.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
```

```
                username="USER@DOMAIN",
                password="PASS",
                ca_file="ca.crt")
except Exception as ex:
    print "Failed to connect to API: %s" % ex


try:
    vm = api.vms.get(name="Win_machine")
except Exception as ex:
    print "Failed to retrieve VM: %s" % ex

cdrom = params.CdRom(file=params.File(id="windows_example.iso"))
floppy = params.Floppy(file=params.File(id="virtio-win_x86.vfd"))
try:
    vm.start(
      action=params.Action(
        vm=params.VM(
          os=params.OperatingSystem(
            boot=[params.Boot(dev="cdrom")]
          ),
          cdroms=params.CdRoms(cdrom=[cdrom]),
          floppies=params.Floppies(floppy=[floppy])
        )
      )
    )
except Exception as ex:
    print "Failed to start VM: %s" % ex
```



NOTE

The CD image and floppy disk file must be available in the ISO domain already. If not, use the ISO uploader tool to upload the files. See The ISO Uploader Tool for more information.

2.20. EXAMPLE: STARTING A VIRTUAL MACHINE WITH CLOUD-INIT USING PYTHON

Starting a virtual machine with Cloud-Init using Python.

Example 2.21. Starting a virtual machine with Cloud-Init using Python

This example shows you how to start a virtual machine using the Cloud-Init tool to set a host name and a static IP for the eth0 interface.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
                username="USER@DOMAIN",
                password="PASS",
                ca_file="ca.crt")
```

```
except Exception as ex:
    print "Failed to connect to API: %s" % ex


try:
    vm = api.vms.get(name="MyVM")
except Exception as ex:
    print "Failed to retrieve VM: %s" % ex

try:
    vm.start(
  use_cloud_init=True,
      action=params.Action(
        vm=params.VM(
          initialization=params.Initialization(
            cloud_init=params.CloudInit(
              host=params.Host(address="MyHost.example.com"),
          network_configuration=params.NetworkConfiguration(
            nics=params.Nics(
              nic=[params.NIC(
                name="eth0",
                boot_protocol="static",
                on_boot=True,
                network=params.Network(
                  ip=params.IP(
                    address="10.10.10.1",
                    netmask="255.255.255.0",
                    gateway="10.10.10.1"
                              )
                            )
                          )
                        ]
                      )
                    )
                  )
                )
              )
            )
except Exception as ex:
    print "Failed to start VM: %s" % ex
```

2.21. EXAMPLE: CHECKING SYSTEM EVENTS USING PYTHON

Red Hat Virtualization Manager records and logs many system events. These event logs are accessible through the user interface, the system log files, and using the API. The ovirtsdk library exposes events using the **events** collection.

Example 2.22. Checking System Events using Python

In this example the **events** collection is listed. Note that:

- The **query** parameter of the **list** method is used to ensure that all available pages of results are returned. By default the **list** method will only return the first page of results which defaults to a maximum of **100** records in length.

which defaults to a maximum of **100** records in length.

- The resultant list is reversed to ensure that events are included in the output in the order that they occurred.

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API (url="https://HOST",
               username="USER@DOMAIN",
               password="PASS",
               ca_file="ca.crt")

    event_list = []
    event_page_index = 1
    event_page_current = api.events.list(query="page %s" %
event_page_index)

    while(len(event_page_current) != 0):
        event_list = event_list + event_page_current
        event_page_index = event_page_index + 1
  try:
            event_page_current = api.events.list(query="page %s" %
event_page_index)
     except Exception as ex:
      print "Error retrieving page %s of list: %s" % (event_page_index,
ex)

    event_list.reverse()

    for event in event_list:
        print "%s %s CODE %s - %s" % (event.get_time(),
                                      event.get_severity().upper(),
                                      event.get_code(),
                                      event.get_description())

except Exception as ex:
    print "Unexpected error: %s" % ex
```

Output from this script will look like this - albeit with different events depending on the state of the environment:

```
2012-09-25T18:40:10.065-04:00 NORMAL CODE 30 - User admin@internal
logged in.
2012-09-25T18:40:10.368-04:00 NORMAL CODE 153 - VM vm1 was started by
admin@internal (Host: Atlantic).
2012-09-25T18:40:10.470-04:00 NORMAL CODE 30 - User admin@internal
logged in.
```

CHAPTER 3. USING THE SOFTWARE DEVELOPMENT KIT

3.1. CONNECTING TO THE API USING PYTHON

To connect to the REST API using Python you must create an instance of the **API** class from the ovirtsdk.api module. To be able to do this it is necessary to first import the class at the start of the script:

```
from ovirtsdk.api import API
```

The constructor of the **API** class takes a number of arguments. Supported arguments are:

url

Specifies the URL of the Manager to connect to, including the ***/api*** path. This parameter is mandatory.

username

Specifies the user name to connect using, in User Principal Name (UPN) format. This parameter is mandatory.

password

Specifies the password for the user name provided by the ***username*** parameter. This parameter is mandatory.

kerberos

Uses a valid Kerberos ticket to authenticate the connection. Valid values are **True** and **False**. This parameter is optional.

key_file

Specifies a PEM formatted key file containing the private key associated with the certificate specified by ***cert_file***. This parameter is optional.

cert_file

Specifies a PEM formatted client certificate to be used for establishing the identity of the client on the server. This parameter is optional.

ca_file

Specifies the certificate file of the certificate authority for the server. This parameter is mandatory unless the ***insecure*** parameter is set to **True**.

port

Specifies the port to connect using, where it has not been provided as component of the ***url*** parameter. This parameter is optional.

timeout

Specifies the amount of time in seconds that is allowed to pass before a request is to be considered as having timed out. This parameter is optional.

persistent_auth

Specifies whether persistent authentication is enabled for this connection. Valid values are **`True`** and **`False`**. This parameter is optional and defaults to **`False`**.

insecure

Allows a connection via SSL without certificate authority. Valid values are **`True`** and **`False`**. If the ***`insecure`*** parameter is set to **`False`** - which is the default - then the ***`ca_file`*** must be supplied to secure the connection.

This option should be used with caution, as it may allow man-in-the-middle (MITM) attackers to spoof the identity of the server.

filter

Specifies whether or not user permission based filter is on or off. Valid values are **`True`** and **`False`**. If the ***`filter`*** parameter is set to **`False`** - which is the default - then the authentication credentials provided must be those of an administrative user. If the ***`filter`*** parameter is set to **`True`** then any user can be used and the Manager will filter the actions available to the user based on their permissions.

debug

Specifies whether debug mode is enabled for this connection. Valid values are **`True`** and **`False`**. This parameter is optional.

You can communicate with multiple Red Hat Virtualization Managers by creating and manipulating separate instances of the ovirtsdk.API Python class.

This example script creates an instance of the **`API`** class, checks that the connection is working using the **`test()`** method, and disconnects using the **`disconnect()`** method.

```
from ovirtsdk.api import API

api_instance = API ( url="https://rhevm31.demo.redhat.com",
                     username="admin@internal",
                     password="Password",
                     ca_file="/etc/pki/ovirt-engine/ca.pem")

print "Connected successfully!"

api_instance.disconnect()
```

For a full list of methods supported by the **`API`** class refer to the pydoc output for the ovirtsdk.api module.

```
$ pydoc ovirtsdk.api
```

3.2. RESOURCES AND COLLECTIONS

The RESTful nature of the API is evident throughout the Python bindings for both theoretical and practical reasons. All RESTful APIs have two key concepts that you need to be aware of:

Collections

A collection is a set of resources of the same type. The API provides both top-level collections and sub-collections. An example of a top-level collection is the **`hosts`** collection which contains all

virtualization hosts in the environment. An example of a sub-collection is the **host.nics** collection which contains resources for all network interface cards attached to a host resource.

The interface for interacting with collections provides methods for adding resources (**add**), getting resources (**get**), and listing resources (**list**).

Resources

A resource in a RESTful API is an object with a fixed interface that also contains a set of attributes that are relevant to the specific type of resource being represented. The interface for interacting with resources provides methods for updating (**update**) and deleting (**delete**) resources. Additionally some resources support actions specific to the resource type. An example is the **approve** method of **Host** resources.

3.3. RETRIEVING RESOURCES FROM A COLLECTION

Resources are retrieved from a collection using the **get** and **list** methods.

get

Retrieves a single resource from the collection. The item to retrieve is determined based on the name provided as an argument. The **get** method takes these arguments:

- **name** - The name of the resource to retrieve from the collection.
- **id** - The globally unique identifier (GUID) of the resource to retrieve from the collection.

list

Retrieves any number of resources from the collection. The items to retrieve are determined based on the criteria provided. The **list** method takes these arguments:

- ****kwargs** - A dictionary of additional arguments allowing keyword based filtering.
- **query** - A query written in the same format as that used for searches executed using the Red Hat Virtualization user interfaces.
- **max** - The maximum number of resources to retrieve.
- **case_sensitive** - Whether or not search terms are to be treated as case sensitive (**True** or **False**, the default is **True**).

3.4. RETRIEVING A SPECIFIC RESOURCE FROM A COLLECTION

In these examples a specific resource is retrieved from a collection using the **get** method.

Example 3.1. Retrieving a Specific Resource by Name

Retrieving the **Default** data center from the **datacenters** collection using the ***name*** parameter of the **get** method:

```
dc = api.datacenters.get("Default")
```

This syntax is equivalent:

```
dc = api.datacenters.get(name="Default")
```

Additional information can be retrieved for **get** requests using the **all_content** header.

Example 3.2. Retrieving Additional Information on a Specific Resource

```
vm = api.vms.get(name="VM01", all_content=True)
```

3.5. RETRIEVING A LIST OF RESOURCES FROM A COLLECTION

In these examples a list of resources is retrieved from a collection using the **list** method.

Example 3.3. Retrieving a List of all Resources in a Collection

Retrieving a list of all resources in the **datacenters** collection. The ***query*** parameter of the **list** method allows the use of engine based queries. In this way the SDK supports the use of queries in the same format as those executed in the Administration and User Portals. The ***query*** parameter is also the mechanism for providing pagination arguments while iterating through the collection.

```
dc_list = []
dc_page_index = 1
dc_page_current = api.datacenters.list(query="page %s" % dc_page_index)
while(len(dc_page_current) != 0):
    dc_list = dc_list + dc_page_current
    dc_page_index = dc_page_index + 1
    dc_page_current = api.datacenters.list(query="page %s" %
dc_page_index)
```

In this example the list of resources contained in the **datacenters** collection is ultimately stored in the locally defined **dc_list** list variable.

WARNING

The **list** method of a collection is restricted to returning only as many elements as allowed by the **SearchResultsLimit** Red Hat Virtualization Manager configuration key.

To ensure that all records in a the **list** are returned it is recommended that you paginate through the results as illustrated in this example.

Alternatively you may choose to set the ***max*** parameter of the **list** method to the maximum number of records that you wish to retrieve.

Example 3.4. Retrieving a List of Resources in a Collection Matching a Keyword Based Filter

Retrieving a list of all resources in the **datacenters** collection that have a storage type of **nfs**. In this example both the ***query*** parameter and *****kwargs*** parameter are supplied. The ***query*** is used for pagination in the same way as illustrated in the previous example. The *****kwargs*** parameter is used to filter based on the storage type of the data center.

```
dc_list = []
dc_page_index = 1
dc_page_current = api.datacenters.list(query="page %s" % dc_page_index,
**{"storage_type": "nfs"})
while(len(dc_page_current) != 0):
    dc_list = dc_list + dc_page_current
    dc_page_index = dc_page_index + 1
    dc_page_current = api.datacenters.list(query="page %s" %
dc_page_index, **{"storage_type": "nfs"})
```

In this example the list of resources contained in the **datacenters** collection with a storage type of **nfs** is ultimately stored in the locally defined **dc_list** list variable.

3.6. ADDING A RESOURCE TO A COLLECTION

The **add** method of a collection adds a resource. The resource to be added is created based on the parameters provided. Parameters are provided to the **add** method using an instance of an object from the ovirtsdk.xml.params module. Which specific class from the module needs to be used varies based on the type of resource being created.

Example 3.5. Adding a Resource to a Collection

In this example a virtual machine resource is created.

```
vm_params = params.VM(name="DemoVM",
                      cluster=api.clusters.get("Default"),
                      template=api.templates.get("Blank"),
                      memory=536870912)
vm = api.vms.add(vm_params)
```

While the virtual machine created by this example is not yet ready to run it illustrates the process for creating any Red Hat Virtualization resource:

- Create an instance of the parameter object for the type of resource being created.
- Identify the collection to which the resource will be added.
- Call the **add** method of the collection passing the parameter object as a parameter.

Some parameter objects also have complex parameters of their own.

Example 3.6. Complex Parameters

In this example an NFS data center running in full version 4.0 compatibility mode is being created. To

do this it is necessary to first construct a **ovirtsdk.xml.params.Version** object. Then this is used as a parameter when creating an instance of a **ovirtsdk.xml.params.DataCenter** object containing parameters of the data center to be created. The resource is then created using the **add** method of the **datacenters** collection.

```
v_params = params.Version(major=4, minor=0)
dc_params = params.DataCenter(name="DemoDataCenter", storage_type="NFS",
version=v_params)
dc = api.datacenters.add(dc_params)
```

3.7. UPDATING A RESOURCE IN A COLLECTION

To update a resource you must retrieve it from the collection it resides in, modify the desired parameters, and then call the **update** method for the resource to save the changes. Parameter modification is performed by using the **set_*** methods of the retrieved resource.

Example 3.7. Updating a Resource

In this example the data center named **DemoDataCenter** has its description updated.

```
dc = api.datacenters.get("DemoDataCenter")
dc.set_description("This data center description provided using the
Python SDK")
dc.update()
```

3.8. REMOVING A RESOURCE FROM A COLLECTION

To remove a resource you must retrieve it from the collection that contains it and call the **delete** method of the resource.

Example 3.8. Removing a Resource from a Collection

Deleting a virtual machine named **DemoVM** from the **vms** collection:

```
vm = api.vms.get("DemoVM")
vm.delete()
```

3.9. HANDLING ERRORS

Where errors are encountered the Software Development Kit uses exceptions to highlight them. The Software Development Kit defines exception types in addition to those defined by the Python interpreter itself. These exceptions are located in the **ovirtsdk.infrastructure.errors** module:

ConnectionError

Raised when a transport layer error has occurred.

DisconnectedError

Raised when attempting to use SDK after it was explicitly disconnected.

ImmutableError

Raised when initiating SDK while an SDK instance already exists under the same domain. Applicable to SDK version 3.2 and higher.

NoCertificatesError

Raised when no CA is provided and --insecure is 'False'.

RequestError

Raised at any kind of oVirt server error.

UnsecuredConnectionAttemptError

Raised when HTTP protocol is used while server is running HTTPS.

MissingParametersError

Raised when you are trying to use get() method without providing either id or name.

These exceptions can be caught and handled like any other Python exception:

Example 3.9. Catching a `ConnectionError` Exception

```
from ovirtsdk.api import API
from ovirtsdk.xml import params

try:
    api = API(url="https://HOST",
              user="USER,
              pass="PASS,
              ca_file="/etc/pki/ovirt-engine/ca.pem")
except ConnectionError, err:
    print "Connection failed: %s" % err
```

CHAPTER 4. PYTHON REFERENCE DOCUMENTATION

4.1. PYTHON REFERENCE DOCUMENTATION

Documentation generated using pydoc is available for the following modules. The documentation is provided by the ovirt-engine-sdk-python package.

- ovirtsdk.api
- ovirtsdk.infrastructure.brokers
- ovirtsdk.infrastructure.errors

Run the following command on the machine on which the Red Hat Virtualization Manager is installed to view the latest version of these documents:

```
$ pydoc [MODULE]
```