



Red Hat Virtualization

4.1

Python SDK 指南

使用 Red Hat Virtualization Python SDK

Red Hat Virtualization 文档团队Red Hat

使用 Red Hat Virtualization Python SDK

Red Hat Virtualization 文档团队
Red Hat 出版部
rhev-docs@redhat.com

法律通告

Copyright © 2017 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本指南提供了如何安装并使用 Red Hat Virtualization Python 软件开发套件 (Python SDK) 版本 3 和版本 4 的信息。

目录

第 1 章 概述	3
1.1. 前提条件	3
1.2. 安装 PYTHON 软件开发套件	3
第 2 章 PYTHON 快速入门示例	5
2.1. PYTHON 快速入门介绍	5
2.2. 示例：使用 PYTHON 访问 API 的进入点	5
2.3. 示例：使用 PYTHON 列出数据中心集合	6
2.4. 示例：使用 PYTHON 列出集群集合	7
2.5. 示例：使用 PYTHON 列出逻辑网络集合	8
2.6. 示例：使用 PYTHON 列出主机集合	8
2.7. 示例：列出 ISO 存储域中的 ISO 文件	9
2.8. 示例：列出虚拟机的大小	10
2.9. 示例：使用 PYTHON 创建 NFS 数据存储	10
2.10. 示例：使用 PYTHON 创建 NFS ISO 存储	12
2.11. 示例：使用 PYTHON 为数据中心附加存储域	13
2.12. 示例：使用 PYTHON 激活存储域	15
2.13. 示例：使用 PYTHON 创建虚拟机	16
2.14. 示例：使用 PYTHON 创建虚拟机 NIC	17
2.15. 示例：使用 PYTHON 创建虚拟机存储磁盘	18
2.16. 示例：使用 PYTHON 为虚拟机附加一个 ISO 镜像	19
2.17. 示例：使用 PYTHON 分离一个磁盘	22
2.18. 示例：使用 PYTHON 启动虚拟机	22
2.19. 示例：使用 PYTHON 启动覆盖参数的虚拟机	23
2.20. 示例：使用 PYTHON 启动使用 CLOUD-INIT 的虚拟机	24
2.21. 示例：使用 PYTHON 检查系统事件	25
第 3 章 使用软件开发套件	27
3.1. 使用 PYTHON 连接 API	27
3.2. 资源和集合	28
3.3. 从集合中获取资源	29
3.4. 从集合中获取特定资源	29
3.5. 从集合中获取资源列表	30
3.6. 为集合添加一个资源	31
3.7. 更新集合中的资源	31
3.8. 从集合中删除资源	32
3.9. 错误处理	32
第 4 章 PYTHON 参考文档	34
4.1. PYTHON 参考文档	34

第 1 章 概述

Python 软件开发套件就是一组类（class）的集合，可以让您在基于 Python 的项目上使用它们与 Red Hat Virtualization Manager 进行互动。下载这些类，并将其添加到您的项目中，您就可以使用一系列的功能自动完成高级别的管理任务。

Red Hat Virtualization 提供了两个版本的 Python 软件开发套件：

V 3

V3 Python 软件开发套件提供了一组和 Red Hat Enterprise Virtualization 3.6 最新版本中提供的 Python 软件开发套件兼容的类和方法结构。使用 Red Hat Enterprise Virtualization 3.6 的 Python 软件开发套件开发的应用程序可以在不经过修改的情况下在这个版本中运行。

V 4

V4 Python 软件开发套件提供了一组更新的类、方法名和签名。使用 Red Hat Enterprise Virtualization 3.6 的 Python 软件开发套件开发的应用程序需要经过更新后才可以在这个版本中运行。

通过安装相应的软件包，并把所需程序库添加到您的 Python 项目，就可以在 Red Hat Virtualization 环境中使用以上版本的 Python 软件开发套件。

1.1. 前提条件

要安装软件开发套件，您必须有：

- ✱ 安装了 Red Hat Enterprise Linux 7 的系统。Server 和 Workstation 版本都被支持。
- ✱ Red Hat Virtualization 权利的订阅。



重要

该软件开发套件是 Red Hat Virtualization REST AP 的接口，因此必须使用与 Red Hat Virtualization 环境对应的软件开发套件版本。例如：如果使用 Red Hat Virtualization 4.1，则必须使用为 4.1 版本设计的软件开发套件版本。

1.2. 安装 PYTHON 软件开发套件

安装 Python 软件开发套件。

安装 Python 软件开发套件 (SDK)

1. 启用所需的频道：

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-rhv-4.1-rpms
```

2. 安装所需软件包：

- a. V3：

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

b. V4 :

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

将 Python 软件开发套件及随附文档下载至 **/usr/lib/python2.7/site-packages/ovirtsdk/** 目录，现在可将其添加至 Python 项目。

第 2 章 PYTHON 快速入门示例

2.1. PYTHON 快速入门介绍

本章提供了一组示例，它们展示了使用 Python SDK 在一个基本 Red Hat Virtualization 环境中创建虚拟机的步骤。



重要

本章中所使用的示例是基于 V3 Python SDK 的。

这些示例使用 **ovirt-engine-sdk-python** 软件包所提供的 **ovirtsdk** Python 库。通过 Red Hat Subscription Manager 订阅 **Red Hat Virtualization** 权利池的系统可以获得这个软件包。如需了解更多关于通过订阅系统来下载软件的信息，请参阅 [第 1.2 节 “安装 Python 软件开发套件”](#)。

您还需要：

- ✧ 已经连接到网络中的 Red Hat Virtualization Manager。
- ✧ 一个已经被配置并连接到网络中的 Red Hat Virtualization Host。
- ✧ 包括需要在虚拟机上安装的操作系统的 ISO 镜像文件。
- ✧ 对组成 Red Hat Virtualization 环境的物理和逻辑项有一定的了解。
- ✧ 对 Python 编程语言有一定的了解。



重要

所有 Python 示例中都包括了代表用户验证信息的占位符（*USER* 代表用户名，*PASS* 代表密码）。请确认 Python 代码中的相应信息满足您所在环境的用户验证要求。



注意

Red Hat Virtualization Manager 会为每个资源的 **id** 属性生成一个 GUID。这里所使用的值可能会和您实际使用的 Red Hat Virtualization 环境中的值不同。



注意

这些 Python 示例只包括基本的异常和错误处理逻辑。如需了解更多关于 SDK 的异常和错误处理信息，请参阅 **ovirtsdk.infrastructure.errors** 模块的 pydoc。

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

2.2. 示例：使用 PYTHON 访问 API 的进入点

ovirtsdk Python 库提供了 **API** 类，它被作为 API 的进入点使用。

例 2.1. 使用 Python 访问 API 的进入点

这个 python 例子连接到 Red Hat Virtualization Manager（位于 `rhev.m.demo.redhat.com`）所提供的一个 REST API 实例（instance）上。如果连接成功，它会创建一个 **API** 类。如果需要断开连接，使用 **API** 类的 `disconnect()` 方法。

在这个示例中，提供给 **API** 类的构造函数的参数包括：

- ✧ 要连接到的 Manager 的 `url`。
- ✧ 用来进行用户验证的用户的 `username`。
- ✧ 用来进行用户验证的用户的 `password`。
- ✧ `ca_file`（到证书的路径）。证书是 Manager 的证书颁发机构所颁发的文件，它可以通过 `https://[engine-fqdn]_ovirt-engine/services/pki-resource?resource=ca-certificate&format=_X509-PEM-CA` 获得。

API 类的构造函数还支持其它参数。这个示例中只包括了必需的参数。

```
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
```

如果连接成功，这个示例会显示以下信息：

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

2.3. 示例：使用 PYTHON 列出数据中心集合

API 类提供了访问数据中心集合（名为 **datacenters**）的功能。这个集合包括了环境中的所有数据中心。

例 2.2. 使用 Python 列出数据中心集合

这个 Python 示例列出了 **datacenters** 集合中的所有数据中心。它同时还显示了集合中的每个数据中心的基本信息。

```
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

```

如果环境中只包括 **Default** 数据中心，而且它还没有被激活，这个示例会输出以下信息：

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

2.4. 示例：使用 PYTHON 列出集群集合

API 类提供了访问集群集合（名为 **clusters**）的功能。这个集合包括了环境中的所有集群。

例 2.3. 使用 Python 列出集群集合

这个 Python 示例列出了 **clusters** 集合中的集群。它同时还显示了集合中的每个集群的基本信息。

```

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

```

如果环境中只包括 **Default** 集群，这个示例会输出以下信息：

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

2.5. 示例：使用 PYTHON 列出逻辑网络集合

API 类提供了访问逻辑网络集合（名为 **networks**）的功能。这个集合包括了环境中的所有逻辑网络。

例 2.4. 使用 Python 列出逻辑网络集合

这个 Python 示例列出了 **networks** 集合中的逻辑网络。它同时还显示了集合中的每个网络的基本信息。

```
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
```

如果环境中只包括默认的管理网络，这个示例会输出以下信息：

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

2.6. 示例：使用 PYTHON 列出主机集合

API 类提供了访问主机集合（名为 **hosts**）的功能。这个集合包括了环境中的所有主机。

例 2.5. 使用 Python 列出主机集合

这个 Python 示例列出了 **hosts** 集合中的主机。

```
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

```

如果一个环境中只附加了一个名为 **Atlantic** 的主机，这个示例会输出以下信息：

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

2.7. 示例：列出 ISO 存储域中的 ISO 文件

API 类提供了访问存储域集合（名为 **storagedomains**）的功能。这个集合包括了一个 **files** 集合来描述存储域中的文件。

例 2.6. 列出 ISO 存储域中的 ISO 文件

这个 Python 示例输出 Red Hat Virtualization 环境中的每个 ISO 存储域中的 ISO 文件列表。

```

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. 示例：列出虚拟机的大小

API 类提供了访问虚拟机集合（名为 **vms**）的功能。这个集合包括了一个 **disks** 集合来描述附加到这个虚拟机上的每个磁盘的详细信息。

例 2.7. 列出虚拟机的大小

这个 Python 示例输出了 Red Hat Virtualization 环境中的虚拟机列表，以及它们的磁盘容量总和（以字节为单位）：

```

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. 示例：使用 PYTHON 创建 NFS 数据存储

当 Red Hat Virtualization 环境被第一次创建时，它需要定义最少一个数据存储域和一个 ISO 存储域。数据存储域被用来存储虚拟磁盘镜像；ISO 存储域被用来存储在虚拟机上安装操作系统所需的安装介质。

API 类提供了访问存储域集合（名为 **storagedomains**）的功能。这个集合包括环境中的所有存储域。**storagedomains** 集合也可以被用来添加和删除存储域。



注意

这个示例中所提供的代码假设，远程 NFS 共享已经为在 Red Hat Virtualization 中使用进行了预配置。请参阅 Red Hat Virtualization *管理指南* 来获得关于准备 NFS 共享的信息。

例 2.8. 使用 Python 创建 NFS 数据存储

这个 Python 示例在 **storagedomains** 集合中添加了一个 NFS 数据域。使用 Python 添加 NFS 存储域可以分为以下几步：

1. 使用 **datacenters** 集合的 **get** 方法指定存储必须被附加到的数据中心。

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

2. 使用 **hosts** 集合的 **get** 方法指定用来附加存储的主机。

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

3. 指定 NFS 存储域的 **Storage** 参数。在这个示例中，NFS 的位置是 **192.0.43.10/storage/data**。

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

4. 使用 **storagedomains** 集合的 **add** 方法请求创建存储域。除了 **Storage** 参数，还需要提供以下信息：

- ✧ 存储域的名称。
- ✧ 从 **datacenters** 集合获得的数据中心项。
- ✧ 从 **hosts** 集合获得的主机项。
- ✧ 添加的存储域类型（**data**、**iso** 或 **export**）。
- ✧ 使用的存储格式（**v1**、**v2** 或 **v3**）。

以上的步骤被组合为：

```
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

```

如果 **add** 方法调用成功，脚本会输出以下信息：

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

2.10. 示例：使用 PYTHON 创建 NFS ISO 存储

要创建虚拟机，则需要为虚拟机操作系统提供安装介质。可在 Red Hat Virtualization 环境中将安装介质保存在 ISO 存储域中。



注意

这个示例中所提供的代码假设，远程 NFS 共享已经为在 Red Hat Virtualization 中使用进行了预配置。请参阅 Red Hat Virtualization *管理指南* 来获得关于准备 NFS 共享的信息。

例 2.9. 使用 Python 创建 NFS ISO 存储

这个 Python 示例在 **storagedomains** 集合中添加了一个 NFS ISO 域。使用 Python 添加 NFS 存储域可以分为以下几步：

1. 使用 **datacenters** 集合的 **get** 方法指定存储必须被附加到的数据中心。

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

2. 使用 **hosts** 集合的 **get** 方法指定用来附加存储的主机。

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


3. 指定 NFS 存储域的 **Storage** 参数。在这个示例中，NFS 的位置是 **192.0.43.10/storage/iso**。

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

4. 使用 **storagedomains** 集合的 **add** 方法请求创建存储域。除了 **Storage** 参数，还需要提供以下信息：

- ✎ 存储域的名称。
- ✎ 从 **datacenters** 集合获得的数据中心项。
- ✎ 从 **hosts** 集合获得的主机项。
- ✎ 添加的存储域类型 (**data**、**iso** 或 **export**) 。
- ✎ 使用的存储格式 (**v1**、**v2** 或 **v3**) 。

以上的步骤被组合为：

```
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
```

如果 **add** 方法调用成功，脚本会输出以下信息：

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

2.11. 示例：使用 PYTHON 为数据中心附加存储域

在把存储域添加到 Red Hat Virtualization 后，需要把它们附加到数据中心并进行激活，然后才可以被使用。

例 2.10. 使用 Python 为数据中心附加存储域

这个 Python 示例把一个名为 **data1** 的数据存储域和一个名为 **iso1** 的 ISO 存储域附加到 **default** 数据中心中。这个附加的操作是通过数据中心的 **storagedomains** 集合的 **add** 方法实现的。

```
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
```

如果 **add** 方法的调用成功，这个脚本会输出：

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

请注意：**status** 的值代表了存储域仍然需要被激活。

2.12. 示例：使用 PYTHON 激活存储域

一旦把存储域添加到 Red Hat Virtualization，并把它们附加到一个数据中心，您需要把它们激活后才可以使⽤。

例 2.11. 使用 Python 激活存储域

这个 Python 示例激活了一个名为 **data1** 的数据存储域和一个名为 **iso1** 的 ISO 存储域。这两个存储域都被附加到 **Default** 数据中心中。激活操作是通过存储域的 **activate** 方法实现的。

```
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
```

如果 **activate** 请求成功，脚本会输出：

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

请注意：**status** 的值代表了存储域已经被激活。

2.13. 示例：使用 PYTHON 创建虚拟机

创建虚拟机需要几个步骤。这里介绍的第一步将创建虚拟机本身的对象。

例 2.12. 使用 Python 创建虚拟机

这个 Python 示例创建了一个名为 **vm1** 的虚拟机。这个虚拟机会满足以下条件：

- ✧ 最少有 512MB 内存（以字节表示）。

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

- ✧ 必须附加到 **Default** 集群（因此也被附加到 **Default** 数据中心）。

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

- ✧ 必须基于默认的 **Blank** 模板。

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

- ✧ 必须从虚拟硬盘引导。

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

在使用 **vms** 集合的 **add** 方法创建虚拟机前，把这些选项组合为一个参数项。

```
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

```

如果 **add** 请求成功，脚本会输出以下信息：

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

2.14. 示例：使用 PYTHON 创建虚拟机 NIC

要使新创建的虚拟机可以访问网络资源，您需要创建一个虚拟 NIC 并把它附加到虚拟机上。

例 2.13. 使用 Python 创建虚拟机 NIC

这个 Python 示例创建一个名为 **nic1** 的 NIC，并把它附加到名为 **vm1** 的虚拟机上。这个 NIC 满足以下条件：

- ✧ 必须是 **virtio** 网络设备。

```
nic_interface = "virtio"
```

- ✧ 必须连接到 **ovirtmgmt** 管理网络。

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

在使用虚拟机的 **nics** 集合的 **add** 方法创建 NIC 前，把这些选项组合在一起作为 NIC 参数项。

```

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

```

如果 **add** 请求成功，脚本会输出以下信息：

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

2.15. 示例：使用 PYTHON 创建虚拟机存储磁盘

要使新创建的虚拟机可以访问具有持久性的存储，您需要创建并附加一个磁盘。

例 2.14. 使用 Python 创建虚拟机存储磁盘

这个 Python 示例创建一个 8GB **virtio** 磁盘，并把它附加到名为 **vm1** 的虚拟机上。这个磁盘满足以下条件：

- ✧ 必须存储在名为 **data1** 的存储域中，

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

- ✧ 磁盘的容量是 8GB，

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

- ✧ 必须是一个 **system** 类磁盘（和 **data** 相对应），

```
disk_type = "system"
```

- ✧ 必须是 **virtio** 存储设备，

```
disk_interface = "virtio"
```

- ✧ 以 **cow** 格式保存，

```
disk_format = "cow"
```

- 必须被标记为一个可用的引导设备。

```
disk_bootable = True
```

在使用虚拟机的 **disks** 集合的 **add** 方法创建磁盘前，把这些选项组合在一起作为磁盘参数项。

```
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
```

如果 **add** 请求成功，脚本会输出以下信息：

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

2.16. 示例：使用 PYTHON 为虚拟机附加一个 ISO 镜像

在一个新建虚拟机上开始安装操作系统前，需要附加包括了操作系统安装介质的 ISO 镜像文件。

例 2.15. 指定 ISO 镜像

ISO 镜像包括在附加到 ISO 存储域中的 **files** 集合中。这个示例列出了一个 ISO 存储域中的 **files** 集合中的内容。

```

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

```

如果运行成功，脚本会为在 **files** 集合中找到的每个文件输出和以下相似的内容：

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

请注意，因为 ISO 域中的每个文件的名称必须是唯一的，所以文件的 **id** 和 **name** 属性是相同的。

例 2.16. 使用 Python 为虚拟机附加一个 ISO 镜像

这个 Python 示例把 **RHEL6.3-Server-x86_64-DVD1.iso** ISO 镜像文件附加到 **vm1** 虚拟机上。使用虚拟机的 **cdroms** 集合的 **add** 方法可以把指定的镜像文件附加到虚拟机上。

```

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

```

如果 **add** 请求成功，脚本会输出以下信息：

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

注意

上面的示例是把一个 ISO 镜像附加到一个状态为 **Down** 的虚拟机上。如果需要把 ISO 附加到一个状态为 **Up** 的虚拟机上时，把第 2 个 **try** 中的代码改为：

```

try:
    cdrom=cd_vm.cdroms.get(id="_000000000-0000-0000-0000-
00000000000000")
    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)

```

例 2.17. 使用 Python 从虚拟机上弹出一个 cdrom

从虚拟机的 **cdrom** 集合中弹出一个 ISO。

```

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="000000000-0000-0000-0000-
00000000000000").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. 示例：使用 PYTHON 分离一个磁盘

您可以使用 Python 软件开发套件从虚拟机上分离虚拟磁盘。

例 2.18. 使用 Python 分离一个磁盘

```

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. 示例：使用 PYTHON 启动虚拟机

启动一个虚拟机

例 2.19. 使用 Python 启动虚拟机

这个示例使用 **start** 方法启动虚拟机。

```

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

```

如果 **start** 请求成功，脚本会输出以下信息：

```
Started 'vm1'.
```

请注意：**status** 的值是 **up**，它代表了主机已经被启动。

2.19. 示例：使用 PYTHON 启动覆盖参数的虚拟机

启动覆盖参数的虚拟机

例 2.20. 使用 Python 启动覆盖参数的虚拟机

这个示例演示了如何使用 Windows ISO 启动虚拟机，并附加包含 Windows 驱动程序的 **virtio-win_x86.vfd** 软盘。这个动作与使用管理或用户门户网站中的 Run Once 窗口启动虚拟机一致。

```

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

```



注意

CD 镜像和软盘文件需要已存在于 ISO 域中。如果还没有存在于 ISO 域中，使用 ISO 上传工具来它们上传到域中。如需了解更多相关信息，请参阅 [The ISO Uploader Tool](#)。

2.20. 示例：使用 PYTHON 启动使用 CLOUD-INIT 的虚拟机

使用 Python 启动使用 Cloud-Init 的虚拟机。

例 2.21. 使用 Python 启动使用 Cloud-Init 的虚拟机

这个示例演示了如何使用 Cloud-Init 工具启动虚拟机，为 eth0 接口设定主机名和静态 IP。

```

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(
        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. 示例：使用 PYTHON 检查系统事件

Red Hat Virtualization Manager 会记录许多系统事件，这些系统事件可以通过用户接口界面、系统日志文件和 API 进行访问。**ovirtsdk** 库使用 **events** 集合来访问事件。

例 2.22. 使用 Python 检查系统事件

在这个示例中，**events** 集合被列出。请注意：

- ✧ 使用 **list** 方法中的 **query** 参数确保了所有结果页都会被返回。在默认的情况下，**list** 方法只返回结果的第 1 页（这个页默认包括最多 **100** 个结果）。
- ✧ 结果列表中的事件顺序和实际发生的顺序相同。

```

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

```

这个脚本的输出会和以下类似（具体事件内容会根据系统的实际情况有所不同）：

```

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.

```

第 3 章 使用软件开发套件

3.1. 使用 PYTHON 连接 API

要使用 Python 连接到 REST API，您需要从 `ovirtsdk.api` 模块中创建一个 **API** 类的实例 (instance)。要实现这个任务，需要在脚本的开始部分首先导入这个类。

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

API 类的构造函数支持以下参数：

url

指定要连接到的 Manager 的 URL（包括 `/api` 路径）。这个参数是必需的。

username

指定连接时使用的用户名（使用 UPD 格式）。这个参数是必需的。

password

指定 **username** 参数提供的用户的密码。这个参数是必需的。

kerberos

使用有效 Kerberos ticket 验证连接。有效值为 **True** 和 **False**。这个参数是自选的。

key_file

指定一个 PEM 格式的密钥文件。这个密钥文件包括了 **cert_file** 指定的证书的私人密钥。这个参数是可选的。

cert_file

指定一个 PEM 格式的客户端证书。这个证书被用来在服务器上建立客户端的身份。这个选项是可选的。

ca_file

指定服务器证书颁发机构的证书文件。除非 **insecure** 参数被设置为 **True**，这个参数是必需的。

port

指定连接使用的端口（当 **url** 参数没有提供相应信息时有效）。这个参数是可选的。

timeout

一个请求的超时时间（以秒为单位）。这个选项是可选的。

persistent_auth

指定是否为连接建立持久性验证。有效值是 **True** 和 **False**。这个参数是可选的，默认值是 **False**。

insecure

允许通过没有证书颁发机构的 SSL 进行连接。有效值包括 **True** 和 **False**。如果 **insecure** 参数被设置为 **False**（默认的值），必须在连接时提供 **ca_file** 来建立安全的连接。

在使用这个选项时需要格外小心。如果使用不当，可能会被“中间人攻击”所利用来进行服务器身份欺骗。

filter

指定是否启用基于用户权限的过滤器。有效值是 **True** 和 **False**。如果 **filter** 参数被设置为 **False**（默认值），所使用的用户验证信息必须是管理员用户。如果 **filter** 参数被设置为 **True**，则任何用户都可以使用，Manager 会根据用户的权限决定用户可以进行什么操作。

debug

指定这个连接是否启用调试（debug）模式。有效值是 **True** 和 **False**。这个参数是可选的。

您可以通过创建并使用多个 ovirtsdk.API Python 项实例来和多个 Red Hat Virtualization Managers 进行交流。

这个示例脚本创建一个 **API**类的实例，使用 **test()** 方法检查连接是否工作正常，然后使用 **disconnect()** 方法断开连接。

```
from ovirtsdk.api import API

api_instance = API ( url="https://rhev31.demo.redhat.com",
                    username="admin@internal",
                    password="Password",
                    ca_file="/etc/pki/ovirt-engine/ca.pem")

print "Connected successfully!"

api_instance.disconnect()
```

有关 **API** 类所支持的方法的完整列表，请参考 ovirtsdk.api 模块的 pydoc 输出结果。

```
$ pydoc ovirtsdk.api
```

3.2. 资源和集合

所有使用 REST 的 API 都包括两个关键的概念，您需要对它们有所了解：

Collections

集合就是相同类型的一组资源。API 提供了顶级集合和子集合。**hosts** 集合是一个顶级集合的例子，它包括了虚拟环境中的所有虚拟主机。**host.nics** 集合是子集合的一个例子，它包括了附加到一个主机资源上的所有网络接口卡。

集合提供了多个方法来添加资源（**add**）、获得资源（**get**）和列出资源（**list**）。

Resources

REST API 中的资源就是一个带有固定接口的对象，它包括了一组和它所代表的特定资源类型相关的属性。所有资源的接口提供了更新资源 (**update**) 和删除资源 (**delete**) 的方法。另外，一些资源还额外支持只适用于它们的某些操作，如 **Host** 资源有一个 **approve** 方法。

3.3. 从集合中获取资源

使用 **get** 和 **list** 方法可以从集合中获取资源。

get

从集合中获取一个单一资源。要被获取的资源是由参数中所提供的名称决定的。**get** 方法可以使用以下参数：

- ✧ **name** - 从集合中获取的资源名称。
- ✧ **id** - 从集合中获取的资源的 GUID。

list

从集合中获取一组资源。要获得的资源由所提供的条件所决定。**list** 方法支持以下参数：

- ✧ ****kwargs** - 允许进行基于关键字过滤的额外参数字典。
- ✧ **query** - 和 Red Hat Virtualization 用户接口中所使用的查询相同的查询。
- ✧ **max** - 可以获得的资源的最大数量。
- ✧ **case_sensitive** - 指定搜索条件是否区分大小写 (**True** 区分大小写或 **False** 不区分大小写，默认值是 **True**)。

3.4. 从集合中获取特定资源

在这些示例中，使用 **get** 方法从集合中获取特定资源。

例 3.1. 获取特定名称的资源

使用 **get** 方法中的 **name** 参数从 **datacenters** 集合中获取 **Default** 数据中心：

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

它等同于：

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

使用头中包括 **all-content** 的 **get** 请求可以获得的额外信息。

例 3.2. 在特定资源中获取额外信息

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

3.5. 从集合中获取资源列表

在这些示例中，使用 **list** 方法从集合中获取一个资源列表。

例 3.3. 获取一个集合中的所有资源列表

获取 **datacenters** 集合中的所有资源列表。**list** 的 **query** 参数可以允许使用基于引擎的查询。这些查询的格式与管理门户和用户门户中所使用的查询格式完全相同。**query** 参数也提供了设置查询结果分页的功能。

```
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)
```

在这个示例中，**datacenters** 集合中的资源列表被保存在本地定义的 **dc_list** 列表变量中。

警告

集合的 **list** 方法所能获得的结果数量仅由 Red Hat Virtualization Manager 的 **SearchResultsLimit** 配置值决定。

如需 **list** 返回所有资源，您可以使用这个示例中所使用的方法对结果进行分页。

或使用 **list** 中的 **max** 参数来指定您需要获取的结果数量的最大值。

例 3.4. 在一个集合中获取一组和关键字过滤器项匹配的资源列表

获取 **datacenters** 集合中的、存储类型为 **nfs** 的资源列表。在这个示例中，使用了 **query** 参数和 ****kwargs** 参数。**query** 参数被用来进行分页；****kwargs** 参数被用来进行基于存储类型的过滤。

```
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"})
```

在这个示例中，**datacenters** 集合中的、存储类型为 **nfs** 的资源列表被保存在本地定义的 **dc_list** 列表变量中。

3.6. 为集合添加一个资源

使用集合的 **add** 方法以及相应的参数可以为这个集合添加一个资源。提供给 **add** 方法的参数使用 **ovirtsdk.xml.params** 模块中的一个项实例。

例 3.5. 为集合添加一个资源

在这个例子中，一个虚拟机资源被创建。

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

这个例子所创建的虚拟机在现阶段还无法运行，但它展示了创建 Red Hat Virtualization 资源的以下过程：

- ✎ 创建一个参数项实例来代表要被创建的资源类型。
- ✎ 指定资源要被添加到的集合。
- ✎ 调用集合的 **add** 方法，使用操作项作为一个参数。

以下参数项也会带有它们自己的复杂参数。

例 3.6. 复杂参数

在这个示例中，一个运行在版本 4.1 完全兼容模式下的 NFS 数据中心被创建。要实现这个操作，首先需要创建一个 **ovirtsdk.xml.params.Version** 项。然后，在创建 **ovirtsdk.xml.params.DataCenter** 项实例中使用这个项（包括了要被创建的数据中心的参数）。最后，使用 **datacenters** 集合的 **add** 方法创建资源。

```
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. 更新集合中的资源

要更新一个资源，您需要先从集合中获得它，然后修改所需的参数，为资源调用 **update** 方法来保存所做的修改。对参数的修改操作是通过使用所获取的资源的 **set_*** 方法进行的。

例 3.7. 更新资源

在这个示例中，名为 **DemoDataCenter** 的数据中心的描述信息被更新。

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

3.8. 从集合中删除资源

要删除一个资源，您需要先从集合中获得它，然后调用 **delete** 方法来删除它。

例 3.8. 从集合中删除资源

从 **vms** 集合中删除一个名为 **DemoVM** 的虚拟机：

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

3.9. 错误处理

当错误发生时，软件开发套件会使用异常（exception）来标识这些错误。软件开发套件定义了除标准 Python 异常外的额外异常。这些异常位于 **ovirtsdk.infrastructure.errors** 模块中：

ConnectionError

在传输层错误发生时出现。

DisconnectedError

当试图在断开连接后使用 SDK 时出现。

ImmutableError

当一个 SDK 实例已经在相同的域中存在时初始 SDK 会出现这个错误。只适用于 SDK 3.2 或更高版本。

NoCertificatesError

当 `--insecure` 是 'False'，而没有提供 CA 时出现。

RequestError

在有 oVirt 服务器错误时出现

UnsecuredConnectionAttemptError

当服务器运行 HTTPS 而使用 HTTP 时出现。

MissingParametersError

在没有为 `get()` 方法提供 `id` 或 `name` 参数的情况下出现。

这些异常的捕获和解决方法与其它 Python 异常的捕获和解决方法完全相同：

例 3.9. 捕获一个 `ConnectionError` 异常

```
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
```

第 4 章 PYTHON 参考文档

4.1. PYTHON 参考文档

以下模块附带 [pydoc](#) 生成的文档。该文档由 **ovirt-engine-sdk-python** 软件包提供。

- ✿ `ovirtsdk.api`
- ✿ `ovirtsdk.infrastructure.brokers`
- ✿ `ovirtsdk.infrastructure.errors`

在安装 Red Hat Virtualization Manager 的机器上运行以下命令来查看这些文档的最新版本：

```
$ pydoc [MODULE]
```