



# Red Hat Virtualization

## 4.1

### Python SDK ガイド

---

Red Hat Virtualization Python SDK の使用

Red Hat Virtualization Documentation TeamRed Hat



## Red Hat Virtualization Python SDK の使用

Red Hat Virtualization Documentation Team  
Red Hat Customer Content Services  
[rhev-docs@redhat.com](mailto:rhev-docs@redhat.com)

## 法律上の通知

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.

## 概要

本ガイドは、Red Hat Virtualization の Python ソフトウェア開発キットバージョン 3 と 4 のインストール方法および使用方法を説明します。

# 目次

<b>第1章 概要</b>	<b>3</b>
1.1. 前提条件	3
1.2. PYTHON ソフトウェア開発キットのインストール	3
<b>第2章 PYTHON クイックスタートの例</b>	<b>5</b>
2.1. PYTHON のクイックスタート について	5
2.2. 例: PYTHON を使用した API エントリーポイントへのアクセス	6
2.3. 例: PYTHON を使用したデータセンターコレクションの一覧表示	6
2.4. 例: PYTHON を使用したクラスターコレクションの一覧表示	7
2.5. 例: PYTHON を使用した論理ネットワークコレクションの一覧表示	8
2.6. 例: PYTHON を使用したホストコレクションの一覧表示	8
2.7. 例: ISO ストレージドメインの ISO ファイルの表示	9
2.8. 例: 仮想マシンのサイズの表示	10
2.9. 例: PYTHON を使用した NFS データストレージの作成	11
2.10. 例: PYTHON を使用した NFS ISO ストレージの作成	12
2.11. 例: PYTHON を使用したデータセンターへのストレージドメインのアタッチ	14
2.12. 例: PYTHON を使用したストレージドメインのアクティブ化	15
2.13. 例: PYTHON を使用した仮想マシンの作成	16
2.14. 例: PYTHON を使用した仮想マシン NIC の作成	18
2.15. 例: PYTHON を使用した仮想マシンのストレージディスクの作成	19
2.16. 例: PYTHON を使用した仮想マシンへの ISO イメージのアタッチ	20
2.17. 例: PYTHON を使用したディスクのデタッチ	23
2.18. 例: PYTHON を使用した仮想マシンの起動	23
2.19. 例: PYTHON を使用した上書きパラメーターでの仮想マシンの起動	24
2.20. 例: PYTHON を使用した CLOUD-INIT での仮想マシンの起動	25
2.21. 例: PYTHON を使用したシステムイベントのチェック	26
<b>第3章 ソフトウェア開発キットの使用</b>	<b>28</b>
3.1. PYTHON を使用した API への接続	28
3.2. リソースおよびコレクション	29
3.3. コレクションからのリソースの取得	30
3.4. コレクションからの特定のリソースの取得	30
3.5. コレクションからのリソースリストの取得	31
3.6. コレクションへのリソースの追加	32
3.7. コレクション内のリソースの更新	33
3.8. コレクションからのリソースの削除	33
3.9. エラーの処理	34
<b>第4章 PYTHON リファレンスドキュメント</b>	<b>36</b>
4.1. PYTHON リファレンスドキュメント	36



## 第1章 概要

Python ソフトウェア開発キットは、Python ベースプロジェクトで Red Hat Virtualization Manager との対話を可能にするクラスのコレクションです。これらのクラスをダウンロードしてプロジェクトに追加することにより、管理タスクを高度に自動化するさまざまな機能を利用することができます。

Red Hat Virtualization では、Python ソフトウェア開発キットの 2 つのバージョンを提供しています。

### バージョン 3

Python ソフトウェア開発キットバージョン 3 では、Red Hat Enterprise Virtualization 3.6 の最新リリースの時点で Python ソフトウェア開発キットで提供されているクラスとメソッド構造との後方互換性があります。Red Hat Enterprise Virtualization 3.6 の Python ソフトウェア開発キットを使用して記述されたアプリケーションは、修正なしにこのバージョンで 사용할 ことができます。

### バージョン 4

Python ソフトウェア開発キットバージョン 4 では、クラスやメソッド名および署名が更新されています。Red Hat Enterprise Virtualization 3.6 の Python ソフトウェア開発キットで記述されたアプリケーションは更新してからでないと、このバージョンでは使用できません。

Red Hat Virtualization 環境では、適切なパッケージをインストールしたり、Python プロジェクトに必要なライブラリーを追加したりする際に必要であるため、Python ソフトウェア開発キットのどちらのバージョンも使用できます。

## 1.1. 前提条件

Python ソフトウェア開発キットをインストールする場合の前提条件は以下のとおりです。

- ※ Red Hat Enterprise Linux 7 がインストールされたシステム。Server バージョンと Workstation バージョンがサポートされます。
- ※ Red Hat Virtualization エンタイトルメントのサブスクリプション。

### 重要

ソフトウェア開発キットは、Red Hat Virtualization REST API のインターフェースです。このように、Red Hat Virtualization 環境のバージョンに適したソフトウェア開発キットのバージョンを使用する必要があります。たとえば、Red Hat Virtualization 4.1 を使用している場合には、4.1 向けに設計されたソフトウェア開発キットのバージョンを使用する必要があります。

## 1.2. PYTHON ソフトウェア開発キットのインストール

Python ソフトウェア開発キットをインストールします。

### Python ソフトウェア開発キットのインストール

1. 必要なチャンネルを有効にします。

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

2. 必要なパッケージをインストールします。

a. バージョン 3 の場合:

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

b. バージョン 4 の場合:

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

Python ソフトウェア開発キットと付属ドキュメントが **/usr/lib/python2.7/site-packages/ovirtsdk/** のディレクトリーにダウンロードされ、Python プロジェクトに追加できる状態となりました。



## 第2章 PYTHON クイックスタートの例

### 2.1. PYTHON のクイックスタート について

本章では、Python SDK を使用して基本的な Red Hat Virtualization 環境で仮想マシンを作成する手順を実例をあげて説明します。



#### 重要

本章の例は、Python SDK のバージョン 3 で使用するように設計されています。

以下にあげる例では、**ovirt-engine-sdk-python** パッケージで提供される **ovirtsdk** Python ライブラリーを使用しています。このパッケージは、Red Hat Subscription Manager 経由で **Red Hat Virtualization** プールにサブスクライブされているシステムに提供されています。お使いのシステムをサブスクライブしてソフトウェアをダウンロードする方法については、「[Python ソフトウェア開発キットのインストール](#)」を参照してください。

作業を行うにあたってのその他の要件

- ✧ ネットワーク接続された Red Hat Virtualization Manager のインストール
- ✧ ネットワーク接続があり、設定済みの Red Hat Enterprise Virtualization Host
- ✧ 仮想マシンにインストールするオペレーティングシステムが格納された ISO イメージファイル
- ✧ Red Hat Virtualization 環境を構成する論理/物理オブジェクトに関する実用的な知識
- ✧ Python プログラミング言語に関する実用的な知識



#### 重要

Python の例にはすべて、認証情報用のプレースホルダー (ユーザー名には **USER**、パスワードには **PASS**) が含まれています。Python で実行されるすべての要求が、ご使用の環境の認証要件を満たしていることを確認してください。



#### 注記

Red Hat Virtualization Manager は、各リソースの **id** 属性のグローバル一意識別子を生成します。以下の例に記載した識別子のコードは、ご使用の Red Hat Virtualization 環境の識別子コードと異なる場合があります。



#### 注記

これらの Python の例には、基本的な例外およびエラー処理ロジックのみが含まれます。SDK に固有な例外処理の詳細については、**ovirtsdk.infrastructure.errors** モジュールの pydoc を参照してください。

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

## 2.2. 例: PYTHON を使用した API エントリーポイントへのアクセス

**ovirtsdk** Python ライブラリーは、API のエントリーポイントとして機能する **API** クラスを提供します。

### 例2.1 Python を使用した API エントリーポイントへのアクセス

以下の Python の例は、**rhevm.demo.redhat.com** で Red Hat Virtualization Manager によって提供される REST API のインスタンスに接続します。以下の例では、**API** クラスのインスタンスを作成して接続します。接続が成功した場合には、メッセージが表示されます。最後に、**API** クラスの **disconnect()** メソッドが呼び出されて接続が終了します。

この例で **API** クラスのコンストラクターに以下のパラメーターを提供します。

- ✧ 接続する Manager の **url**
- ✧ 認証するユーザーの **username**
- ✧ 認証するユーザーの **password**
- ✧ 証明書へのパスである **ca\_file**。証明書は、Manager の認証局用コピーであることが想定されます。これは、**https://[engine-fqdn]\_ovirt-engine/services/pki-resource?resource=ca-certificate&format=\_X509-PEM-CA** から取得できます。

**API** クラスのコンストラクターは、他のパラメーターをサポートしますが、以下の例では、必須パラメーターのみを指定しています。

```
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
```

接続試行が成功した場合には、次の例のような出力が表示されます。

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

## 2.3. 例: PYTHON を使用したデータセンターコレクションの一覧表示

**API** クラスは、**datacenters** という名前のデータセンターコレクションへのアクセスを提供します。このコレクションには、環境内の全データセンターが含まれます。

### 例2.2 Python を使用したデータセンターコレクションの一覧表示

以下の Python の例は、**datacenters** コレクション内のデータセンターを一覧表示します。また、コレクション内の各データセンターの基本的な情報も出力します。

```
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
```

**Default** データセンターのみが存在する環境で、そのデータセンターがアクティブ化されていない場合は、次の例のような出力が表示されます。

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

## 2.4. 例: PYTHON を使用したクラスターコレクションの一覧表示

API クラスは、**clusters** という名前のクラスターコレクションを提供します。このコレクションには、環境内の全クラスターが含まれます。

### 例2.3 Python を使用したクラスターコレクションの一覧表示

以下の Python の例は、**clusters** コレクション内のクラスターを一覧表示します。また、コレクション内の各クラスターの基本的な情報も出力します。

```
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

```

**Default** クラスターのみが存在する環境の場合は、次の例のような出力が表示されます。

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

## 2.5. 例: PYTHON を使用した論理ネットワークコレクションの一覧表示

**API** クラスは、**networks** という名前の論理ネットワークへのアクセスを提供します。このコレクションには、環境内の全論理ネットワークが含まれます。

### 例2.4 Python を使用した論理ネットワークコレクションの一覧表示

以下の Python の例は、**networks** コレクション内の論理ネットワークを一覧表示します。また、コレクション内の各ネットワークの基本的な情報も出力します。

```

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

```

デフォルトの管理ネットワークのみが存在する環境の場合は、次の例のような出力が表示されます。

```
ovirtmgmt (000000000-0000-0000-0000-000000000009)
```

## 2.6. 例: PYTHON を使用したホストコレクションの一覧表示

**API** クラスは、**hosts** という名前のホストコレクションへのアクセスを提供します。このコレクションには、環境内の全ホストが含まれます。

### 例2.5 Python を使用したホストコレクションの一覧表示

以下の Python の例は、**hosts** コレクション内のホストを一覧表示します。

```

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

```

**Atlantic** という名前のホストが1台のみアタッチされた環境では、次のような出力が表示されます。

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

## 2.7. 例: ISO ストレージドメインの ISO ファイルの表示

**API** クラスは **storagedomains** という名前のストレージドメインコレクションへのアクセスを提供します。同様に、このコレクションには、ストレージドメイン内のファイルを記述する **files** コレクションが含まれています。

### 例2.6 ISO ストレージドメインの ISO ファイルの表示

この Python の例は、Red Hat Virtualization 環境内の各 ISO ストレージドメイン内にある ISO ファイルを一覧で出力します。

```

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. 例: 仮想マシンのサイズの表示

**API** クラスは、**vms** という名前の仮想マシンコレクションへのアクセスを提供します。同様に、このコレクションには仮想マシンにアタッチされた各ディスクの詳細を記述する **disks** コレクションが含まれています。

### 例2.7 仮想マシンのサイズの表示

以下の Python の例は、Red Hat Virtualization 環境の仮想マシンとその合計ディスクサイズ (バイト単位) を一覧で出力します。

```

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. 例: PYTHON を使用した NFS データストレージの作成

Red Hat Virtualization 環境の初回作成時には、少なくともデータストレージドメインを 1 つと ISO ストレージドメインを 1 つ定義する必要があります。データストレージドメインは、仮想マシンのディスクイメージを格納するのに使用する一方、ISO ストレージドメインはゲストオペレーティングシステムのインストールメディアを格納するのに使用します。

API クラスは、**storagedomains** という名前のストレージドメインコレクションへのアクセスを提供します。このコレクションには、環境内の全ストレージドメインが含まれています。**storagedomains** コレクションはストレージドメインの追加と削除に使用することができます。



### 注記

以下の例に記載したコードは、リモートの NFS 共有が Red Hat Virtualization 用に事前設定済みであることを前提としています。NFS 共有の使用準備に関する詳しい説明は、**Red Hat Virtualization 管理ガイド**を参照してください。

### 例2.8 Python を使用した NFS データストレージの作成

以下の Python の例は、NFS データドメインを **storagedomains** コレクションに追加します。Python を使用した NFS ストレージドメイン追加は以下のような手順で行います。

1. **datacenters** コレクションの **get** メソッドを使用して、ストレージをアタッチする必要のあるデータセンターを特定します。

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

2. **hosts** コレクションの **get** メソッドを使用して、ストレージのアタッチに使用する必要のあるホストを特定します。

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

3. NFS ストレージドメインの **Storage** パラメーターを定義します。以下の例では、NFS の場所に **192.0.43.10/storage/data** を使用しています。

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

4. **storagedomains** コレクションの **add** メソッドを使用して、ストレージドメイン作成を要求します。**Storage** パラメーターに加えて、以下の情報をすべて渡す必要があります。

- ✧ ストレージドメインの名前
- ✧ **datacenters** コレクションから取得したデータセンターオブジェクト
- ✧ **hosts** コレクションから取得したホストオブジェクト
- ✧ 追加するストレージドメインのタイプ (**data**、**iso**、または **export**)
- ✧ 使用するストレージフォーマット (**v1**、**v2**、または **v3**)

これらの手順を組み合わせた完全なスクリプトは以下のようになります。

```
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
```

**add** メソッド呼び出しが成功した場合は、スクリプトによって以下のような出力が表示されます。

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

## 2.10. 例: PYTHON を使用した NFS ISO ストレージの作成

仮想マシンを作成するには、ゲストオペレーティングシステムのインストールメディアを使用可能な状態にする必要があります。Red Hat Virtualization 環境では、インストールメディアを ISO ストレージドメインに保管します。





## 注記

以下の例に記載したコードは、リモートの NFS 共有が Red Hat Virtualization 用に事前設定済みであることを前提としています。NFS 共有の使用準備に関する詳しい説明は、**Red Hat Virtualization 管理ガイド**を参照してください。

### 例2.9 Python を使用した NFS ISO ストレージの作成

以下の Python の例は、NFS データドメインを **storagedomains** コレクションに追加します。Python を使用した NFS ストレージドメイン追加は以下のような手順で行います。

1. **datacenters** コレクションの **get** メソッドを使用して、ストレージをアタッチする必要のあるデータセンターを特定します。

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

2. **hosts** コレクションの **get** メソッドを使用して、ストレージのアタッチに使用する必要のあるホストを特定します。

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

3. NFS ストレージドメインの **Storage** パラメーターを定義します。以下の例では、NFS の場所に **192.0.43.10/storage/iso** を使用しています。

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

4. **storagedomains** コレクションの **add** メソッドを使用して、ストレージドメイン作成を要求します。**Storage** パラメーターに加えて、以下の情報をすべて渡す必要があります。

- ※ ストレージドメインの名前
- ※ **datacenters** コレクションから取得したデータセンターオブジェクト
- ※ **hosts** コレクションから取得したホストオブジェクト
- ※ 追加するストレージドメインのタイプ (**data**、**iso**、または **export**)
- ※ 使用するストレージフォーマット (**v1**、**v2**、または **v3**)

これらの手順を組み合わせた完全なスクリプトは以下のようになります。

```
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

```

**add** メソッド呼び出しが成功した場合は、スクリプトによって以下のような出力が表示されます。

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

## 2.11. 例: PYTHON を使用したデータセンターへのストレージドメインのアタッチ

Red Hat Virtualization にストレージドメインを追加した後は、データセンターにアタッチしてアクティブ化し、使用できる状態にする必要があります。

### 例2.10 Python を使用した、データセンターへのストレージドメインのアタッチ

この Python の例は、**data1** という名前のデータストレージドメインと、**iso1** という名前の ISO ストレージドメインを **default** データセンターにアタッチします。アタッチのアクションは、データセンターの **storagedomains** コレクションの **add** メソッドを使用して実行されます。

```

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_iso = dc.storagedomains.add(sd_iso)
    print "Attached ISO storage domain '%s' to data center '%s'
(Status: %s)." %
        (dc_iso.get_name(), dc.get_name, dc_iso.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

```

**add** メソッドの呼び出しが成功した場合は、スクリプトによって以下のような出力が表示されます。

```

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

```

**status** には、ストレージドメインのアクティブ化がまだ必要であることが表示される点に注意してください。

## 2.12. 例: PYTHON を使用したストレージドメインのアクティブ化

ストレージドメインを Red Hat Virtualization に追加してデータセンターにアタッチした後には、これらのストレージドメインをアクティブ化して使用できる状態にする必要があります。

### 例2.11 Python を使用したストレージドメインのアクティブ化

以下の Python の例は、**data1** という名前のデータストレージドメインおよび**iso1** という名前の ISO ストレージドメインをアクティブ化します。これらのストレージドメインはいずれも **Default** データセンターにアタッチされています。アクティブ化のアクションは、ストレージドメインの **activate** メソッドを使用して実行されます。

```

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

```

**activate** 要求が成功した場合は、スクリプトにより以下のような出力が表示されます。

```

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

```

**status** には、ストレージドメインがアクティブ化されたことが表示される点に注意してください。

## 2.13. 例: PYTHON を使用した仮想マシンの作成

仮想マシンを作成するには、いくつかの手順を実行する必要があります。第 1 のステップ (以下に説明) は、仮想マシンオブジェクト自体を作成する手順です。

### 例2.12 Python を使用した仮想マシンの作成

以下の Python の例は、**vm1** という名前の仮想マシンを作成します。この仮想マシンの要件は以下のとおりです。

- ※ 512 MB のメモリーが割り当てられていること (バイト単位で表示)。

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

- ※ **Default** クラスターにアタッチされ、その結果**Default** データセンターにもアタッチされていること。

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

- ※ デフォルトの **Blank** テンプレートをベースとしていること。

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

- ※ 仮想ハードディスクドライブから起動すること。

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

上記のオプションは、**vms** コレクションの **add** メソッドを使用して仮想マシン自体を作成する前に、仮想マシンパラメーターオブジェクトに組み込みます。

```
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
```

**add** 要求が成功した場合には、スクリプトにより以下のような出力が表示されます。

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

## 2.14. 例: PYTHON を使用した仮想マシン NIC の作成

新規作成した仮想マシンが確実にネットワークにアクセスできるようにするには、仮想 NIC を作成してアタッチする必要があります。

### 例2.13 Python を使用した仮想マシン NIC の作成

以下の Python の例は、**nic1** という名前の NIC を作成し、**vm1** という名前の仮想マシンにアタッチします。この例の NIC は以下の条件を満たしている必要があります。

- ✱ **virtio** ネットワークデバイスであること。

```
nic_interface = "virtio"
```

- ✱ **ovirtmgmt** 管理ネットワークにリンクされていること。

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

上記のオプションは、仮想マシンの **nics** コレクションの **add** メソッドを使用して NIC を作成する前に、NIC パラメーターオブジェクトに組み込みます。

```
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

```

**add** 要求が成功した場合には、スクリプトにより以下のような出力が表示されます。

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

## 2.15. 例: PYTHON を使用した仮想マシンのストレージディスクの作成

新規作成した仮想マシンが永続的なストレージに確実にアクセスできるようにするには、ディスクを作成してアタッチする必要があります。

### 例2.14 Python を使用した仮想マシンストレージの作成

以下の Python の例は、8 GB の **virtio** ディスクドライブを作成し、**vm1** という名前の仮想マシンにアタッチします。この例のディスクは、以下の条件を満たしている必要があります。

- ✧ **data1** という名前のストレージドメイン上に格納されること。

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

- ✧ サイズは 8 GB とすること。

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

- ✧ ディスクタイプは、(**data** ではなく) **system** タイプであること。

```
disk_type = "system"
```

- ✧ **virtio** ストレージデバイスであること。

```
disk_interface = "virtio"
```

- ✧ **cow** フォーマットで格納されること。

```
disk_format = "cow"
```

- ✧ 使用可能なブートデバイスとしてマークされること。

```
disk_bootable = True
```

上記のオプションは、仮想マシンの **disks** コレクションの **add** メソッドを使用してディスク自体を作成する前に、ディスクパラメーターオブジェクトに組み込みます。

```

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

```

**add** 要求が成功した場合には、スクリプトにより以下のような出力が表示されます。

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

## 2.16. 例: PYTHON を使用した仮想マシンへの ISO イメージのアタッチ

新規作成した仮想マシンにゲストオペレーティングシステムのインストールを開始するには、オペレーティングシステムのインストールメディアを格納した ISO ファイルをアタッチする必要があります。

### 例2.15 ISO イメージの特定

ISO イメージは、ISO ストレージドメインにアタッチされた **files** コレクション内にあります。以下の例は、ISO ストレージドメイン上の **files** コレクションのコンテンツを一覧表示します。

```
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

```

成功した場合には、スクリプトにより、**files** コレクション内のファイルごとに以下の例のようなエントリーが出力されます。

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

ISO ドメイン上のファイルには必ず一意名が付けられるため、ファイルの **id** および **name** 属性が共有される点に注意してください。

### 例2.16 Python を使用した仮想マシンへの ISO イメージのアタッチ

以下の Python の例では、**RHEL6.3-Server-x86\_64-DVD1.iso** の ISO イメージファイルを **vm1** 仮想マシンにアタッチします。イメージファイルを特定し、仮想マシンの **cdroms** コレクションの **add** メソッドを使用してアタッチします。

```

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
```

**add** 要求が成功した場合には、スクリプトにより以下のような出力が表示されます。

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

## 注記

上記の例は、ステータスが **Down** の仮想マシンに ISO をアタッチするための手順です。ステータスが **Up** の仮想マシンに ISO をアタッチするには、第 2 の **try** ステートメントを以下のように変更してください。

```
try:
    cdrom=cd_vm.cdroms.get(id="_000000000-0000-0000-0000-000000000000_")
    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)
```

## 例2.17 Python を使用する、仮想マシンからの CD-ROM の取り出し

仮想マシンの **cdrom** コレクションから ISO を取り出します。

```
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="000000000-0000-0000-0000-000000000000").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. 例: PYTHON を使用したディスクのデタッチ

Python ソフトウェア開発キットを使用して、仮想マシンから仮想ディスクをデタッチすることができます。

### 例2.18 Python を使用したディスクのデタッチ

```

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. 例: PYTHON を使用した仮想マシンの起動

仮想マシンを起動します。

### 例2.19 Python を使用した仮想マシンの起動

以下の例では、**start** メソッドを使用して仮想マシンを起動します。

```

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

```

**start** 要求が成功した場合は、スクリプトによって以下のような出力が表示されます。

```
Started 'vm1'.
```

**status** には、仮想マシンが起動して **up** の状態となったことが表示される点に注意してください。

## 2.19. 例: PYTHON を使用した上書きパラメーターでの仮想マシンの起動

上書きパラメーターで仮想マシンを起動します。

### 例2.20 Python を使用した上書きパラメーターでの仮想マシンの起動

以下の例では、Windows ISO を使用して仮想マシンをブートして、Windows ドライバーを含む **virtio-win\_x86.vfd** フロッピーディスクをアタッチします。このアクションは、管理ポータルまたはユーザーポータルで 1 回実行ウィンドウを使用して仮想マシンを起動するのと同じです。

```

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

```

### 注記

CD イメージとフロッピーディスクファイルはすでに ISO ドメインで利用できる状態であればなりません。まだ用意されていない場合は、ISO アップローダーを使用してこれらのファイルをアップロードしてください。詳しい情報は「[ISO アップローダーツール](#)」を参照してください。

## 2.20. 例: PYTHON を使用した CLOUD-INIT での仮想マシンの起動

Python を使用して Cloud-Init で仮想マシンを起動します。

### 例2.21 Python を使用した Cloud-Init での仮想マシンの起動

以下の例では、Cloud-Init ツールを使用して仮想マシンを起動し、etho0 インターフェースの静的 IP ホスト名を設定する方法を示しています。

```

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. 例: PYTHON を使用したシステムイベントのチェック

Red Hat Virtualization Manager は、多くのシステムイベントを記録およびログ記録します。これらのイベントログには、ユーザーインターフェース、システムログファイル、および API を使用してアクセスできます。**ovirtsdk** ライブラリーは、**events** コレクションを使用してイベントを公開します。

### 例2.22 Python を使用したシステムイベントのチェック

以下の例では、**events** コレクションがリストされます。次の点に注意してください。

- ※ 利用可能なすべての結果ページが返されるようにするために、**list** メソッドの **query** パラメーターを使用しています。デフォルトでは、**list** メソッドは長さが最大 **100** のレコードにデフォルト設定される結果の最初のページのみを返します。
- ※ イベントが発生した順に出力に示されるよう、結果となるリストは逆順にソートされます。

```

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

```

このスクリプトの出力は以下のようになります。ただし、環境の状態によってイベントが異なります。

```

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 ソフトウェア開発キットの使用

### 3.1. PYTHON を使用した API への接続

Python を使用して REST API に接続するには、**ovirtsdk.api** モジュールから **API** クラスのインスタンスを作成する必要があります。この操作を行うには、最初にスクリプトの先頭でクラスをインポートする必要があります。

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

**API** クラスのコンストラクターは、複数の引数を取ります。サポートされる引数は以下のとおりです。

#### url

接続する Manager の URL (**/api** パスを含む) を指定します。このパラメーターは必須です。

#### username

接続時に使用するユーザー名をユーザープリンシパル名 (UPN) 形式で指定します。このパラメーターは必須です。

#### password

**username** パラメーターにより提供されたユーザー名のパスワードを指定します。このパラメーターは必須です。

#### kerberos

有効な Kerberos チケットを使用して接続を認証します。有効な値は **True** と **False** です。このパラメーターはオプションです。

#### key\_file

**cert\_file** で指定された証明書に関連付けられた秘密キーを含む PEM 形式のキーファイルを指定します。このパラメーターは必須です。

#### cert\_file

サーバーでクライアントの ID を確立するために使用する PEM 形式のクライアント証明書を指定します。このパラメーターはオプションです。

#### ca\_file

サーバーの認証局の証明書ファイルを指定します。**insecure** パラメーターが **True** に設定されていない限り、このパラメーターは必須です。

#### port

接続時に使用するポートを指定します (**url** パラメーターの一部として提供されなかった場合)。このパラメーターはオプションです。

#### timeout

要求がタイムアウトするまでの時間を秒単位で指定します。このパラメーターはオプションです。



### persistent\_auth

この接続に対して永続認証を有効にするかどうかを指定します。有効な値は **True** と **False** です。このパラメーターはオプションで、デフォルトでは **False** に設定されています。

### insecure

認証局を使用しない SSL 経由の接続を許可します。有効な値は **True** と **False** です。**insecure** パラメーターが **False** (デフォルト値) に設定されている場合は、**ca\_file** を指定して接続をセキュリティ保護する必要があります。

このオプションにより、サーバーになりすました中間者 (MITM) 攻撃が可能となる場合があるため、細心の注意を払って使用してください。

### filter

ユーザーパーミッションベースのフィルターをオンまたはオフにするかどうかを指定します。有効な値は、**True** と **False** です。**filter** パラメーターが **False** (デフォルト値) に設定された場合は、管理ユーザーの認証情報を提供する必要があります。**filter** パラメーターが **True** に設定された場合は、どのユーザーでも使用でき、Manager はユーザーのパーミッションに基づいてユーザーが利用可能なアクションをフィルタリングします。

### debug

この接続に対してデバッグモードを有効にするかどうかを指定します。有効な値は **True** と **False** です。このパラメーターはオプションです。

ovirtsdkAPI Python クラスの別個のインスタンスを作成して操作することにより、複数の Red Hat Virtualization Manager と通信することが可能です。

このサンプルスクリプトは、**API** クラスのインスタンスを作成し、**test()** メソッドを使用して接続が動作していることを確認し、**disconnect()** メソッドを使用して接続解除します。

```
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()
```

**API** クラスによりサポートされるメソッドの完全な一覧については、**ovirtsdk.api** モジュールの Pydoc の出力を参照してください。

```
$ pydoc ovirtsdk.api
```

## 3.2. リソースおよびコレクション

API の RESTful 特性は、理論的および実践的な理由から Python バインディング全体で明白です。すべての RESTful API には、認識する必要がある次の 2 つの重要な概念があります。

### Collections

コレクションは、同じタイプのリソースのセットです。API は、最上位コレクションとサブコレクションの両方を提供します。最上位コレクションの例としては、環境内の全仮想化ホストを含む **hosts** コレクションがあげられます。また、サブコレクションの例としては、ホストリソースにアタッチされた全ネットワークインターフェースカードのリソースを含む **host.nics** コレクションがあげられます。

コレクションと対話するインターフェースは、リソースを追加するメソッド (**add**)、リソースを取得するメソッド (**get**)、およびリソースをリストするメソッド (**list**) を提供します。

## Resources

RESTful API のリソースは、表現されるリソースの特定のタイプに関連する属性セットも含む固定インターフェースがあるオブジェクトです。リソースと対話するインターフェースは、リソースを更新するメソッド (**update**) とリソースを削除するメソッド (**delete**) を提供します。また、一部のリソースはリソースタイプに固有なアクションをサポートします。**Host** リソースの **approve** メソッドはその一例です。

### 3.3. コレクションからのリソースの取得

リソースは、**get** および **list** メソッドを使用してコレクションから取得されます。

#### get

コレクションから単一のリソースを取得します。取得するアイテムは、引数として提供される名前に基づいて決定されます。**get** メソッドでは以下の引数を使用できます。

- ✧ **name** - コレクションから取得するリソースの名前。
- ✧ **id** - コレクションから取得するグローバル一意識別子 (GUID)。

#### list

コレクションから単一のリソースを取得します。取得するアイテムは、引数として提供される名前に基づいて決定されます。**get** メソッドでは以下の引数を使用できます。

- ✧ **\*\*kwargs** - キーワードベースのフィルタリングを可能にする追加引数のディクショナリー。
- ✧ **query** - Red Hat Virtualization ユーザーインターフェースを使用して実行される検索に使用するのと同じ形式で記述されたクエリー。
- ✧ **max** - 取得するリソースの最大数。
- ✧ **case\_sensitive** - 検索の用語で大文字と小文字を区別するかどうかを決定します (**True** または **False**、デフォルト値は **True**)。

### 3.4. コレクションからの特定のリソースの取得

これらの例では、特定のリソースは **get** メソッドを使用してコレクションから取得されます。

#### 例3.1 名前による特定のリソースの取得

**get** メソッドの **name** パラメーターを使用して、**datacenters** コレクションから **Default** データセンターを取得します。

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

以下の構文と同等です。

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

**get** 要求に関する追加の情報は、**all\_content** ヘッダーを使用して取得することができます。

### 例3.2 特定のリソースの追加情報の取得

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

## 3.5. コレクションからのリソースリストの取得

以下の例では、**list** メソッドを使用して、コレクションからリソースのリストを取得します。

### 例3.3 コレクション内の全リソースのリスト取得

**datacenters** コレクションの全リソースのリストを取得します。**list** メソッドの **query** パラメーターを指定すると、engine ベースのクエリーを使用できます。この方法では、SDK は、管理およびユーザーポータルで実行されたのと同じ形式のクエリーをサポートします。また、**query** パラメーターは、コレクションを通じて反復処理を行う間にページネーションの引数を提供するメカニズムでもあります。

```
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)
```

上記の例では、最終的に **datacenters** コレクションに含まれるリソースのリストがローカルで定義された **dc\_list** リスト変数に格納されます。

**警告**

コレクションの **list** メソッドは、**SearchResultsLimit** Red Hat Virtualization Manager 設定キーで許可された数の要素のみを返すよう制限されています。

**list** のすべてのレコードが返されるようにするには、この例で示したように、結果をページネーションすることを推奨します。

また、**list** メソッドの **max** パラメーターに、取得するレコードの最大数を設定することもできます。

**例3.4 キーワードベースフィルタリングに一致するコレクション内のリソースのリスト取得**

ストレージタイプが **nfs** の **datacenters** コレクション内にある全リソースのリストを取得します。この例では、**query** パラメーターと **\*\*kwargs** パラメーターの両方が提供されます。前の例と同様に、ページネーションには **query** が使用されます。**\*\*kwargs** パラメーターは、データセンターのストレージタイプに基づいてフィルタリングするために使用されます。

```
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"})
```

上記の例では、ストレージタイプが **nfs** の **datacenters** コレクションに含まれるリソースのリストが最終的に、ローカルで定義された **dc\_list** リスト変数に格納されます。

**3.6. コレクションへのリソースの追加**

コレクションの **add** メソッドはリソースを追加します。追加するリソースは、提供されたパラメーターに基づいて作成されます。パラメーターは、**ovirtsdk.xml.params** モジュールからオブジェクトのインスタンスを使用して **add** メソッドに提供されます。モジュールからどのクラスを使用するかは、作成するリソースのタイプによって異なります。

**例3.5 コレクションへのリソースの追加**

以下の例では、仮想マシンリソースが作成されます。

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

この例で作成される仮想マシンは、実行する準備が整っていない状態です。Red Hat Virtualization リソースを作成するプロセスを示すために提供されています。

- ✦ 作成されるリソースのタイプに対してパラメーターオブジェクトのインスタンスを作成します。
- ✦ リソースを追加するコレクションを特定します。
- ✦ コレクションの **add** メソッドを呼び出し、パラメーターオブジェクトをパラメーターとして渡します。

一部のパラメーターオブジェクトには、独自の複合型パラメーターがあります。

### 例3.6 複合型パラメーター

この例では、完全なバージョン 4.1 互換モードで稼働する NFS データセンターを作成します。この作業を行うには、最初に **ovirtsdk.xml.params.Version** オブジェクトを構築する必要があります。次にこのオブジェクトは、作成するデータセンターのパラメーターを含む **ovirtsdk.xml.params.DataCenter** オブジェクトのインスタンスの作成時にパラメーターとして使用されます。リソースは、**datacenters** コレクションの **add** メソッドを使用して作成されます。

```
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. コレクション内のリソースの更新

リソースを更新するには、リソースが存在するコレクションからリソースを取得し、必要なパラメーターを変更し、リソースに対して **update** メソッドを呼び出して変更を保存する必要があります。パラメーターの変更は、取得されたリソースの **set\_\*** メソッドを使用して実行されます。

### 例3.7 リソースの更新

以下の例では、**DemoDataCenter** という名前のデータセンターの説明が更新されます。

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

## 3.8. コレクションからのリソースの削除

リソースを削除するには、リソースを含むコレクションからリソースを取得し、リソースの **delete** メソッドを呼び出す必要があります。

### 例3.8 コレクションからのリソースの削除

**DemoVM** という名前の仮想マシンを **vms** コレクションから削除します。

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

## 3.9. エラーの処理

エラーが発生した場合に、ソフトウェア開発キットは例外を使用してエラーを強調します。ソフトウェア開発キットは、Python インタープリター自体が定義したものに加え、例外タイプも定義します。これらの例外は、**ovirtsdk.infrastructure.errors** モジュールにあります。

### ConnectionError

トランスポート層エラーが発生した場合に送出されます。

### DisconnectedError

SDK が明示的に切断された後に使用を試みた場合に送出されます。

### ImmutableError

同じドメイン下に SDK インスタンスがすでに存在している間に SDK の開始を試みた場合に送出されます。SDK 3.2 以降のバージョンに適用されます。

### NoCertificatesError

CA が提供されず、`--insecure` が 'False' に指定されている場合に送出されます。

### RequestError

あらゆる種類の oVirt サーバーエラーが発生した場合に送出されます。

### UnsecuredConnectionAttemptError

サーバーが HTTPS を実行中に HTTP プロトコルが使用された場合に送出されます。

### MissingParametersError

ID または名前のいずれも提供せずに `get()` メソッドの使用を試みた場合に送出されます。

これらの例外は、他の Python 例外と同様にキャッチおよび処理できます。

### 例3.9 ConnectionError 例外のキャッチ

```
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 リファレンスドキュメント

### 4.1. PYTHON リファレンスドキュメント

[pydoc](#) で作成したドキュメントは、以下のモジュールで使用できます。このドキュメントは、**ovirt-engine-sdk-python** パッケージで提供されています。

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

Red Hat Virtualization Manager がインストールされているマシンで、以下のコマンドを実行して、これらのドキュメントの最新版を確認します。

```
$ pydoc [MODULE]
```