Red Hat Virtualization 4.0

Python SDK ガイド

Red Hat Virtualization Python SDK の使用

Last Updated: 2022-12-25

Red Hat Virtualization 4.0 Python SDK ガイド

Red Hat Virtualization Python SDK の使用

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、Red Hat Virtualization Python ソフトウェア開発キットのバージョン 3 およびバージョン 4 をインストールして操作する方法を説明します。

目次

パート I. PYTHON SOFWARE DEVELOPMENT KIT

第1章 概要

Python ソフトウェア開発キットは、Python ベースのプロジェクトで 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、管理タスクの高レベルな自動化のためのさまざまな機能にアクセスできます。

Red Hat Virtualization 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ます。

Version 3

V3 Python ソフトウェア開発キットは、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るクラスおよびメソッド構造との後方互換性を提供します。Red Hat Enterprise Virtualization 3.6 の Python ソフトウェア開発キットを使用して作成されたアプリケーションは、変更せずにこのバージョンで使用できます。

Version 4

V4 Python ソフトウェア開発キットは、更新されたクラス名とメソッド名と署名のセットを提供します。Red Hat Enterprise Virtualization 3.6 の Python ソフトウェア開発キットを使用して作成されたアプリケーションは、このバージョンで使用する前に更新する必要があります。

Python ソフトウェア開発キットのいずれのバージョンも、対応するパッケージをインストールし、必要なライブラリーを Python プロジェクトに追加することで、必要に応じて Red Hat Virtualization 環境で使用できます。

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には、以下が必要です。

- Red Hat Enterprise Linux 7 がインストールされているシステム。サーバーとワークステーションの両方のバリアントがサポートされています。
- 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 のインターフェイスです。そのため、Red Hat Virtualization 環境のバージョンに対応する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 4.0 を使用している場合は、4.0 用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。

手順1.1 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-4.0-rpms
```

2. 必要なパッケージをインストールします。

- V3 の場合：

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

- V4 の場合：

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

Python ソフトウェア開発キットと付属のドキュメントが **/usr/lib/python2.7/site-packages/ovirtsdk/** 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ようになりました。

第2章 PYTHON クイックスタートの例

2.1. PYTHON クイックスタートの概要

本章では、Python SDK を使用して、基本的な Red Hat Virtualization 環境内で仮想マシンを作成する手順を示す一連の例を紹介します。

重要

本章の例では、Python SDK の V3 と連携するように設計されています。

これらの例では、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ます。このパッケージは、Red Hat Subscription Manager の **Red Hat Virtualization** エンタイトルメントプールにサブスクライブしているシステムで利用できます。ソフトウェアをダウンロードするためにシステムをサブスクライブする方法は、「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。

以下も必要になります。

- Red Hat Virtualization Manager のネットワークインストール。
- ネットワーク接続され、接続された Red Hat Virtualization ホスト。
- 仮想マシンにインストールするためのオペレーティングシステムを含む ISO イメージファイル。
- Red Hat Virtualization 環境を設定する論理オブジェクトと物理オブジェクトの両方に関する実用的な理解。
- Python プログラミング言語の実用的な理解。

重要

すべての Python の例には、認証の詳細のプレースホルダー（ユーザー名*USER*、パスワードの場合は *PASS* ）が含まれます。Python で実行されるすべてのリクエストが、お使いの環境の認証要件を満たしていることを確認します。

注記

Red Hat Virtualization Manager は、各リソースの **id** 属性に対してグローバルに一意の識別子 (GUID) を生成します。これらの例の識別子コードは、お使いの Red Hat Virtualization 環境の識別子コードとは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 固有の例外処理の詳細は、**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

この Python の例では、rhevm.demo.redhat.com で Red Hat Virtualization Manager が提供する REST API のインスタンスに接続します。この例に接続するには、**API** クラスのインスタンスを作成します。接続に成功した場合、メッセージが出力されます。最後に、**API** クラスの **disconnect （）** メソッドが呼び出され、接続を閉じます。

この例の **API** クラスのコンストラクターに提供されるパラメーターは以下のとおりです。

- **接続** する Manager の URL。
- 認証するユーザーのユーザー **名**。
- 認証するユーザーの **パスワード**。
- 証明書へのパスである **ca_file**。証明書は、Manager の認証局のコピーであることが予想されます。**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。この例では、必須パラメーターのみが使用されています。

```
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
```

接続の試行に成功すると、例は次のテキストを出力します。

```
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、環境内のすべての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

この Python の例では、**datacenters** コレクションのデータセンターを一覧表示しています。また、コレクション内の各データセンターの基本情報も出力します。

```
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し、アクティベートされていない環境では、例は次のように出力します。

```
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、環境内のすべての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

この Python の例では、**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では、例は次のように出力します。

```
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、ネットワーク という名前の論理ネットワーク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、環境内のすべての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

この Python の例では、**networks** コレクション内の論理ネットワークを一覧表示しています。また、コレクション内の各ネットワークに関する基本情報も出力します。

```
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では、例は次のように出力します。

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

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

この Python の例では、**hosts** コレクション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 つしかない環境では、出力例が添付されています。

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

2.7. 例：ISO ストレージドメインでの ISO ファイルの一覧表示

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。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 ファイルの一覧表示

この Python の例では、Red Hat Virtualization 環境の各 ISO ストレージドメインの ISO ファイルの一覧を出力します。

```
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一覧表示

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。このコレクションには、仮想マシンに接続されている各ディスクの詳細を記述する **disks** コレクションが含まれます。

例2.7 仮想マシンのサイズの一覧表示

この Python の例では、Red Hat Virtualization 環境内の仮想マシンの一覧とその合計ディスクサイズをバイト単位で出力します。

```
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 環境が最初に作成される場合は、少なくともデータストレージドメインと ISO ストレージドメインを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ディスクイメージの

保存に使用されますが、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保存するために使用されます。

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、環境内のすべてのストレージドメインが含まれます。**storagedomains** コレクションは、ストレージドメインの追加および削除にも使用できます。

注記

この例で提供されるコードは、リモート NFS 共有が Red Hat Virtualization で使用するために事前設定されていることを前提としています。使用するための NFS 共有の準備の詳細については、Red Hat Virtualization 『Administration Guide』 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

この Python の例では、NFS データドメインを **storagedomains** コレクションに追加します。Python に NFS ストレージドメインを追加すると、いくつかの手順に分類できます。

1. Datacenter 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. **ホストコレクションの get メソッドを使用して、ストレージの割り当てに使用する 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。この例では、NFS の場所 192.0.43.10/storage/data が使用されています。**

```
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の作成を要求します。Storage パラメーターに加えて、以下を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。**

- host コレクションから取得したホストオブジェクト。
- 追加され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形式 (v1、v2、または v3)。

これらのステップが組み合わされると、完成した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 メソッド呼び出しに成功すると、スクリプトが出力されます。

```
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提供できる必要があります。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存します。

注記

この例で提供されるコードは、リモート NFS 共有が Red Hat Virtualization で使用するために事前設定されていることを前提としています。使用するための NFS 共有の準備の詳細については、Red Hat Virtualization 『Administration Guide』 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

この Python の例では、NFS ISO ドメインを storagedomains コレクションに追加します。Python に NFS ストレージドメインを追加すると、いくつかの手順に分類できます。

1. Datacenter 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. ホストコレクションの get メソッドを使用して、ストレージの割り当てに使用する 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。この例では、NFS の場所 192.0.43.10/storage/iso が使用されています。

```
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の作成を要求します。Storage パラメーターに加えて、以下を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。
- host コレクションから取得したホストオブジェクト。
- 追加され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形式 (v1、v2、または v3)。

これらのステップが組み合わされると、完成した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 メソッド呼び出しに成功すると、スクリプトが出力されます。

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

2.11. 例：PYTHON を使用したストレージドメインのデータセンターへの接続

ストレージドメインを Red Hat Virtualization に追加したら、使用できるようになる前にそれらをデータセンターにアタッチし、アクティブ化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割り当て

この Python の例では、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 デフォルト のデータセンターに割り当てます。attach アクションは、データセンターの storagedomains コレクションの add メソッドによって容易になります。

```
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")

    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
```

add メソッドへの呼び出しに成功すると、スクリプトが出力されます。

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

ステータス は、ストレージドメインをアクティブ化する必要があることを反映しています。

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ら、使用できるようになる前にアクティブ化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

この Python の例では、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アクティベートします。どちらのストレージドメインも Default データセンターに割り当てられます。activate アクションは、ストレージドメインの activate メソッドによって容易になります。

```
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")

    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
```

アクティベーション要求が 成功 すると、スクリプトにより以下が出力されます。

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

ステータス は、ストレージドメインがアクティブになったことを反映しています。

2.13. 例：PYTHON を使用した仮想マシンの作成

仮想マシンの作成は、いくつかの手順で実行されます。ここで説明する最初の手順は、仮想マシンオブジェクト自体を作成することです。

例2.12 Python を使用した仮想マシンの作成

この Python の例では、vm1 という名前の仮想マシンを作成します。この例の仮想マシン：

- 512 MB のメモリーが必要です（バイト単位で表されます）。

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

- Default クラスターにアタッチする必要があるため、Default データセンターに接続されている必要があります。

```
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する必要があります。

```
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必要があります。

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

-

これらのオプションは、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統合されます。

```
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")

    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
```

追加 要求に成功すると、スクリプトにより以下が出力されます。

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

2.14. 例：PYTHON を使用した仮想マシン NIC の作成

新規作成された仮想マシンが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

この Python の例では、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、以下のとおりです。

- virtio ネットワークデバイスである必要があります。

```
nic_interface = "virtio"
```

- ovirtmgmt 管理ネットワークにリンクする必要があります。

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

これらのオプションは、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統合されます。

```
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")

    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
```

追加 要求に成功すると、スクリプトにより以下が出力されます。

```
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

新しく作成された仮想マシンが永続ストレージにアクセスできるようにするには、ディスクを作成して割り当てる必要があります。

例2.14 Python を使用した仮想マシンのストレージディスクの作成

この Python の例では、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：

- data1 という名前のストレージドメインに保存する必要があります。

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

- サイズは 8 GB である必要があります。

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

- システム タイプのディスク(データではなく)である必要があります。

```
disk_type = "system"
```

- virtio ストレージデバイスである必要があります。

```
disk_interface = "virtio"
```

- cow 形式で保存する必要があります。

```
disk_format = "cow"
```

- 使用可能なブートデバイスとしてマークされている必要があります。

```
    disk_bootable = True
```

これらのオプションは、仮想マシンのディスクコレクションの 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統合されます。

```
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")

    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
```

追加 要求に成功すると、スクリプトにより以下が出力されます。

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

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

新しく作成され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含む ISO ファイルを割り当てる必要があります。

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 ファイル コレクションにあります。この例では、ISO ストレージドメイン上の files コレクションの内容を一覧表示します。

```
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
```

成功すると、スクリプトは files コレクションにある各ファイルで以下のようなエントリーを出力します。

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

ISO ドメインのファイルは、ファイルの id 属性と name 属性を一意に付ける必要があるため、共有されている必要があること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

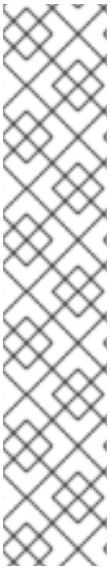
この Python の例では、RHEL6.3-Server-x86_64-DVD1.iso ISO イメージファイルを vm1 仮想マシンに割り当てます。イメージファイルが特定されると、仮想マシンの cdroms コレクションの add メソッドを使用してイメージファイルがアタッチされます。

```
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")

    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
```

追加 要求に成功すると、スクリプトにより以下が出力されます。

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



注記

この手順では、ステータスが Down の仮想マシンに ISO イメージをアタッチします。Up ステータスの仮想マシンに ISO をアタッチするには、2 番目の try ステートメントを以下のように変更します。

```
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)
```

例2.17 Python を使用した仮想マシンからの 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="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. 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の割り当てを解除できます。

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

仮想マシンの起動

例2.19 Python を使用した仮想マシンの起動

この例では、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
```

開始 要求に成功すると、スクリプトにより以下が出力されます。

```
Started 'vm1'.
```

ステータス は、仮想マシンが起動し、が up であることを反映していることに注意してください。

2.19. 例：PYTHON を使用したオーバーライドパラメーターを使用した仮想マシンの起動

パラメーターを上書きして仮想マシンの起動

例2.20 Python を使用して上書きされたパラメーターで仮想マシンの起動

この例では、Windows ISO で仮想マシンを起動し、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の Run Once ウィンドウを使用して仮想マシンを起動することと同じです。

```
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必要があります。そうでない場合は、ISO アップローダーツールを使用してファイルをアップロードします。詳細 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 で仮想マシンの起動

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

この例は、Cloud-Init ツールを使用して仮想マシンを起動し、eth0 インターフェイスのホスト名と静的 IP 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(
  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. 例：PYTHON を使用したシステムイベントの確認

Red Hat Virtualization Manager は、多くのシステムイベントを記録およびログします。これらのイベントログには、ユーザーインターフェイス、システムログファイルからアクセスでき、API を使用してアクセスすることもできます。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確認

この例では、events コレクションが一覧表示されます。以下の点に留意してください。

- list メソッドの query パラメーターは、使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使用されます。デフォルトでは、list メソッドは結果の最初のページのみを返します。デフォルトは最大 100 レコードの長さになります。
- 結果のリストが逆となり、発生した順序でイベントが出力に含まれるようにします。

```
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")

    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
```

このスクリプトからの出力は、以下のようになります。環境の状態に応じて異なるイベントを持ちます。

```
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。これを可能にするには、スクリプトの開始時に クラスを最初に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、いくつかの引数を取ります。サポートされている引数は次のとおりです。

url

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

username

ユーザープリンシパル名(UPN)形式で、接続するユーザー名を指定します。このパラメーターは必須です。

password

username パラメーターで指定し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 および False です。このパラメーターは任意です。

key_file

cert_file で指定された証明書に関連付けられた秘密鍵を含む 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任意です。

cert_file

サーバー上のクライアントのアイデンティティーを確立するために使用される 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任意です。

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

使用するポートを指定します。ここで、*url* パラメーターのコンポーネントとして提供されていません。このパラメーターは任意です。

timeout

要求がタイムアウトであると見なす前に経過できる時間を秒単位で指定します。このパラメーターは任意です。

persistent_auth

この接続に対して永続的な認証を有効にするかどうかを指定します。有効な値は True および False です。このパラメーターはオプションであり、デフォルトは False です。

insecure

認証局なしで SSL 経由で接続できるようにします。有効な値は True および False です。*insecure* パラメーターがデフォルトの False に設定されている場合、*ca_file* を指定して接続のセキュリティーを保護する必要があります。

このオプションは、中間者(MITM)攻撃者がサーバーのアイデンティティーを偽装できる可能性があるため、注意して使用する必要があります。

filter

ユーザーパーミッションベースのフィルターがオンまたはオフであるかを指定します。有効な値は True および False です。*フィルター* パラメーターが False（デフォルト）に設定されている場合には、指定される認証情報が管理ユーザーのものである必要があります。*filter* パラメーターを True に設定すると、任意のユーザーを使用でき、Manager は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ーします。

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 および False です。このパラメーターは任意です。

ovirtsdk.API Python クラスの別のインスタンスを作成し、操作することで、複数の Red Hat Virtualization Manager と通信できます。

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機能していることを確認します。また、disconnect （） メソッドを使用して切断します。

```
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 クラスでサポートされるメソッドの完全リストは、ovirtsdk.api モジュールの pydoc 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 の性質は、理論的および実用的な理由の両方で Python バインディング全体で明らかになります。すべての RESTful API には、認識する必要のある 2 つの主要な概念があります。

コレクション

コレクションは、同じタイプのリソースのセットです。API は最上位のコレクションとサブコレクションの両方を提供します。トップレベルのコレクションの例は、環境内のすべての仮想化ホストが含まれる hosts コレクションです。サブコレクションの例は、ホストリソースに割り当てられたすべてのネットワークインターフェイスカードのリソースが含まれる host.nics コレクションです。

コレクションと対話するインターフェイスは、リソースの追加（の追加）、リソースの取得(get)、およびリソースの一覧表示を行うメソッド(リスト)を提供します。

リソース

RESTful API のリソースは、固定されたインターフェイスを持つオブジェクトで、表現される特定のタイプのリソースに関連する属性のセットも含まれます。リソースと対話するインターフェイスは、リソースの更新(更新)およびリソースの削除(削除)を行う方法を提供します。また、リソー

スによっては、リソースタイプに固有のアクションをサポートするものもあります。たとえば、ホストリソースの 承認 方法が挙げられ ます。

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。取得する項目は、引数として提供される名前に基づいて決定されます。get メソッドは、以下の引数を取ります。

- 名前 - 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リソースのグローバル一意識別子(GUID)。

list

コレクションから任意の数のリソースを取得します。取得する項目は、提供される基準に基づいて決定されます。list メソッドは、以下の引数を取ります。

- **kwargs: キーワードベースのフィルターリングを可能にする追加の引数のディクショナリー。
- クエリー - Red Hat Virtualization ユーザーインターフェイスを使用して実行される検索に使用される形式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 - 検索用語が大文字と小文字を区別するかどうか(True または False)。デフォルトは Trueです。

3.4. コレクションからの特定のリソースの取得

この例では、get メソッドを使用して、特定のリソースが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、データ センター 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")
```

この構文は同等です。

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

all_content ヘッダーを使用して、取得 リクエストに関する追加情報を取得できます。

例3.2 特定のリソースに関する追加情報の取得

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

3.5. コレクションからのリソース一覧の取得

この例では、list メソッドを使用して、リソースの 一覧 をコレクションから取得します。

例3.3 コレクション内の全リソース一覧の取得

Datacenter コレクション内の全リソース一覧の取得。list メソッドの *クエリー* パラメーターを使用すると、エンジンベースのクエリーを使用できます。これにより、SDK は、管理ポータルおよびユーザーポータルで実行される形式と同じ形式でクエリーの使用をサポートします。*クエリー* パラメーターは、コレクションを介して反復処理中にページネーション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)
```

この例では、datacenters コレクションに含まれるリソースのリストは、最終的にローカルで定義された dc_list リスト変数に保存されます。

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ている数だけを返すように制限されます。

一覧 のすべてのレコードが返されるようにするには、以下の例のように結果をページ分割することが推奨されます。

または、list メソッドの *max* パラメーターを、取得する最大レコード数に設定することもできます。

例3.4 キーワードベースのフィルターに一致するコレクション内のリソース一覧の取得

nfs のストレージタイプを持つ datacenter コレクション内の全リソース一覧を取得します。この例では、*クエリー* パラメーターと ***kwargs* パラメーターの両方が指定されます。*クエリー* は、前述の例のようにページネーションに使用されます。***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ー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"})
```

この例では、ストレージタイプが nfs の datacenter コレクションに含まれるリソースの一覧は、最終的にローカルで定義された dc_list リスト変数に保存されます。

3.6. リソースのコレクションへの追加

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、提供される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 モジュールの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。モジュールの特定のクラスは、作成されるリソースのタイプによって異なります。

例3.5 リソースのコレクションへの追加

この例では、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)
```

この例で作成された仮想マシンを実行する準備ができていませんが、Red Hat Virtualization リソースの作成プロセスを示しています。

- 作成されるリソースタイプの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。
- パラメーターオブジェクトをパラメーターとして渡すコレクションの add メソッドを呼び出します。

一部のパラメーターオブジェクトには、固有の複雑なパラメーターもあります。

例3.6 複雑なパラメーター

この例では、フルバージョン 4.0 互換モードで実行中の NFS データセンターが作成されます。これを実行するには、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、これは、作成するデータセンターのパラメーターが含まれる ovirtsdk.xml.params.DataCenter オブジェクトのインスタンスを作成する際にパラメーターとして使用されます。その後、リソースは datacenter 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、それが置かれているコレクションから取得し、必要なパラメーターを変更し、リソースの 更新 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、取得したリソースの 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

この例では、DemoDataCenter という名前のデータセンターの説明が更新されています。

```
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、リソースが含まれるコレクションから取得し、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

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

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

3.9. エラーの処理

エラーが発生すると、ソフトウェア開発キットは例外を使用して強調表示します。ソフトウェア開発キットは、Python インタープリター自体で定義されているものに加えて、例外タイプを定義します。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ときに発生します。

DisconnectedError

明示的に切断された後に SDK の使用を試みると発生します。

ImmutableError

SDK インスタンスがすでに同じドメインに存在する間に SDK を開始すると発生します。SDK バージョン 3.2 以降に適用されます。

NoCertificatesError

CA が指定されておらず、--insecure が False の場合に発生します。

RequestError

あらゆる種類の oVirt サーバーエラーが発生しました。

UnsecuredConnectionAttemptError

サーバーが HTTPS の実行中に HTTP プロトコルを使用すると発生します。

MissingParametersError

id または name を指定せずに get （）メソッドを使用しようとすると発生します。

これらの例外は、他の Python 例外のようにキャッチし、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 を使用して生成されたドキュメントは、以下のモジュールで利用できます。ドキュメントは、ovirt-engine-sdk-python パッケージで提供されます。

- 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バージョンを表示します。

```
$ pydoc [MODULE]
```