Red Hat Virtualization 4.4

『Java SDK Guide』

Red Hat Virtualization Java SDK の使用

Last Updated: 2021-11-25

Red Hat Virtualization 4.4 『Java SDK Guide』

Red Hat Virtualization Java 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 Java ソフトウェア開発キットのバージョン 4 をインストールし、使用する方法を説明します。

目次

第1章 概要

Java ソフトウェア開発キットのバージョン 4 は、Java ベースのプロジェクトで Red Hat Virtualization Manager と対話できるよう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、プロジェクトに追加すると、管理タスクの高レベルの自動化でさまざまな機能にアクセスできます。

注記

SDK のバージョン 3 はサポート対象外になりました。詳細は、本書の RHV 4.3 バージョンを参照してください。

1.1. 前提条件

Java 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 8 がインストールされているシステム。Server および Workstation バリアントの両方がサポートされます。
- Red Hat Virtualization の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。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します。たとえば、Red Hat Virtualization 4.3 を使用している場合は、V4 Java ソフトウェア開発キットを使用します。

1.2. JAVA SOFTWARE DEVELOPMENT KIT のインストール

Java ソフトウェア開発キットと付随ドキュメントをインストールします。

Java Software Development Kit のインストール

1. リポジトリーを有効にします。

```
# subscription-manager repos \
    --enable=rhel-8-for-x86_64-baseos-rpms \
    --enable=rhel-8-for-x86_64-appstream-rpms \
    --enable=rhv-4.4-manager-for-rhel-8-x86_64-rpms\
    --enable=jb-eap-7.4-for-rhel-8-x86_64-rpms
```

2. **pki-deps** モジュールを有効にします。

```
# dnf module -y enable pki-deps
```

3. Java SDK V4 に必要なパッケージをインストールします。

```
# dnf install java-ovirt-engine-sdk4
```

V4 Java ソフトウェア開発キットおよび付随ドキュメントは、**/usr/share/java/java-ovirt-engine-sdk4** ディレクトリーにダウンロードされ、Java プロジェクトに追加できます。

1.3. 依存関係

Java アプリケーションで Java ソフトウェア開発キットを使用するには、以下の JAR ファイルをこれらのアプリケーションのクラスパスに追加する必要があります。

- **commons-beanutils.jar**
- **commons-codec.jar**
- **httpclient.jar**
- **httpcore.jar**
- **jakarta-commons-logging.jar**
- **log4j.jar**

これらの JAR ファイルを提供するパッケージは、**ovirt-engine-sdk-java** パッケージへの依存関係としてインストールされます。デフォルトでは、これらは Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 システムの **/usr/share/java** ディレクトリーで利用できます。

1.4. SSL の設定

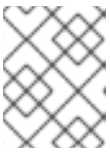
Red Hat Virtualization Manager Java SDK は、Java Secure Socket Extension(JSSE)を使用した HTTP over Secure Socket Layer Security(SSL)および IETF Transport Layer Security(TLS)プロトコルに完全に対応します。JSSE は、バージョン 1.4 の時点で Java 2 プラットフォームに統合され、追加設定なしで Java SDK で動作します。以前の Java 2 バージョンでは、JSSE を手動でインストールして設定する必要があります。

1.4.1. SSL の設定

以下の手順では、Java SDK を使用して SSL を設定する方法の概要を説明します。

SSL の設定

1. Red Hat Virtualization Manager が使用する証明書をダウンロードします。



注記

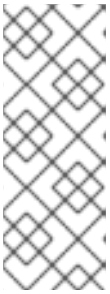
デフォルトでは、Red Hat Virtualization Manager が使用する証明書の場所は **/etc/pki/ovirt-engine/ca.pem にあります**。

2. トラストストアを作成します。

```
$ keytool -import -alias "server.crt truststore" -file ca.crt -keystore server.truststore
```

3. **Api** または **Connection** オブジェクトのインスタンスを構築する際に **trustStoreFile** および **trustStorePassword** 引数を指定します。

```
myBuilder.trustStoreFile("/home/username/server.truststore");
myBuilder.trustStorePassword("p@ssw0rd");
```



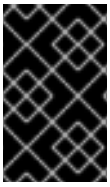
注記

接続の作成時に **trustStoreFile** オプションを指定しないと、Java SDK は、システム変数 **javax.net.ssl.trustStore** によって指定されるデフォルトのトラストストアの使用を試行します。このシステム変数がトラストストアを指定しない場合、Java SDK は $JAVA_HOME/ **lib/security/jssecacerts または $JAVA_HOME/ lib/security/cacerts** で指定されたトラストストアの使用を試みます。

1.4.2. ホストの検証

デフォルトでは、Red Hat Virtualization Manager への接続を開く際に、証明書のホスト名のアイデンティティーが検証されます。**Connection** クラスのインスタンスを構築する際に、以下の引数を渡すと検証を無効にできます。

```
myBuilder.insecure(true);
```



重要

この方法は、セキュリティー上の意思決定で、ホスト ID を検証しないセキュリティー影響について認識している場合を除き、セキュリティー上の理由から、実稼働システムには使用しないでください。

第2章 SOFTWARE DEVELOPMENT KIT の使用

本章では、Java Software Development Kit の使用方法例をいくつか説明します。本章のすべての例では、特に記述がない限り、ソフトウェア開発キットのバージョン 3 を使用します。

2.1. バージョン 4 での RED HAT VIRTUALIZATION MANAGER への接続

Java ソフトウェア開発キットの V4 では、**Connection** クラスは Red Hat Virtualization 環境のオブジェクトを接続および操作するために使用するメインクラスです。このクラスのインスタンスを宣言するには、Connection **Builder** クラスのインスタンスを宣言し、ビルダーメソッドを使用してこのインスタンスに必要な引数を渡して、インスタンスで **ビルド** メソッドを呼び出します。**ビルド** メソッドは、変数に割り当て、後続のアクションの実行に使用できる **Connection** クラスのインスタンスを返します。

以下は、ソフトウェア開発キットのバージョン 4 を使用して、Red Hat Virtualization 環境との接続を作成する単純な Java SE プログラムの例です。

例2.1 Red Hat Virtualization Manager への接続

```
package rhevm;

import org.ovirt.engine.sdk4.Connection;
import org.ovirt.engine.sdk4.ConnectionBuilder;

public class rhevm {

    public static void main(String[] args) {

            ConnectionBuilder myBuilder = ConnectionBuilder.connection()

            .url("https://rhevm.example.com/ovirt-engine/api")
            .user("admin@internal")
            .password("p@ssw0rd")
            .trustStoreFile("/home/username/server.truststore")
            .trustStorePassword("p@ssw0rd");

        try (Connection conn = myBuilder.build()) {

            // Requests

        } catch (Exception e) {

            // Error handling

        }
    }
}
```

この例では、Basic 認証を使用して接続を作成しますが、他の方法も利用できます。**ConnectionBuilder** クラスのインスタンスに渡すことのできるキー引数の一覧は、付録A *ConnectionBuilder メソッド* を参照してください。

2.2. エンティティーの一覧表示

以下の例は、Red Hat Virtualization Manager でエンティティーを一覧表示する方法を概説します。この例では、一覧表示されるエンティティーは仮想マシンであり、Api **クラスの getVM（）** メソッドを使用して一覧表示されます。

エンティティーの一覧表示

1. 一覧表示されるエンティティーの **タイプの一覧** を宣言し、対応する方法を使用してエンティティーのリストを取得します。

```
List<VM> vms = api.getVMs().list();
```

2.3. リソースの属性の変更

以下の例は、リソースの属性を変更する方法の概要を示しています。この例では、変更する属性は 'test' という名前の仮想マシンの説明で、'java_sdk' に変更されています。

リソースの属性の変更

1. 属性を変更するリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. 属性の新しい値を設定します。

```
vm.setDescription("java_sdk");
```

3. 仮想マシンを更新して、変更を適用します。

```
VM newVM = vm.update();
```

2.4. リソースの取得

Java Software Development Kit では、リソースを **name** および **UUID の 2 つの属性で参照できます。** オブジェクトが存在する場合、どちらも指定された属性を持つオブジェクトを返します。

name 属性の値を使用してリソースを取得するには、以下を実行します。

```
VM vm = api.getVMs().get("test");
```

UUID 属性の値を使用してリソースを取得するには、次のコマンドを実行します。

```
VM vm = api.getVMs().get(UUID.fromString("5a89a1d2-32be-33f7-a0d1-f8b5bc974ff6"));
```

2.5. リソースの追加

以下の例は、Red Hat Virtualization Manager にリソースを追加する 2 つの方法を説明します。この例では、追加するリソースは仮想マシンです。

例 1

この例では、**VM** クラスのインスタンスは、追加する新しい仮想マシンを表すように宣言されています。次に、その仮想マシンの属性を優先値に設定します。最後に、新しい仮想マシンが Manager に追加されます。

```
org.ovirt.engine.sdk.entities.VM vmParams = new org.ovirt.engine.sdk.entities.VM();

vmParams.setName("myVm");
vmParams.setCluster(api.getClusters().get("myCluster"));
vmParams.setTemplate(api.getTemplates().get("myTemplate"));
...
```

```
VM vm = api.getVMs().add(vmParams);
```

例 2

この例では、**VM** クラスのインスタンスは Example 1 と同様に宣言されます。ただし、get **メソッドを使用して** Manager の既存のオブジェクトを参照する代わりに、各属性はその属性のインスタンスを宣言して参照されます。最後に、新しい仮想マシンが Manager に追加されます。

```
org.ovirt.engine.sdk.entities.VM vmParams = new org.ovirt.engine.sdk.entities.VM();

vmParams.setName("myVm");
org.ovirt.engine.sdk.entities.Cluster clusterParam = new Cluster();
clusterParam.setName("myCluster");
vmParams.setCluster(clusterParam);
org.ovirt.engine.sdk.entities.Template templateParam = new Template();
templateParam.setName("myTemplate");
vmParams.setTemplate(templateParam);
...
```

```
VM vm = api.getVMs().add(vmParams);
```

2.6. リソースに対するアクションの実行

以下の例は、リソースに対してアクションを実行する方法を概説します。この例では、「test」という名前の仮想マシンを起動します。

リソースに対するアクションの実行

1. リソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. リソースに送信するアクションパラメーターを宣言します。

```
Action actionParam = new Action();
org.ovirt.engine.sdk.entities.VM vmParam = new org.ovirt.engine.sdk.entities.VM();
actionParam.setVm(vmParam);
```

3. アクションを実行します。

```
Action res = vm.start(actionParam);
```

または、内部メソッドとしてアクションを実行できます。

```
Action res = vm.start(new Action()
{
   {
      setVm(new org.ovirt.engine.sdk.entities.VM());
   }
});
```

2.7. サブリソースの一覧表示

以下の例は、リソースのサブリソースを一覧表示する方法の概要を示しています。この例では、「test」という名前の仮想マシンのサブリソースが一覧表示されます。

サブリソースの一覧表示

1. サブリソースを一覧表示するリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. サブリソースを一覧表示します。

```
List<VMDisk>; disks = vm.getDisks().list();
```

=== サブリソースの取得

以下の例は、リソースのサブリソースを参照する方法の概要を示しています。この例では、「test」という名前の仮想マシンに属する「my disk」の名前を持つディスクが参照されます。

リソースのサブリソースの取得

1. サブリソースが参照するリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. 参照するサブリソースのインスタンスを宣言します。

```
VMDisk disk = vm.getDisks().get("my disk");
```

2.8. リソースへのサブリソースの追加

以下の例は、サブリソースをリソースに追加する方法を概説します。この例では、サイズが「1073741824L」の新規ディスクが、インターフェース「virtio」およびフォーマットが「test」という名前の仮想マシンに追加されます。

サブリソースのリソースへの追加

1. サブリソースを追加するリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. リソースの属性を定義するためのパラメーターを作成します。

```
Disk diskParam = new Disk();
diskParam.setProvisionedSize(1073741824L);
diskParam.setInterface("virtio");
diskParam.setFormat("cow");
```

3. サブリソースを追加します。

```
Disk disk = vm.getDisks().add(diskParam);
```

2.9. サブリソースの変更

以下の例は、サブリソースを変更する方法の概要を示しています。この例では、'test' という名前の仮想マシンに属する名前 'test_Disk1' を持つディスクの名前が 'test_Disk1_updated' に変更されます。

サブリソースの更新

1. サブリソースが変更されるリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. 変更するサブリソースのインスタンスを宣言します。

```
VMDisk disk = vm.getDisks().get("test_Disk1");
```

3. 属性の新しい値を設定します。

```
disk.setAlias("test_Disk1_updated");
```

4. サブリソースを更新します。

```
VMDisk updateDisk = disk.update();
```

2.10. サブリソースでのアクションの実行

以下の例は、サブリソースに対してアクションを実行する方法を概説します。この例では、「test」という名前の仮想マシンに属する「test_Disk1」のディスクが有効になります。

Sub-Resource でのアクションの実行

1. アクションを実行するサブリソースが含まれるリソースのインスタンスを宣言します。

```
VM vm = api.getVMs().get("test");
```

2. サブリソースのインスタンスを宣言します。

```
VMDisk disk = vm.getDisks().get("test_Disk1");
```

3. サブリソースに送信するアクションパラメーターを宣言します。

```
Action actionParam = new Action();
```

4. アクションを実行します。

```
Action result = disk.activate(actionParam);
```

付録A CONNECTIONBUILDER メソッド

以下の表は、Java ソフトウェア開発キットの V4 で使用される **ConnectionBuilder** クラスで使用できる主要なメソッドの概要を示しています。

表A.1 ConnectionBuilder メソッド

メソッド	引数タイプ	説明
user	**文字列**	Manager に接続するユーザーの名前。**admin@internal** などのユーザー名とドメインの両方を指定する必要があります。この方法は、**password** メソッドと共に使用する必要があります。
password	**文字列**	Manager に接続するユーザーのパスワード。
compress	**ブール値**	Manager がホストされるサーバーからの応答を圧縮するかどうかを指定します。このオプションはデフォルトでは無効になっているため、このオプションを有効にするためにのみこの方法が必要になります。
timeout	**整数**	要求への応答を待つタイムアウト（秒単位）。要求がこの値よりも時間がかかると、リクエストはキャンセルされ、例外が発生します。この引数は任意です。
ssoUrl	**文字列**	Manager がホストされるサーバーのベース URL。たとえば、パスワード認証の場合は **https://server.example.com/ovirt-engine/sso/oauth/token?\grant_type=password&scope=ovirt-app-api** です。
ssoRevokeUrl	**文字列**	SSO revoke サービスのベース URL。このオプションは、外部認証サービスを使用する場合にのみ指定する必要があります。デフォルトでは、この URL は **url** オプションの値から自動的に計算され、SSO トークン取り消しがエンジンの一部である SSO サービスを使用して実行されます。

メソッド	引数タイプ	説明
ssoTokenName	**文字列**	SSO サーバーから返された JSON SSO 応答のトークン名。デフォルトでは、この値は **access_token** です。
insecure	**ブール値**	Manager がホストするサーバーが提供する SSL 証明書のホスト名の検証を有効または無効にします。デフォルトでは、ホスト名のアイデンティティーが検証され、ホスト名が正しくない場合は接続は拒否されます。そのため、このオプションを無効にするだけで、この方法が必要になります。
trustStoreFile	**文字列**	Manager がホストするサーバーにより提示される証明書を検証するために使用される CA 証明書が含まれるファイルの場所を指定します。このメソッドは **trustStorePassword** メソッドと共に使用する必要があります。
trustStorePassword	**文字列**	**trustStorePath** メソッドで指定されたキーストアファイルへのアクセスに使用されるパスワード。