Red Hat Virtualization 4.4

Java SDK ガイド

Red Hat Virtualization Java SDK の使用

Last Updated: 2023-08-22

Red Hat Virtualization 4.4 Java SDK ガイド

Red Hat Virtualization Java SDK の使用

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

法律上の通知

概要

このガイドでは、Red Hat Virtualization Java ソフトウェア開発キットのバージョン 4 をインストールして操作する方法について説明します。

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して、管理タスクの高レベルでの自動化を実現するため、さまざまな機能を利用できます。

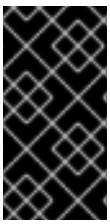
注記

SDK バージョン 3 は、今後サポートされません。詳細は、このガイドの RHV4.3 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。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。お使いの Red Hat Virtualization 環境バージョンに対応するバージョンのソフトウェア開発キットを使用してください。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てください。

1.2. JAVA ソフトウェア開発キットのインストール

Java ソフトウェア開発キットと付属のドキュメントをインストールします。

Java ソフトウェア開発キットのインストール

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と付属のドキュメントは **/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することができます。

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、それらのアプリケーションのクラスパスに次の JAR ファイルを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** ディレクトリーで入手可能です。

1.4. SSL の設定

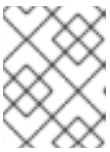
Red Hat Virtualization Manager Java SDK は、HTTP over Secure Socket Layer (SSL) および Java Secure Socket Extension (JSSE) を使用した IETF トランスポート層セキュリティー (TLS) プロトコルを完全にサポートします。JSSE はバージョン 1.4 の時点で Java 2 プラットフォームに統合されており、そのまま Java SDK で動作します。以前の Java 2 バージョンでは、JSSE を手動でインストールして設定する必要があります。

1.4.1. SSL の設定

次の手順は、Java SDK を使用して SSL を設定する方法の概要を示しています。

SSL の設定

1. Red Hat Virtualization Manager で使用され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 によって使用される証明書の場所は **/etc/pki/ovirt-engine/ca.pem** です。

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作成するときに、**trustStoreFile** 引数と **trustStorePassword** 引数を指定します。

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

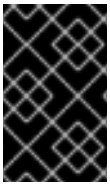
注記

接続の作成時に **trustStoreFile** オプションを指定しない場合、Java SDK は、システム変数 **javax.net.ssl.trustStore** で指定されたデフォルトのトラストストアを使用しようとします。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を使用しようとします。

1.4.2. ホストの検証

デフォルトでは、Red Hat Virtualization Manager への接続を開こうとすると、証明書内のホスト名の ID が検証されます。**Connection** クラスのインスタンスを作成する際に次の引数を渡すことで、検証を無効にできます。

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



重要

この方法は、意図的に使用しており、ホストのアイデンティティーを検証しないことによるセキュリティーへの影響を認識している場合以外は、セキュリティー上の理由から実稼働システムでは使用しないようにください。

第2章 ソフトウェア開発キットの使用

この章では、Java ソフトウェア開発キットの使用方法のいくつかの例について概説します。この章のすべての例では、特に明記されていない限り、ソフトウェア開発キットのバージョン 3 を使用しています。

2.1. バージョン 4 の RED HAT VIRTUALIZATION 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 環境でオブジェクトに接続して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、**ConnectionBuilder** 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から、インスタンスで **build** メソッドを呼び出す必要があります。**build** メソッドは、**Connection** クラスのインスタンスを返します。このインスタンスを変数に割り当てて、後続のアクションを実行するために使用できます。

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 JavaSE プログラムの例になります。

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、他の認証方法を使用することもできます。**ConnectionBuilder** クラスのインスタンスに渡すことができる主要な引数のリストについては、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

次の例は、Red Hat Virtualization Manager でエンティティーを一覧表示する方法の概要を示しています。この例では、リストされるエンティティーは仮想マシンであり、**Api** クラスの **getVMs()** 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. 一覧表示するエンティティーのタイプの **List** を宣言し、対応するメソッドを使用してエンティティーの一覧を取得します。

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

2.3. リソースの属性の変更

次の例は、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、変更される属性は、'test' という名前の仮想マシンの説明であり、これは '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 ソフトウェア開発キットでは、リソースは **name** と **UUID** の 2 つの属性を介して参照できます。オブジェクトが存在する場合、両方と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、以下を実行します。

```
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 つの方法の概要を示しています。これらの例では、追加され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が、追加され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が優先値に設定されます。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");
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が例 1 と同じ方法で宣言されています。ただし、**get** メソッドを使用して Manager 内の既存のオブジェクトを参照するのではなく、各属性はその属性のインスタンスを宣言することによって参照されます。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

次の例は、リソースに対してアクションを実行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が起動されます。

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することもできます。

```
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

次の例は、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ています。

サブリソースの一覧表示

1. サブリソースが一覧表示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. 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

次の例は、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' という名前のディスクが参照されています。

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. 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

次の例は、サブリソースをリソースに追加する方法の概要を示しています。この例では、サイズが '1073741824L'、インターフェイスが 'virtio'、フォーマットが 'cow' の新しいディスクが、'test' という名前の仮想マシンに追加されています。

リソースへのサブリソースの追加

1. サブリソースの追加先とな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

次の例は、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 'test_Disk1' という名前のディスクの名前が '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. 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. サブリソースに対するアクションの実行

次の例は、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 'test_Disk1' という名前のディスクをアクティブ化します。

サブリソースに対するアクションの実行

1. アクションが実行されるサブリソースを含む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

次の表は、Java ソフトウェア開発キットの V4 で使用される **ConnectionBuilder** クラスで利用可能な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	引数のタイプ	説明
user	**String**	Manager に接続するためのユーザーの名前。**admin@internal** など、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。このメソッドは、**password** メソッドと一緒に使用する必要があります。
password	**String**	Manager に接続するためのユーザーのパスワード。
compress	**Boolean**	Manager がホストされているサーバーからのレスポンス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無効になっているため、このメソッドはこのオプションを有効にする場合にのみ必要です。
timeout	**Integer**	リクエストへのレスポンスを待機するためのタイムアウト (秒単位)。リクエストのレスポンスにこの値よりも時間がかかる場合、リクエストはキャンセルされ、例外が出力されます。この引数は任意です。
ssoUrl	**String**	Manager がホストされてい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。

メソッド	引数のタイプ	説明
ssoRevokeUrl	**String**	SSO 取り消しサービスのベース 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るため、エンジンの一部である SSO サービスを使用して SSO トークンの取り消しが実行されます。
ssoTokenName	**String**	SSO サーバーから返された JSONSSO レスポンスのトークン名。デフォルトでは、この値は **access_token** です。
insecure	**Boolean**	Manager がホストされているサーバーによって提示された 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 ID が検証され、ホスト名が正しくない場合は接続が拒否されるため、このメソッドはこのオプションを無効にする場合にのみ必要です。
trustStoreFile	**String**	Manager がホストされているサーバーによって提示された証明書を検証するために使用される CA 証明書を含むファイルの場所を指定します。このメソッドは、**trustStorePassword** メソッドと一緒に使用する必要があります。
trustStorePassword	**String**	**trustStorePath** メソッドで指定されたキーストアファイルにアクセスするために使用されるパスワード。