



Red Hat Virtualization

4.1

Java SDK ガイド

Red Hat Virtualization Java SDK の使用

Red Hat Virtualization Documentation TeamRed Hat

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

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 Java 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

第1章 概要	3
1.1. 前提条件	3
1.2. JAVA ソフトウェア開発キットのインストール	3
1.3. 依存関係	4
1.4. SSL の設定	4
第2章 ソフトウェア開発キットの使用	6
2.1. バージョン 3 での RED HAT ENTERPRISE VIRTUALIZATION MANAGER への接続	6
2.2. バージョン 4 での RED HAT VIRTUALIZATION MANAGER への接続	7
2.3. エンティティの一覧表示	8
2.4. リソースの属性の変更	8
2.5. リソースの取得	9
2.6. リソースの追加	9
2.7. リソースに対するアクションの実行	10
2.8. サブリソースの一覧表示	10
2.9. サブリソースの取得	11
2.10. リソースへのサブリソース追加	11
2.11. サブリソースの変更	11
2.12. サブリソースに対するアクションの実行	12
付録A APIBUILDER メソッド	13
付録B CONNECTIONBUILDER メソッド	15

第1章 概要

Java ソフトウェア開発キットは、Java 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、Java 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

Java 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 Java 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 Java 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使用することができます。

バージョン 4

Java 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 Java 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、Java 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、Java 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

Java 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. 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. 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
# subscription-manager repos --enable=jb-eap-7-for-rhel-7-server-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-java ovirt-engine-sdk-javadoc
```

b. バージョン 4 の場合:

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

Java ソフトウェア開発キットと付属ドキュメントが **/usr/share/java/rhevmsdk-java** の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る状態となりました。

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、これらのアプリケーションのクラスパスに以下の 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が **ovirt-engine-sdk-java** パッケージ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、Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 システムでは、**/usr/share/java** から入手できます。

1.4. SSL の設定

Red Hat Virtualization Manager Java SDK は、Java Secure Socket Extension (JSSE) を使用して、Secure Sockets Layer (SSL) および IETF Transport Layer Security (TLS) プロトコルを介した HTTP の完全なサポートを提供します。JSSE は Java 2 にバージョン 1.4 として実装されており、Java-SDK では追加設定なしで使用することができます。Java 2 の旧バージョンには、JSSE は手でインストールおよび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 を設定する方法を説明します。

SSL の設定

1. Red Hat Virtualization Manager で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 で使用する証明書の場所は `/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構築する際には、**trustStoreFile** および **trustStorePassword** の引数を指定します。

```
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には、Java SDK はシステム変数 **javax.net.ssl.trustStore** で指定したデフォルトのトラストストアを使用するように試行します。このシステム変数がトラストストアを指定しない場合は、Java SDK は **\$JAVA_HOME/lib/security/jssecacerts** または **\$JAVA_HOME/lib/security/cacerts** で指定のトラストストアを使用するように試行します。

1.4.2. ホストの検証

デフォルトでは、Red Hat Virtualization Manager への接続を確立しようとする、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している場合に、以下の引数を渡すとその検証を無効にできます。

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



重要

ホストのアイデンティティーを検証しないことによってもたらされるセキュリティへの影響を認識した上での意識的な決断でない限りは、この方法は、セキュリティ上の理由から、実稼働システムで使用すべきではありません。

第2章 ソフトウェア開発キットの使用

本章には、Java ソフトウェア開発キットの使用法に関する例を記載します。特に記載がない限り、本章の例はすべて、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 3 での RED HAT ENTERPRISE VIRTUALIZATION MANAGER への接続

Java ソフトウェア開発キットバージョン 3 では、**Api** クラスは Red Hat Enterprise Virtualization 環境のオブジェクトに接続して操作するために使用する主要なクラスです。このクラスのインスタンスを宣言するには、**ApiBuilder** クラスのインスタンスを宣言して、**builder** メソッドでこのインスタンスに必要な引数を渡し、このインスタンス上で **build** メソッドを呼び出します。**build** メソッドは、**Api** クラスのインスタンスを返し、その後にこのクラスを変数に割り当てて、後続のアクションを実行する際に使用することができます。

Red Hat Enterprise Virtualization 環境との接続を作成して正常にシャットダウンし、接続を終了するという簡単な Java SE プログラムの例を以下に示します。

例2.1 Red Hat Enterprise Virtualization Manager への接続

```
package rhvm;

import org.ovirt.engine.sdk.Api;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.ovirt.engine.sdk.ApiBuilder;
import org.ovirt.engine.sdk.exceptions.ServerException;
import org.ovirt.engine.sdk.exceptions.UnsecuredConnectionAttemptError;

public class rhvm {

    public static void main(String[] args) {

        Api api = null;

        try {

            ApiBuilder myBuilder = new ApiBuilder()

                .url("https://rhvm.example.com/api")
                .user("admin@internal")
                .password("p@ssw0rd")
                .keyStorePath("/home/username/server.truststore")
                .keyStorePassword("p@ssw0rd");

            api = myBuilder.build();

            api.shutdown();

        } catch (ServerException | UnsecuredConnectionAttemptError |
IOException ex) {
```

```

        Logger.getLogger(Ovirt.class.getName()).log(Level.SEVERE,
null, ex);
    } finally {
        if (api != null) {
            try {
                api.close();
            } catch (Exception ex) {

Logger.getLogger(Ovirt.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
}
}
}

```

この例は、基本的な認証を使用した接続を作成していますが、ほかのメソッドも利用できます。**ApiBuilder** 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一覧については、「[付録 A ApiBuilder メソッド](#)」を参照してください。

注記

Api クラスは、**Autocloseable** インターフェースを実装しない点に注意してください。そのため、上記の例のように **finally** ブロック内の **Api** クラスのインスタンスをシャットダウンして Red Hat Virtualization Manager との接続が正常にシャットダウンされるようにすることを推奨します。

2.2. 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バージョン 4 では、**Connection** クラスは Red Hat Virtualization 環境のオブジェクトに接続して操作するために使用する主要なクラスです。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て、**builder** メソッドでこのインスタンスに必要な引数を渡し、このインスタンス上で **build** メソッドを呼び出します。**build** メソッドは、**Connection** クラスのインスタンスを返し、その後このクラスを変数に割り当てて、後続のアクションを実行する際に使用することができます。

ソフトウェア開発キットバージョン 4 を使用して Red Hat Virtualization 環境の接続を作成する簡単な Java SE プログラムの例を以下に示します。

例2.2 Red Hat Virtualization Manager への接続

```

package rhvm;

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

public class rhvm {

    public static void main(String[] args) {

```

```

        ConnectionBuilder myBuilder =
        ConnectionBuilder.connection()

            .url("https://rhev.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

        }
    }
}

```

この例は、基本的な認証を使用した接続を作成していますが、ほかのメソッドも利用できます。**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一覧については、「[付録B ConnectionBuilder メソッド](#)」を参照してください。

2.3. エンティティの一覧表示

以下の例では、Red Hat Virtualization Manager のエンティティを一覧表示する方法を説明します。この例では、一覧表示するエンティティは仮想マシンで、**Api** クラスの **getVMs()** メソッドを使用して一覧表示します。

エンティティの一覧表示

1. 一覧表示するエンティティのタイプの **List** を宣言し、対応するメソッドを使用してエンティティを一覧表示します。

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

2.4. リソースの属性の変更

以下の例では、リソースの属性を変更する方法について説明します。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、これを「java_sdk」に変更します。

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. 新しい属性値を設定します。

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

3. 変更を適用する仮想マシンを更新します。

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

2.5. 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** および **UUID** の 2 つの属性で参照することができます。そのオブジェクトが存在している場合には、いずれも指定した属性のオブジェクトが返されます。

name 属性値を使用してリソースを取得するには、以下の方法を使用します。

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

UUID 属性値を使用してリソースを取得するには、以下の方法を使用します。

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

2.6. リソースの追加

以下の例では、Red Hat Virtualization Manager にリソースを追加する 2 つの方法について説明します。これらの例では、追加するリソースは仮想マシンです。

例 1

以下の例では、**VM** クラスを宣言して、追加する新規仮想マシンを表現し、次にその仮想マシンの属性を希望する値に設定した後、最後に新規仮想マシンを Manager に追加します。

```
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

以下の例では、**VM** クラスのインスタンスは、例 1 と同じ方法で宣言しますが、**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方法の代わりに、各属性のインスタンスを宣言することによってその属性を参照します。最後に新規仮想マシンを Manager に追加します。

```
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);
```

2.7. リソースに対するアクションの実行

以下の例では、リソースに対してアクションを実行する方法について説明します。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);
```

代わりに、内部メソッドとしてアクションを実行することも可能です。

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

2.8. サブリソースの一覧表示

以下の例では、リソースのサブリソースを一覧表示する方法について説明します。この例では、「test」という名前の仮想マシンのサブリソースを一覧表示します。

サブリソースの一覧表示

1. 一覧表示するサブリソースが含まれてい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();
```

2.9. サブリソースの取得

以下の例では、リソースのサブリソースを参照する方法について説明します。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を参照します。

リソースのサブリソースの取得

1. 参照するサブリソースが含まれてい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");
```

2.10. リソースへのサブリソース追加

以下の例では、リソースにサブリソースを追加する方法を説明します。この例では、サイズが「1073741824L」で、「virtio」インターフェースを使用する「cow」形式の新規ディスクを「test」という名前の仮想マシンに追加します。

リソースへのサブリソースの追加

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. その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);
```

2.11. サブリソースの変更

以下の例では、サブリソースの変更方法を説明します。この例では、「test」という名前の仮想マシンに属する「test_Disk1」という名前のディスクを「test_Disk1_updated」という名前に変更します。

サブリソースの更新

1. 変更するサブリソースが含まれてい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. 新しい属性値を設定します。

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

4. サブリソースを更新します。

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

2.12. サブリソースに対するアクションの実行

以下の例では、サブリソースに対するアクションの実行方法を説明します。この例では、「test」という名前の仮想マシンに属する「test_Disk1」という名前のディスクがアクティブ化されます。

サブリソースに対するアクションの実行

1. アクションを実行する対象となるサブリソースが含まれてい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 APIBUILDER メソッド

以下の表では、Java ソフトウェア開発キット V3 で使用する **ApiBuilder** クラスで利用可能な主要メソッドについて説明します。

表A.1 ApiBuilder メソッド

メソッド	引数タイプ	説明
user	文字列	Manager への接続に使用するユーザー。 admin@internal のように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 メソッドと合わせて使用する必要があります。
password	文字列	Manager への接続に使用するユーザーのパスワード。
sessionID	文字列	Manager への接続に使用するセッションの ID。Manager の認証が済んでいて、セッションが利用できる場合には、ユーザー名とパスワードではなくこの引数を指定することができます。
requestTimeout	整数	要求に応答するまで待機するタイムアウトの値 (秒)。要求への応答時間がこの値よりも長い場合には、要求はキャンセルされ、例外が送出されます。この引数は任意です。
sessionTimeout	整数	Manager に要求が出されない場合にアクティブなセッションを破棄する
persistentAuth	ブール値	cookie を使用した永続的な認証を有効化または無効化します。デフォルトではこのオプションは有効となっているため、このオプションを無効にする場合のみこのメソッドが必要です。

メソッド	引数タイプ	説明
noHostVerification	ブール値	Manager のホスト先のサーバーが公開する SSL 証明書のホスト名の検証を有効化または無効化します。デフォルトでは、ホスト名のアイデンティティは検証され、ホスト名が正しくない場合には接続が拒否されます。そのため、このオプションを無効にする場合のみこのメソッドが必要です。
keyStorePath	文字列	Manager のホスト先のサーバーが公開する証明書の検証に使用する 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、 keyStorePassword メソッドと合わせて使用する必要があります。
keyStorePassword	文字列	keyStorePath メソッドで指定するキーストアファイルにアクセスする際に使用するパスワード。
filter	ブール値	要求を行うユーザーのパーミッションをベースにしたオブジェクトのフィルタリングを有効化または無効化します。デフォルトでは、このオプションは無効になっており、この環境内のオブジェクトがすべてユーザーに表示されます。このメソッドは、要求を行うユーザーにだけ環境内のオブジェクトを表示するように制限する際にのみ必要です。
debug	ブール値	デバッグの出力を有効化または無効化します。デフォルトでは、このオプションは無効になっています。

付録B CONNECTIONBUILDER メソッド

以下の表では、Java ソフトウェア開発キット V4 で使用する **ConnectionBuilder** クラスで利用可能な主要メソッドについて説明します。

表B.1 ConnectionBuilder メソッド

メソッド	引数タイプ	説明
user	文字列	Manager への接続に使用するユーザー。 admin@internal のように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 メソッドと合わせて使用する必要があります。
password	文字列	Manager への接続に使用するユーザーのパスワード。
compress	ブール値	Manager をホストしているサーバーからの応答を圧縮するべきかどうかを指定します。デフォルトでは、このオプションは無効になっているため、このオプションを有効にする場合のみにこのメソッドは必要です。
timeout	整数	要求に応答するまで待機するタイムアウトの値 (秒)。要求への応答時間がこの値よりも長い場合には、要求はキャンセルされ、例外が送出されます。この引数は任意です。
ssoUrl	文字列	Manager をホストするサーバーのベース URL。たとえば、パスワード認証には https://server.example.com/ovirt-engine/sso/oauth/token? \grant_type=password&scope=ovirt-app-api を使用します。

メソッド	引数タイプ	説明
ssoRevokeUrl	文字列	SSO 呼び出しサービスのベース URL。外部認証サービスを使用する場合にのみこのオプションを指定する必要があります。デフォルトでは、engine の一部となっている SSO サービスを使用して SSO トークンの呼び出しが実行されるように、この URL は自動的に url オプションの値から算出されます。
ssoTokenName	文字列	SSO サーバーから返される JSON SSO の応答にあるトークン名。デフォルトでは、この値は access_token です。
insecure	ブール値	Manager のホスト先のサーバーが公開する SSL 証明書のホスト名の検証を有効化または無効化します。デフォルトでは、ホスト名のアイデンティティは検証され、ホスト名が正しくない場合には接続が拒否されます。そのため、このオプションを無効にする場合のみこのメソッドが必要です。
trustStoreFile	文字列	Manager のホスト先のサーバーが公開する証明書の検証に使用する 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 メソッドと合わせて使用する必要があります。
trustStorePassword	文字列	trustStorePath メソッドで指定するキーストアファイルにアクセスする際に使用するパスワード。