Red Hat Virtualization 4.1 Java SDK 指南

使用 Red Hat Virtualization Java SDK

Red Hat Virtualization 文档团队Red Hat

Red Hat Virtualization 4.1 Java SDK 指南

使用 Red Hat Virtualization Java SDK

Red Hat Virtualization 文档团队
Red Hat 出版部
rhev-docs@redhat.com

法律通告

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.

摘要

本指南介绍了如何安装并使用 Red Hat Virtualization Java 软件开发套件版本 3 和版本 4。

目录

第 **1** 章 概述 **3**
1.1. 前提条件 3
1.2. 安装 JAVA 软件开发套件 3
1.3. 依赖性 4
1.4. 配置 SSL 4
1.4.1. 配置 SSL 4
1.4.2. 主机验证 5

第 **2** 章 使用软件开发套件 **6**
2.1. 使用 V3 连接到 RED HAT ENTERPRISE VIRTUALIZATION MANAGER 6
2.2. 使用 V4 连接到 RED HAT VIRTUALIZATION MANAGER 7
2.3. 列出项 8
2.4. 修改资源属性 8
2.5. 创建资源 8
2.6. 添加资源 9
2.7. 在资源上进行操作 9
2.8. 列出子资源 10
2.9. 获得子资源 10
2.10. 为资源添加子资源 11
2.11. 修改子资源 11
2.12. 在子资源上进行操作 11

附录 **A. APIBUILDER** 的方法 **13**

附录 **B. CONNECTIONBUILDER** 的方法 **15**

第 1 章 概述

Java 软件开发套件（Java SDK）是一组类（class）的集合，您可以在基于 Java 的项目中使用它们与 Red Hat Virtualization Manager 进行交互。下载这些类并将其添加到项目中即可使用一系列功能自动完成高级别的管理任务。

Red Hat Virtualization 提供了两个版本的 Java 软件开发套件：

V 3

V3 Java 软件开发套件提供了对 Red Hat Enterprise Virtualization 3.6 最新版本中的 Java 软件开发套件的类（class）和方法（method）结构的后向兼容功能。使用 Red Hat Enterprise Virtualization 3.6 中的 Java 软件开发套件开发的应用程序可以在不需要进行修改的情况下在这个版本中运行。

V 4

V4 Java 软件开发套件提供了一组更新的类和方法。使用 Red Hat Enterprise Virtualization 3.6 中的 Java 软件开发套件开发的应用程序需要进行更新后才可以在这个版本中运行。

以上两个版本的 Java 软件开发套件都可以在 Red Hat Virtualization 环境中使用。这需要安装相关的软件包，并把所需的程序库添加到 Java 项目中。

1.1. 前提条件

为了安装 Java 软件开发套件，需要满足以下前提条件：

- 安装了 Red Hat Enterprise Linux 7（Server 版和 Workstation 版都被支持）的系统。
- 订阅了 Red Hat Virtualization 权利。

重要

该软件开发套件是 Red Hat Virtualization REST AP 的接口，因此必须使用与 Red Hat Virtualization 环境对应的软件开发套件版本。例如：如果使用 Red Hat Virtualization 4.1，则必须使用为 4.1 版本设计的软件开发套件版本。

1.2. 安装 JAVA 软件开发套件

安装 Java 软件开发套件及随附文档。

安装 Java 软件开发套件（SDK）

1. 启用所需的频道：

```
# 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. 安装所需软件包：

a. V3:

```
# yum install ovirt-engine-sdk-java ovirt-engine-sdk-
javadoc
```

b. V4:

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

Java 软件开发套件和相关的文档被下载到 **/usr/share/java/rhevm-sdk-java** 目录中，它们现在可以被添加到所需的 Java 项目中。

1.3. 依赖性

在 Java 应用程序中使用 Java 软件开发套件时，必须在那些应用程序的类路径中添加以下 JAR 文件：

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

将提供这些 JAR 文件的软件包作为 **`ovirt-engine-sdk-java`** 软件包的相依性软件包安装。默认情况下，它们位于 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 系统的 **/usr/share/java** 目录中。

1.4. 配置 SSL

Red Hat Virtualization Manager Java SDK 使用 JSSE（Java Secure Socket Extension）提供了对 HTTP over SSL（Secure Socket Layer）和 IETF TLS（Transport Layer Security）的支持功能。JSSE 已经被集成到 Java 2 平台的 1.4 版本中，并可以直接和 Java SDK 一起使用。而在其它较老的 Java 2 版本中，JSSE 需要被手工安装和配置。

1.4.1. 配置 SSL

以下过程展示了如何配置 SSL 来使用 Java SDK。

配置 SSL

1. 下载 Red Hat Virtualization Manager 使用的证书：

注意

在默认情况下，Red Hat Virtualization Manager 使用的证书位于 **`/etc/pki/ovirt-engine/ca.pem`**。

2. 创建一个 truststore：

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

3. 在构建一个 **Api** 或 **Connection** 对象实例时，指定 **trustStoreFile** 和 **trustStorePassword** 参数。

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

注意

如果在创建一个连接时没有指定 **trustStoreFile** 选项，Java SDK 会尝试使用系统变量 **javax.net.ssl.trustStore** 所指定的默认 truststore。如果系统变量没有指定 truststore，Java SDK 会尝试使用在 **$JAVA_HOME/lib/security/jssecacerts** 或 **$JAVA_HOME/lib/security/cacerts** 中指定的 truststore。

1.4.2. 主机验证

在默认情况下，当试图创建一个到 Red Hat Virtualization Manager 的连接时，证书中的主机名的身份会被验证。如需禁用身份验证的过程，则在构建 **Connection** 类的实例时使用以下参数：

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

重要

因为安全的原因，这个方法一般不应该在生产环境中使用。如要使用这个方法，您需要了解不对主机身份进行验证可能会带来的安全风险。

第 2 章 使用软件开发套件

本章包括了一组展示如何使用 Java 软件开发套件的示例。除非特别声明，本章中的示例都使用软件开发套件的 V3 版本。

2.1. 使用 V3 连接到 RED HAT ENTERPRISE VIRTUALIZATION MANAGER

在 Java 软件开发套件的 V3 版本中，**Api** 类是连接并处理 Red Hat Enterprise Virtualization 环境中的项的主要类。为了声明这个类的一个实例，您需要声明 **ApiBuilder** 类的一个实例，并使用 builder 方法把需要的参数传递到这个实例。**build** 方法会返回 **Api** 类的一个实例，然后您就可以为它分配变量，并执行相关的操作。

以下示例是一个简单的 Java SE 程序，它创建一个到 Red Hat Enterprise Virtualization 环境的连接，然后安全地关闭并断开连接。

例 2.1. 连接到 Red Hat Enterprise Virtualization Manager

```
package rhevm;

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 rhevm {

 public static void main(String[] args) {

 Api api = null;

 try {

 ApiBuilder myBuilder = new ApiBuilder()

 .url("https://rhevm.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);
 }
 }
 }
 }
}
```

这个示例创建了一个使用基础验证的连接，但您也可以使用其它方法。如需了解可以传递给 **ApiBuilder** 类实例的参数列表，请参阅 附录 A, *ApiBuilder 的方法*。

注意

请注意，**Api** 类不采用 **Autocloseable** 接口。因此建议在 **finally** 块中关闭 **Api** 类实例，以保证可正常关闭与 Red Hat Enterprise Virtualization Manager 的连接。

2.2. 使用 V4 连接到 RED HAT VIRTUALIZATION MANAGER

在 Java 软件开发套件的 V4 版本中，**Connection** 类是连接并处理 Red Hat Virtualization 环境中的项的主要类。为了声明这个类的一个实例，您需要声明 **ConnectionBuilder** 类的一个实例，并使用 builder 方法把需要的参数传递到这个实例。**build** 方法会返回 **Connection** 类的一个实例，然后您就可以为它分配变量，并执行相关的操作。

以下是一个简单的 Java SE 程序示例，它使用软件开发套件的 V4 版本创建一个到 Red Hat Virtualization 环境的连接。

例 2.2. 连接到 Red Hat Virtualization Manager

```
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

        }
    }
}
```

这个示例创建了一个使用基础验证的连接，但您也可以使用其它方法。如需了解可以传递给 **ConnectionBuilder** 类实例的参数列表，请参阅 附录 B, *ConnectionBuilder 的方法*。

2.3. 列出项

以下示例展示了如何列出 Red Hat Virtualization Manager 中的项。在这个示例中，所列出的项是虚拟机，使用 **Api** 类的 **getVMs()** 方法可以列出它们。

列出项

1. 声明一个要被列出项的类型的 **List**，使用相应方法获得项列表：

```
List&lt;VM&gt; vms = api.getVMs().list();
```

2.4. 修改资源属性

以下示例展示了如何修改资源的属性。在这个示例中，要被修改的属性是名为 'test' 的虚拟机的描述信息，它被改为 'java_sdk'。

修改资源的属性

1. 声明一个需要被更改属性的资源的实例。

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

2. 为属性设置新的值：

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

3. 更新虚拟机来应用所做的修改：

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

2.5. 创建资源

在 Java 软件开发套件中，资源可以通过两个属性 - **name** 和 **UUID** 来代表。

使用 **name** 属性获得一个资源：

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

■

使用 **UUID** 属性获得一个资源：

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

2.6. 添加资源

以下示例展示了两种为 Red Hat Virtualization Manager 添加资源的方法。在这些示例中，被添加的资源是虚拟机。

示例 1

在这个示例中，声明了一个代表了新添加的虚拟机的 **VM** 类。接下来，为虚拟机的属性设置所需的值。最后，虚拟机被添加到 Manager。

```
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);
```

示例 2

在这个示例中，首先和示例 1 一样，声明了一个 **VM** 类实例。不同的是，它没有使用 **get** 方法来获得 Manager 中存在的项，而是为每个属性声明一个属性实例。最后，虚拟机被添加到 Manager。

```
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. 在资源上进行操作

以下示例展示了如何对资源进行操作。在这个示例中，一个名为 'test' 的虚拟机被启动。

在资源上执行一个操行

1. 声明资源的一个实例：

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

2. 声明要发送到资源上的操作参数：

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

3. 执行操作：

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

另外，还可以作为一个内部方法来执行操作：

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

2.8. 列出子资源

以下示例展示了如何列出资源的子资源。在这个示例中，列出了一个名为 'test' 的虚拟机的子资源。

列出子资源

1. 声明一个需要被列出子资源的资源实例。

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

2. 列出子资源：

```
List&lt;VMDisk&gt; disks = vm.getDisks().list();
```

2.9. 获得子资源

以下示例展示了如何引用资源的子资源。在这个示例中，属于名为 'test' 的虚拟机的、名为 'my disk' 的磁盘被引用。

获得一个资源的子资源

1. 声明一个需要被引用的子资源所属资源的实例。

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

2. 声明需要被引用的子资源的一个实例：

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

2.10. 为资源添加子资源

以下示例展示了如何为资源添加子资源。在这个示例中，为名为 'test' 的虚拟机添加了一个大小为 '1073741824L'、接口是 'virtio'、格式是 'cow' 的新磁盘。

为一个资源添加子资源

1. 声明一个需要添加子资源的资源实例。

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

2. 创建参数来定义资源的属性：

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

3. 添加子资源：

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

2.11. 修改子资源

以下示例展示了如何修改子资源。在这个示例中，属于名为 'test' 的虚拟机的、名为 'test_Disk1' 的磁盘被改名为 'test_Disk1_updated'。

更新子资源

1. 声明一个需要被更改的子资源所属资源的实例。

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

2. 声明需要被更改的子资源的实例：

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

3. 为属性设置新的值：

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

4. 更新子资源：

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

2.12. 在子资源上进行操作

以下示例展示了如何修改子资源。在这个示例中，属于名为 'test' 的虚拟机的、名为 'test_Disk1' 的磁盘被激活。

在子资源上执行一个操行

1. 声明需要进行操作的子资源所属资源的一个实例。

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

2. 声明子资源的一个实例：

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

3. 声明要发送到子资源上的操作参数：

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

4. 执行操作：

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

附录 A. APIBUILDER 的方法

下表列出了 Java 软件开发套件 V3 版本中 **`ApiBuilder`** 类可以使用的关键方法（method）。

表 A.1. ApiBuilder 的方法

方法	参数类型	描述
`user`	**`String`**	连接到 Manager 的用户的名称。您需要知道用户名和域（如 **`admin@internal`**）。这个方法需要和 **`password`** 方法一起使用。
`password`	**`String`**	连接到 Manager 的用户的密码。
`sessionID`	**`String`**	连接到 Manager 的会话 ID。如果已和 Manager 进行了验证并有可用的会话，则可以指定这个参数来替代用户名和密码。
`requestTimeout`	**`Integer`**	等待请求响应的超时时间（以秒为单位）。如果请求在这个设置的时间内没有获得响应，则这个请求会被取消，并产生一个异常（exception）。这个参数是可选的。
`sessionTimeout`	**`Integer`**	会话超时时间（以分钟为单位）。当经过这个指定的时间后还没有对 Manager 的请求，则活跃的会话会被删除。这个参数是可选的。
`persistentAuth`	**`Boolean`**	指定是否使用 cookies 实现持久性验证。这个选项在默认情况下被启用，您只需要在禁用这个选项时才需要使用它。

方法	参数类型	描述
noHostVerification	**Boolean**	指定是否验证运行 Manager 的服务器上的 SSL 证书中的主机名。在默认情况下，主机名会被验证，如果主机名不正确，连接会被拒绝。只有在需要禁用这个选项时才使用这个方法。
keyStorePath	**String**	指定包括一个 CA 证书的文件位置。这个证书被用来验证运行 Manager 的服务器上的证书。这个方法需要和 **keyStorePassword** 方法一起使用。
keyStorePassword	**String**	访问在 **keyStorePath** 方法中指定的 keystore 文件时使用的密码。
filter	**Boolean**	启用或禁用基于发出请求的用户的权限对项进行过滤的功能。它的默认设置是禁用，这会允许任何用户查看环境中的所有项。如果需要限制环境中的项只对有权限的用户可见，可以使用这个方法。
debug	**Boolean**	启用或禁用 debug 输出。在默认情况下，这个选项被禁用。

附录 B. CONNECTIONBUILDER 的方法

下表列出了 Java 软件开发套件 V4 版本中 **ConnectionBuilder** 类可以使用的关键方法（method）。

表 B.1. ConnectionBuilder 的方法

方法	参数类型	描述
user	**String**	连接到 Manager 的用户的名称。您需要知道用户名和域（如 **admin@internal**）。这个方法需要和 **password** 方法一起使用。
password	**String**	连接到 Manager 的用户的密码。
compress	**Boolean**	指定从运行 Manager 的服务器上返回的信息是否要压缩。这个选项在默认情况下被禁用，您只有在需要启用这个选项时才使用这个方法。
timeout	**Integer**	等待请求响应的超时时间（以秒为单位）。如果请求在这个设置的时间内没有获得响应，则这个请求会被取消，并产生一个异常（exception）。这个参数是可选的。
ssoUrl	**String**	运行 Manager 的服务器的基本 URL。例如，**https://server.example.com/ovirt-engine/sso/oauth/token?\grant_type=password&scope=ovirt-app-api** 代表密码验证。

方法	参数类型	描述
ssoRevokeUrl	**String**	SSO revoke 服务的基本 URL。只有在使用外部验证服务时才使用这个选项。在默认情况下，这个 URL 的值会根据 **url** 选项的值自动计算，SSO token revoke 会使用作为引擎一部分的 SSO 服务进行。
ssoTokenName	**String**	SSO 服务器返回的 JSON SSO 响应数据中的 token 名。在默认情况下，这个值是 **access_token**。
insecure	**Boolean**	指定是否验证运行 Manager 的服务器上的 SSL 证书中的主机名。在默认情况下，主机名会被验证，如果主机名不正确，连接会被拒绝。只有在需要禁用这个选项时才使用这个方法。
trustStoreFile	**String**	指定包括一个 CA 证书的文件位置。这个证书被用来验证运行 Manager 的服务器上的证书。这个方法需要和 **trustStorePassword** 方法一起使用。
trustStorePassword	**String**	访问在 **trustStorePath** 方法中指定的 keystore 文件时使用的密码。