Red Hat Virtualization 4.4

Java SDK 指南

使用 Red Hat Virtualization Java SDK

Last Updated: 2021-09-16

Red Hat Virtualization 4.4 Java SDK 指南

使用 Red Hat Virtualization Java SDK

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.

法律通告

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Java_SDK_Guide.ent file |.

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, the Red Hat 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 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

本指南描述了如何安装和使用 Red Hat Virtualization Java 软件开发工具包的版本 4。

目录

第 1 章 概述

Java 软件开发套件的版本 4 是一组类，允许您在基于 Java 的项目中与 Red Hat Virtualization Manager 进行交互。通过下载这些课程并将其添加到项目中，您可以访问一系列功能，以实现高层次的管理任务自动化。

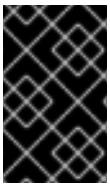
注意

SDK 的版本 3 不再被支持。如需更多信息，请参阅本指南的 RHV 4.3 版本。

1.1. 先决条件

要安装 Java 软件开发套件，您必须有：

- 安装 Red Hat Enterprise Linux 8 的系统。支持 Server 和 Workstation 变体。
- Red Hat Virtualization 权利订阅。



重要

软件开发套件是 Red Hat Virtualization REST API 的接口。使用与 Red Hat Virtualization 环境版本对应的软件开发套件的版本。例如，如果您使用 Red Hat Virtualization 4.3，请使用 V4 Java 软件开发工具包。

1.2. 安装 JAVA 软件开发套件

安装 Java 软件开发套件和附带的文档。

安装 Java 软件开发套件

1. 启用存储库：

```
# 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-7.3-for-rhel-8-x86_64-rpms
```

2. 启用 **pki-deps** 模块。

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

3. 安装 Java SDK V4 所需的软件包：

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

V4 Java 软件开发工具包和附带的文档已下载到 **/usr/share/java/java-ovirt-engine-sdk4** 目录中，并可以添加到 Java 项目中。

1.3. 依赖项

要在 Java 应用程序中使用 Java 软件开发套件，您必须将这些 JAR 文件添加到这些应用程序的类路径中：

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

提供这些 JAR 文件的软件包安装为 **ovirt-engine-sdk-java** 软件包的依赖软件包。默认情况下，它们位于 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 系统上的 **/usr/share/java** 目录中。

1.4. 配置 SSL

红帽虚拟化管理器 Java SDK 完全支持使用 Java 安全套接字扩展(JSSE)的 HTTP over 套接字层(SSL)和 IETF 传输层安全(TLS)协议。从 1.4 版开始，JSSE 已集成到 Java 2 平台中，开箱即用 Java SDK。在较早的 Java 2 版本中，必须手动安装和配置 JSSE。

1.4.1. 配置 SSL

以下流程概述了如何使用 Java SDK 配置 SSL。

配置 SSL

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

注意

默认情况下，Red Hat Virtualization Manager 使用的证书的位置位于 **/etc/pki/ovirt-engine/ca.pem** 中。

2. 创建信任存储：

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

3. 在构建 **Api** 或 **Connection** 对象的实例时，指定 **trustStoreFile** 和 **trustStorePassword** 参数：

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

注意

如果您在创建连接时没有指定 **trustStoreFile** 选项，Java SDK 会尝试使用系统变量 **javax.net.ssl.trustStore** 指定的默认信任存储。如果此系统变量没有指定信任存储，Java SDK 会尝试使用 **$JAVA_HOME/lib/security/jssecacerts** 或 **$JAVA_HOME/lib/security/cacerts** 中指定的信任存储。

1.4.2. 主机验证

默认情况下，当尝试打开到 Red Hat Virtualization Manager 的连接时，证书中主机名的身份会被验证。您可以在构建 **Connection** 类的实例时传递以下参数来禁用验证：

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

重要

由于安全原因，这种方法不应用于生产系统，除非是一个明智的决定，并且您清楚不验证主机身份所带来的安全隐患。

第 2 章 使用软件开发套件

本章概述了如何使用 Java 软件开发套件的几个示例。除非另有说明，否则本章中的所有示例都使用软件开发工具包的版本 3。

2.1. 连接到版本 4 中的 RED HAT VIRTUALIZATION MANAGER

在 Java 软件开发套件的 V4 中，**连接** 类是您用于在 Red Hat Virtualization 环境中连接和操作对象的主要类。若要声明此类的实例，您必须声明 **ConnectionBuilder** 类的实例，使用构建器方法将必要的参数传递给此实例，然后在实例上调用 **构建** 方法。**构建** 方法返回 **Connection** 类的实例，您可以将其分配到变量并使用 执行后续操作。

以下是一个简单的 Java SE 程序示例，它使用软件开发套件的版本 4 与 Red Hat Virtualization 环境建立连接：

例 2.1. 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

      }
    }
}
```

本例使用基本身份验证创建连接，但也提供其他方法。有关可传递给 **ConnectionBuilder** 类实例的关键参数列表，请参阅 附录 A, *ConnectionBuilder 方法*。

2.2. 列出实体

以下示例概述了如何列出 Red Hat Virtualization Manager 中的实体。在本例中，列出的实体是虚拟机，使用 **Api** 类的 **getVMs（）** 方法列出。

列出实体

1. 声明要列出的实体类型 **列表**，并使用对应方法获取实体列表：

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

2.3. 修改资源属性

以下示例概述了如何修改资源的属性。在本例中，要修改的属性是虚拟机的描述，名称为 'test'，它被更改为 'java_sdk'。

修改资源的属性

1. 声明要修改其属性的资源实例：

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

2. 设置属性的新值：

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

3. 更新虚拟机以应用更改：

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

2.4. 获取资源

在 Java 软件开发套件中，可以通过两个属性来引用资源： **name** 和 **UUID**。如果对象存在，两者均返回具有指定属性的对象。

使用 **name** 属性的值获取资源：

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

使用 **UUID** 属性的值获取资源：

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

2.5. 添加资源

以下示例概述了向红帽虚拟化管理器添加资源的两种方式。在这些示例中，要添加的资源是虚拟机。

例 1

在本例中，**VM** 类的实例被声明为代表要添加新的虚拟机。接下来，该虚拟机集的属性设置为首选值。最后，新虚拟机将添加到 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

在本例中，**虚拟机** 类实例的声明方式与示例 1 相同。不过，通过声明该属性的实例来引用各个属性，而不是使用 **get** 方法引用管理器中的现有对象。最后，新虚拟机将添加到 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);
```

2.6. 对资源执行操作

以下示例概述了如何对资源执行操作。在本例中，将启动名为 'test' 的虚拟机。

对资源执行操作

1. 声明资源实例：

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

2. 声明要发送到资源的 action 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，您还可以以内部方法执行操作：

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

2.7. 列出子资源

以下示例概述了如何列出资源的子资源。本例中列出了名为 'test' 的虚拟机的子资源。

列出子资源

1. 声明要列出其子资源的资源实例：

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

2. 列出子资源：

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

=== 获取子资源

以下示例概述了如何引用资源的子资源。在本例中，引用名称为 'my disk' 的磁盘，它属于虚拟机。

获取资源的子资源

1. 声明要引用其子资源的资源实例：

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

2. 声明要引用的子资源实例：

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

2.8. 在资源中添加子资源

以下示例概述了如何向资源添加子资源。在本例中，大小为 '1073741824L'、接口 'virtio' 和 format 'cow' 的新磁盘被添加到名为 'test' 的虚拟机中。

将子资源添加到资源

1. 声明要将子资源添加到的资源实例：

```
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);
```

2.9. 修改子资源

以下示例概述了如何修改子资源。在本例中，名称为 'test_Disk1' 的磁盘名称属于名为 'test' 的虚拟机，它被改为 'test_Disk1_updated'。

更新子资源

1. 声明要修改其子资源的资源实例：

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

2. 声明要修改的子资源实例：

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

3. 设置属性的新值：

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

4. 更新子资源：

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

2.10. 对子资源执行操作

以下示例概述了如何对子资源执行操作。在本例中，激活名称为 'test_Disk1' 的磁盘，它属于名为 'test' 的虚拟机。

对子资源执行操作

1. 声明一个资源实例，其中包含要对其执行该操作的子资源：

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

2. 声明子资源实例：

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

3. 声明要发送到子资源的操作参数：

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

4. 执行操作：

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

附录 A. CONNECTIONBUILDER 方法

下表概述了可供 Java 软件开发工具包 V4 中使用的 **ConnectionBuilder** 类使用的主要方法。

表 A.1. ConnectionBuilder 方法

方法	参数类型	描述
user	**字符串**	要连接到 Manager 的用户名称。您必须同时指定用户名和域，如 **admin@internal**。此方法必须与 **password** 方法一同使用。
password	**字符串**	要连接到管理器的用户密码。
compress	**布尔值**	指定是否应该压缩来自托管管理器的服务器的响应。默认情况下禁用这个选项，因此仅需要启用这个选项。
timeout	**整数**	等待响应请求的超时时间（以秒为单位）。如果响应请求的用时超过这个值，则请求将被取消，并抛出异常。这个参数是可选的。
ssoUrl	**字符串**	托管管理器的服务器的基本 URL。例如：**https://server.example.com/ovirt-engine/sso/oauth/token?\grant_type=password&scope=ovirt-app-api** 用于密码身份验证。
ssoRevokeUrl	**字符串**	SSO 的基础 URL 撤销服务。仅当您使用外部身份验证服务时，才需要指定这个选项。默认情况下，此 URL 会自动从 **url** 选项的值计算，以便通过作为引擎一部分的 SSO 服务来执行 SSO 令牌。
ssoTokenName	**字符串**	从 SSO 服务器返回的 JSON SSO 响应中的令牌名称。默认情况下，这个值是 **access_token**。
insecure	**布尔值**	启用或禁用在托管管理器的服务器提供的 SSL 证书中主机名的验证。默认情况下，会验证主机名的身份，如果主机名不正确，连接将被拒绝，因此仅需要使用此方法禁用此选项。

方法	参数类型	描述
trustStoreFile	**字符串**	指定包含 CA 证书的文件位置，用于验证托管 Manager 的服务器所提供的证书。这个方法必须与 **trustStorePassword** 方法一同使用。
trustStorePassword	**字符串**	用于访问 **trustStorePath** 方法中指定的密钥存储文件的密码。