Red Hat Virtualization 4.4

Java SDK 指南

使用 Red Hat Virtualization Java SDK

Last Updated: 2023-09-08

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

摘要

本指南论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 是一套类，可用于在基于 Java 的项目中与 Red Hat Virtualization Manager 交互。通过下载这些类并将它们添加到您的项目中，您可以访问一系列功能，以实现高级管理任务的自动化。

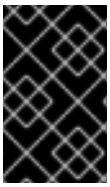
注意

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。支持服务器和 Workstation 变体。
- Red Hat Virtualization 权利订阅。



重要

软件开发套件是 Red Hat Virtualization REST API 的界面。使用与您 Red Hat Virtualization 环境版本对应的软件开发组件版本。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-7.4-for-rhel-8-x86_64-rpms
```

2. 启用 **pki-deps** 模块。

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

3. 为 Java SDK V4 安装所需的软件包：

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

V4 Java 软件开发套件和附带的文档已下载到 **/usr/share/java/java-ovirt-engine-sdk4** 目录中，并可添加到 Java 项目中。

1.3. 依赖项

要在 Java 应用程序中使用 Java 软件开发工具包，您必须将以下 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 文件的软件包作为依赖项安装到 **ovirt-engine-sdk-java** 软件包。默认情况下，它们位于 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 系统的 **/usr/share/java** 目录中。

1.4. 配置 SSL

Red Hat Virtualization Manager Java SDK 使用 Java 安全套接字扩展(JSSE)为 HTTP over Secure Socket Layer(SSL)和 IETF 传输层安全(TLS)协议提供全面支持。自 1.4 版以来，JSSE 已集成到 Java 2 平台，并开箱即用了 Java SDK。在以前的 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. 下载由 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未指定 **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。如果这个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/security/cacerts 中指定的信任存储。**

1.4.2. 主机验证

默认情况下，在尝试打开与 Red Hat Virtualization Manager 的连接时，会验证证书中的主机名身份。您可以通过在构建 **Connection** 类的实例时传递以下参数来禁用验证：

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

重要

出于安全考虑，这个方法不应该用于生产系统，除非它有明确的决定，您了解到不会验证主机身份的安全隐患。

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。本章的所有示例都使用软件开发工具包的版本 3，除非另有说明。

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，**Connection** 类是用于连接到并操作 Red Hat Virtualization 环境中的对象的主要类。要声明这个类的实例，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。**build** 方法返回 **Connection** 类的实例，然后您可以分配给一个变量并使用 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 创建与 Red Hat Virtualization 环境的连接：

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，但也有其他方法可用。有关可传递给 **ConnectionBuilder** 类实例的关键参数的列表，请参阅 附录 A, *ConnectionBuilder Methods*。

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 **getVM（）** 方法列出。

列出实体

1. 声明要列出的实体类型列表，并使用对应的方法获取实体列表：

```
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'，它变为 'java_sdk'。

修改资源的属性

1. 声明要修改的属性的资源实例：

```
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，可通过以下两个属性引用资源： name 和 UUID。如果对象存在，则返回带有指定属性的对象。

要使用 name 属性的值获取资源：

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

要使用 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

以下示例概述了在 Red Hat Virtualization Manager 中添加资源的两种方法。在这些示例中，要添加的资源是虚拟机。

示例 1

在本例中，声明了 VM 类的实例，以代表要添加的新虚拟机。接下来，该虚拟机的属性设置为首选值。最后，新虚拟机会添加到 Manager 中。

```
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，声明了 VM 类的实例，其方式与示例 1 相同。但是，通过声明此属性的实例来引用管理器中的现有对象，而不是使用 get 方法来引用每个属性。最后，新虚拟机会添加到 Manager 中。

```
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在资源上执行操作。在本例中，名为"test"的虚拟机已启动。

对资源执行操作

1. 声明资源的实例：

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

2. 声明要发送到资源的操作参数：

```
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);
```

或者，您可以作为内部方法执行操作：

```
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。在本例中，列出了名称 '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();
```

=== getting Sub-Resources

以下示例概述了如何引用资源的子资源。在这个示例中，引用名为 'my disk' 的磁盘，其名称为 'test' 的虚拟机。

获取资源子资源

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的实例：

```
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将子资源添加到资源。在本例中，大小为 '1073741824L' 的新磁盘，接口 'virtio' 和 format 'cow' 则添加到带有名称 'test' 的虚拟机中。

在资源中添加子资源

1. 声明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);
```

2.9. 修改子资源

以下示例概述了如何修改子资源。在本例中，名为 'test_Disk1' 的磁盘名称更改为 'test_Disk1'，名称为 'test' 变为 'test_Disk1_updated'。

更新子资源

1. 声明要修改的子资源的资源实例：

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

2. 声明要修改的子资源的实例：

```
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. 在 SUB-RESOURCES 上执行操作

以下示例概述了如何在子资源上执行操作。在本例中，名为 'test_Disk1' 的磁盘将被激活为名称为 'test' 的虚拟机。

在子资源上执行操作

1. **声明包含要执行操作的子资源的资源实例：**

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

2. **声明子资源的实例：**

```
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 METHODS

下表概述了 Java 软件开发套件的 V4 中可用于 ConnectionBuilder 类的关键方法。

表 A.1. ConnectionBuilder Methods

方法	参数类型	描述
user	**字符串**	要连接到 Manager 的用户的名称。您必须同时指定用户名和域，如 **admin@internal**。此方法必须与 **password** 方法一起使用。
password	**字符串**	要连接到 Manager 的用户的密码。
压缩	**布尔值**	指定是否应该压缩管理器的服务器的响应。此选项默认为禁用，因此只需要此方法才能启用这个选项。
timeout	**整数**	等待响应请求的超时时间（以秒为单位）。如果请求需要超过这个值要响应的时间，则请求将被取消，并抛出异常。此参数是可选的。
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** 进行密码验证。
ssoRevokeUrl	**字符串**	SSO 撤销服务的基本 URL。只有在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 SSO 令牌撤销使用作为引擎的 SSO 服务来执行。
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 Manager 的服务器所提供的 SSL 证书中验证主机名。默认情况下，验证主机名的身份，如果主机名不正确，连接将被拒绝，因此仅需要此方法来禁用此选项。

方法	参数类型	描述
trustStoreFile	**字符串**	指定包含 CA 证书的文件位置，用于验证托管管理器的服务器所呈现的证书。此方法必须与 **trustStorePassword** 方法一起使用。
trustStorePassword	**字符串**	用于访问 **trustStorePath** 方法中指定的密钥存储文件的密码。