



Red Hat Virtualization 4.4

Java SDK Guide

Using the Red Hat Virtualization Java SDK

Red Hat Virtualization 4.4 Java SDK Guide

Using the Red Hat Virtualization Java SDK

Red Hat Virtualization Documentation Team

Red Hat Customer Content Services

rhev-docs@redhat.com

Legal Notice

Copyright © 2022 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, 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.

Abstract

This guide describes how to install and work with version 4 of the Red Hat Virtualization Java software development kit.

Table of Contents

CHAPTER 1. OVERVIEW	3
1.1. PREREQUISITES	3
1.2. INSTALLING THE JAVA SOFTWARE DEVELOPMENT KIT	3
1.3. DEPENDENCIES	4
1.4. CONFIGURING SSL	4
1.4.1. Configuring SSL	4
1.4.2. Host Verification	5
CHAPTER 2. USING THE SOFTWARE DEVELOPMENT KIT	6
2.1. CONNECTING TO THE RED HAT VIRTUALIZATION MANAGER IN VERSION 4	6
2.2. LISTING ENTITIES	7
2.3. MODIFYING THE ATTRIBUTES OF RESOURCES	7
2.4. GETTING A RESOURCE	7
2.5. ADDING RESOURCES	7
2.6. PERFORMING ACTIONS ON RESOURCES	8
2.7. LISTING SUB-RESOURCES	9
2.8. ADDING SUB-RESOURCES TO A RESOURCE	9
2.9. MODIFYING SUB-RESOURCES	10
2.10. PERFORMING ACTIONS ON SUB-RESOURCES	10
APPENDIX A. CONNECTIONBUILDER METHODS	12

CHAPTER 1. OVERVIEW

Version 4 of the Java software development kit is a collection of classes that allows you to interact with the Red Hat Virtualization Manager in Java-based projects. By downloading these classes and adding them to your project, you can access a range of functionality for high-level automation of administrative tasks.



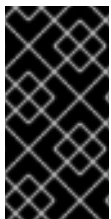
NOTE

Version 3 of the SDK is no longer supported. For more information, consult [the RHV 4.3 version of this guide](#).

1.1. PREREQUISITES

To install the Java software development kit, you must have:

- A system where Red Hat Enterprise Linux 8 is installed. Both the Server and Workstation variants are supported.
- A subscription to Red Hat Virtualization entitlements.



IMPORTANT

The software development kit is an interface for the Red Hat Virtualization REST API. Use the version of the software development kit that corresponds to the version of your Red Hat Virtualization environment. For example, if you are using Red Hat Virtualization 4.3, use V4 Java software development kit.

1.2. INSTALLING THE JAVA SOFTWARE DEVELOPMENT KIT

Install the Java software development kit and accompanying documentation.

Installing the Java Software Development Kit

1. Enable the repositories:

```
# 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. Enable the **pki-deps** module.

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

3. Install the required packages for Java SDK V4:

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

The V4 Java software development kit and accompanying documentation are downloaded to the `/usr/share/java/java-ovirt-engine-sdk4` directory and can be added to Java projects.

1.3. DEPENDENCIES

To use the Java software development kit in Java applications, you must add the following JAR files to the class path of those applications:

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

The packages that provide these JAR files are installed as dependencies to the **ovirt-engine-sdk-java** package. By default, they are available in the `/usr/share/java` directory on Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 systems.

1.4. CONFIGURING SSL

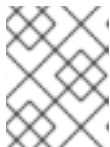
The Red Hat Virtualization Manager Java SDK provides full support for HTTP over Secure Socket Layer (SSL) and the IETF Transport Layer Security (TLS) protocol using the Java Secure Socket Extension (JSSE). JSSE has been integrated into the Java 2 platform as of version 1.4 and works with the Java SDK out of the box. On earlier Java 2 versions, JSSE must be manually installed and configured.

1.4.1. Configuring SSL

The following procedure outlines how to configure SSL using the Java SDK.

Configuring SSL

1. Download the certificate used by the Red Hat Virtualization Manager.



NOTE

By default, the location of the certificate used by the Red Hat Virtualization Manager is in **`/etc/pki/ovirt-engine/ca.pem`**.

2. Create a truststore:

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

3. Specify the **`trustStoreFile`** and **`trustStorePassword`** arguments when constructing an instance of the **`Api`** or **`Connection`** object:

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



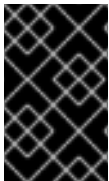
NOTE

If you do not specify the **trustStoreFile** option when creating a connection, the Java SDK attempts to use the default truststore specified by the system variable **javax.net.ssl.trustStore**. If this system variable does not specify a truststore, the Java SDK attempts to use a truststore specified in **\$JAVA_HOME/lib/security/jssecacerts** or **\$JAVA_HOME/lib/security/cacerts**.

1.4.2. Host Verification

By default, the identity of the host name in the certificate is verified when attempting to open a connection to the Red Hat Virtualization Manager. You can disable verification by passing the following argument when constructing an instance of the **Connection** class:

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



IMPORTANT

This method should not be used for production systems due to security reasons, unless it is a conscious decision and you are aware of the security implications of not verifying host identity.

CHAPTER 2. USING THE SOFTWARE DEVELOPMENT KIT

This chapter outlines several examples of how to use the Java Software Development Kit. All examples in this chapter use version 3 of the software development kit unless otherwise noted.

2.1. CONNECTING TO THE RED HAT VIRTUALIZATION MANAGER IN VERSION 4

In V4 of the Java software development kit, the **Connection** class is the main class you use to connect to and manipulate objects in a Red Hat Virtualization environment. To declare an instance of this class, you must declare an instance of the **ConnectionBuilder** class, pass the required arguments to this instance using builder methods, then call the **build** method on the instance. The **build** method returns an instance of the **Connection** class that you can then assign to a variable and use to perform subsequent actions.

The following is an example of a simple Java SE program that creates a connection with a Red Hat Virtualization environment using version 4 of the software development kit:

Example 2.1. Connecting to the Red Hat Virtualization Manager

```
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://rhvm.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

        }
    }
}
```

This example creates a connection using basic authentication, but other methods are also available. For a list of the key arguments that can be passed to instances of the **ConnectionBuilder** class, see [Appendix A, ConnectionBuilder Methods](#).

2.2. LISTING ENTITIES

The following example outlines how to list entities in the Red Hat Virtualization Manager. In this example, the entities to be listed are virtual machines, which are listed using the **getVMs()** method of the **Api** class.

Listing Entities

1. Declare a **List** of the type of entity to be listed and use the corresponding method to get the list of entities:

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

2.3. MODIFYING THE ATTRIBUTES OF RESOURCES

The following example outlines how to modify the attributes of a resource. In this example, the attribute to be modified is the description of the virtual machine with the name 'test', which is changed to 'java_sdk'.

Modifying the Attributes of a Resource

1. Declare an instance of the resource whose attributes are to be modified:

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

2. Set the new value of the attribute:

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

3. Update the virtual machine to apply the change:

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

2.4. GETTING A RESOURCE

In the Java Software Development Kit, resources can be referred to via two attributes: **name**, and **UUID**. Both return an object with the specified attribute if that object exists.

To get a resource using the value of the **name** attribute:

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

To get a resource using the value of the **UUID** attribute:

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

2.5. ADDING RESOURCES

The following examples outline two ways to add resources to the Red Hat Virtualization Manager. In these examples, the resource to be added is a virtual machine.

Example 1

In this example, an instance of the **VM** class is declared to represent the new virtual machine to be added. Next, the attributes of that virtual machine set to the preferred values. Finally, the new virtual machine is added to the 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);
```

Example 2

In this example, an instance of the **VM** class is declared in the same way as Example 1. However, rather than using the **get** method to reference existing objects in the Manager, each attribute is referenced by declaring an instance of that attribute. Finally, the new virtual machine is added to the 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. PERFORMING ACTIONS ON RESOURCES

The following example outlines how to perform actions on a resource. In this example, a virtual machine with the name 'test' is started.

Performing an Action on a Resource

1. Declare an instance of the resource:

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

2. Declare action parameters to send to the resource:

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

3. Perform the action:

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

Alternatively, you can perform the action as an inner method:

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

2.7. LISTING SUB-RESOURCES

The following example outlines how to list the sub-resources of a resource. In this example, the sub-resources of a virtual machine with the name 'test' are listed.

Listing Sub-Resources

1. Declare an instance of the resource whose sub-resources are to be listed:

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

2. List the sub-resources:

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

=== Getting Sub-Resources

The following example outlines how to reference the sub-resources of a resource. In this example, a disk with the name 'my disk' that belongs to a virtual machine with the name 'test' is referenced.

Getting the Sub-Resources of a Resource

1. Declare an instance of the resource whose sub-resources are to be referenced:

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

2. Declare an instance of the sub-resource to be referenced:

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

2.8. ADDING SUB-RESOURCES TO A RESOURCE

The following example outlines how to add sub-resources to a resource. In this example, a new disk with a size of '1073741824L', interface 'virtio' and format 'cow' are added to a virtual machine with the name 'test'.

Adding a Sub-Resource to a Resource

1. Declare an instance of the resource to which sub-resources are to be added:

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

2. Create parameters to define the attributes of the resource:

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

3. Add the sub-resource:

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

2.9. MODIFYING SUB-RESOURCES

The following example outlines how to modify sub-resources. In this example, the name of a disk with the name 'test_Disk1' belonging to a virtual machine with the name 'test' is changed to 'test_Disk1_updated'.

Updating a Sub-Resource

1. Declare an instance of the resource whose sub-resource is to be modified:

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

2. Declare an instance of the sub-resource to be modified:

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

3. Set the new value of the attribute:

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

4. Update the sub-resource:

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

2.10. PERFORMING ACTIONS ON SUB-RESOURCES

The following example outlines how to perform actions on sub-resources. In this example, a disk with the name 'test_Disk1' belonging to a virtual machine with the name 'test' is activated.

Performing an Action on a Sub-Resource

1. Declare an instance of the resource containing the sub-resource on which the action is to be performed:

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

2. Declare an instance of the sub-resource:

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

3. Declare action parameters to send to the sub-resource:

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

4. Perform the action:

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

APPENDIX A. CONNECTIONBUILDER METHODS

The following table outlines the key methods available to the **ConnectionBuilder** class used in V4 of the Java software development kit.

Table A.1. ConnectionBuilder Methods

Method	Argument Type	Description
user	String	The name of the user with which to connect to the Manager. You must specify both the user name and domain, such as admin@internal . This method must be used together with the password method.
password	String	The password of the user with which to connect to the Manager.
compress	Boolean	Specifies whether responses from the server where the Manager is hosted should be compressed. This option is disabled by default, so this method is only required to enable this option.
timeout	Integer	The timeout, in seconds, to wait for responses to requests. If a request takes longer than this value to respond, the request is cancelled, and an exception is thrown. This argument is optional.
ssoUrl	String	The base URL of the server where the Manager is hosted. For example, https://server.example.com/ovirt-engine/sso/oauth/token?grant_type=password&scope=ovirt-app-api for password authentication.
ssoRevokeUrl	String	The base URL of the SSO revoke service. This option only needs to be specified when you use an external authentication service. By default, this URL is automatically calculated from the value of the url option so that SSO token revoke is performed using the SSO service that is part of the engine.

Method	Argument Type	Description
ssoTokenName	String	The token name in the JSON SSO response returned from the SSO server. By default, this value is access_token .
insecure	Boolean	Enables or disables verification of the host name in the SSL certificate presented by the server where the Manager is hosted. By default, the identity of host names is verified, and the connection is rejected if the host name is not correct, so this method is only required to disable this option.
trustStoreFile	String	Specifies the location of a file containing the CA certificate used to verify the certificate presented by the server where the Manager is hosted. This method must be used together with the trustStorePassword method.
trustStorePassword	String	The password used to access the keystore file specified in the trustStorePath method.