Oracle® Fusion Middleware

User's Guide for Oracle JRockit Virtual Edition

11*g* Release 1 (11.1.1.3.0)

E15206-02

April 2010

This document provides an overview of Oracle JRockit Virtual Edition, and describes how to use it to create and configure virtual machine images.

ORACLE

Oracle Fusion Middleware User's Guide for Oracle JRockit Virtual Edition, 11*g* Release 1 (11.1.1.3.0)

E15206-02

Primary Authors: Kumar Dhanagopal, Savija T V

Contents

5 Accessing the File System of a Running Virtual Machine by Using the SSH Protocol

6 Security in Oracle JRockit Virtual Edition

7 Diagnosing and Troubleshooting Problems

A Image Tool Command-Line Option Reference

B Configuration File Element Reference

C Sample Configuration Files

D Known Issues

List of Figures

List of Tables

Preface

Welcome to *Oracle Fusion Middleware User's Guide for Oracle JRockit Virtual Edition*. This document provides an overview of Oracle JRockit Virtual Edition, and describes how to use it to create and change virtual machine images.

Audience

This document is intended for system administrators who are responsible for building, running, and administering virtual machines.

It is assumed that readers understand the basic concepts of virtualization. Readers should also be familiar with running, monitoring, tuning, and troubleshooting the Oracle JRockit JVM.

Operational knowledge of Linux and other UNIX-like operating systems, though not necessary, would be an added advantage.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents:

- *Oracle JRockit JVM Diagnostics Guide*
- *Oracle VM Manager User's Guide*
- *Oracle VM Server User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Oracle JRockit Virtual Edition

This chapter provides an introduction to Oracle JRockit Virtual Edition. It contains the following topics.

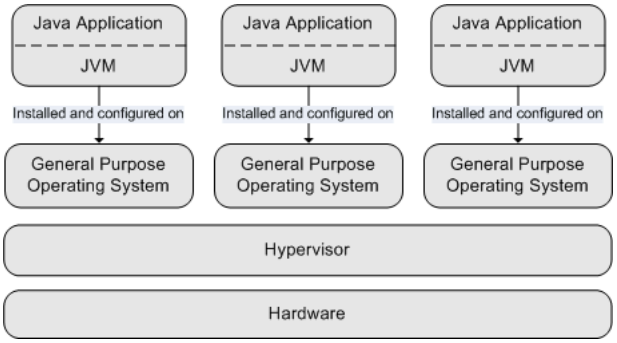
1.1 About Virtualization

Virtualization is the process of abstracting hardware resources—CPU, memory, hard disk, and network interfaces—so that multiple operating systems and applications can share the same hardware. The hardware runs a virtualization software (for example, a hypervisor) that enables you to run multiple operating systems, each capable of running simultaneously and independently, in its own isolated and secure environment.

In a virtualized environment, each isolated partition runs an operating system and application, and **looks** like a real computer with its own CPU, network interfaces, storage, and operating system.

Figure 1–1 shows several Java applications running in a hypervisor-based virtualized environment.

Figure 1–1 Java Applications Running in a Hypervisor-Based Virtualized Environment



1.2 About Oracle JRockit Virtual Edition

Oracle JRockit Virtual Edition 11.1.1.3.0 is a virtualization-enabled version of the Oracle JRockit JVM R27.6.6. Oracle JRockit Virtual Edition can run on Oracle VM 2.2, allowing Java applications to run directly on virtualized hardware.

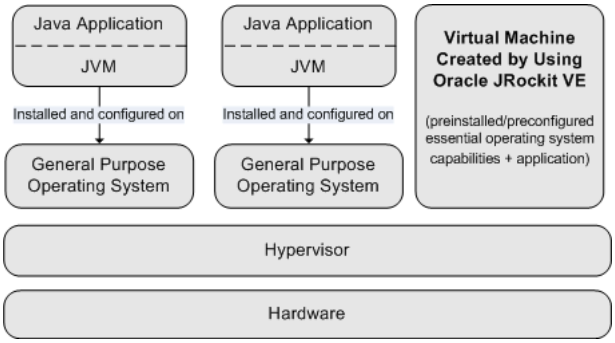
Most operating systems are general-purpose products that are very complex and large. JVMs do not require or use all of the functions that a general purpose operating system offers. So, in a typical Java environment, a significant portion of the investment in the operating system, hard-disk space, and memory remains unutilized.

Oracle JRockit Virtual Edition addresses this problem by packaging the Oracle JRockit JVM with the bare minimum operating system capabilities that are essential for a Java environment (file, network, and memory management). Oracle JRockit Virtual Edition can therefore run directly on the Oracle VM hypervisor, without the need for a general-purpose operating system.

Running your Java applications on Oracle JRockit Virtual Edition makes them easy to manage, fast, and very secure.

Figure 1–2 shows a virtual machine created by using Oracle JRockit Virtual Edition running on a hypervisor.

Figure 1–2 Virtual Machine Created by Using Oracle JRockit Virtual Edition, Running on a Hypervisor



1.2.1 About Virtual Machine Images

You can use Oracle JRockit Virtual Edition to create a virtual machine image, which is a virtualized version of a local Java application.

A virtual machine image created by using Oracle JRockit Virtual Edition consists of a binary file and an Oracle VM-specific configuration file. The virtual machine image has the same format as that of a virtual machine template, which you can use to create and run virtual machine on Oracle VM.

> **Note:** For more information about virtual machine templates, see "Managing Virtual Machine Templates" in the *Oracle VM Manager User's Guide* at `http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/resources.htm#BABEJJGA`.

For more information about creating virtual machine images by using Oracle JRockit Virtual Edition, see Chapter 2, "Creating Virtual Machine Images."

1.2.2 Benefits of Oracle JRockit Virtual Edition

The key benefits of Oracle JRockit Virtual Edition include:

- Improved Java performance

- Simplified configuration of the JVM and operating system
- No installation or configuration required (reconfiguration is possible)
- Simplified patching and version control
- Support from Oracle for the complete virtual machine stack
- Increased security
- Reduced disk and memory footprint
- Reduced operating system license costs
- Improved high-availability

1.2.3 Limitations of Oracle JRockit Virtual Edition

The following are feature limitations in Oracle JRockit Virtual Edition 11.1.1.3.0. Some of these limitations are the result of product design choices and constraints.

- Oracle JRockit Virtual Edition does not support third-party native code in the JVM. For example, loading JNI libraries is not supported.
- Virtual machines created by using Oracle JRockit Virtual Edition run only in headless mode; that is, they do not provide a graphical user interface at run time.
- You can configure a maximum of eight virtual CPUs for a virtual machine created by using Oracle JRockit Virtual Edition.
- The virtual machine created by using Oracle JRockit Virtual Edition runs as a single process. You cannot run multiple processes in the virtual machine.

2

Creating Virtual Machine Images

This chapter describes how to convert a Java application into a virtual machine image. You can use the image as a virtual machine template to create and run virtual machines on Oracle VM.

Oracle JRockit Virtual Edition provides an Image Tool, which is a `jar` file (`jrockitve-imagetool.jar`) available in the directory in which you installed Oracle JRockit Virtual Edition.

This chapter contains the following topics:

2.1 Overview of the Process to Create a Virtual Machine Image

The Image Tool enables you to create a virtual machine image from a Java application in two steps:

1. Create a configuration file for the virtual machine (see Section 2.2).

   In this step, you create an XML file in which you specify the name of the virtual machine, the amount of memory that the hypervisor must make available for the virtual machine, the number of CPUs that the virtual machine requires, and so on. In the configuration file, you also specify the `java` command for running your application; the command can include arguments for both the JRockit JVM and for the Java application.

   By using the Image Tool, you can generate a sample configuration file (`.xml`). You can use the sample configuration file as a template and modify it to specify the required configuration information, by editing the XML file in any text editor.

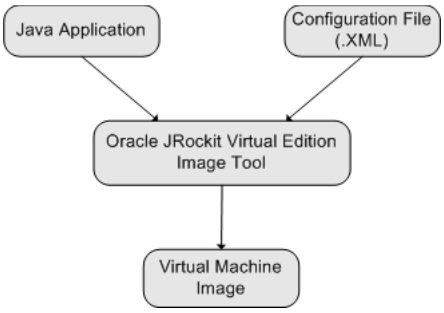
   For information about the elements of the configuration file and their attributes, see Appendix B, "Configuration File Element Reference."

2. Assemble a virtual machine image (see Section 2.3).

   In this step, you create a virtual machine image that you can run on Oracle VM, based on the configuration that you defined in the configuration file.

Figure 2–1 illustrates the process of creating a virtual machine image from a Java application.

Figure 2–1 Creating a Virtual Machine Image from a Java Application



2.2 Creating the Configuration File for the Virtual Machine

To create the configuration file for your virtual machine, perform the following steps by using the Image Tool (`jrockitve-imagetool.jar`).

1. Create a sample brief configuration file by running the following command.

   ```
   java -jar jrockitve-imagetool.jar -c [config.xml] [vm_name]
   ```

   If you run this command from a directory other than the one that contains the Oracle JRockit Virtual Edition installation, you must specify the full path (absolute or relative) to the `jrockitve-imagetool.jar` file.

   This command creates a brief sample configuration file named *`config.xml`*, with *`vm_name`* as the name of the virtual machine and placeholder values for some of the elements in the configuration file, as shown in Appendix C.1, "Sample Brief Configuration File."

   If you do not specify the optional *`config.xml`* argument, the sample configuration file is written to `stdout`. You can then create the configuration file manually in a text editor, by copying the content written to `stdout`.

   Note: You can create a detailed sample configuration file containing all the possible configuration elements, by using the **`-create-full-config`** option, and then remove the elements that are not necessary for your environment.

   You can also create the configuration XML file in the following alternative ways:

   - Extract the configuration settings pertaining to an existing virtual machine by using the **`-r`** *`vm_cfg`* `get config` option, and use the extracted configuration settings to create the new configuration file.
   - Disassemble an existing virtual machine image by using the **`-d`** option, and use the resulting XML file as the starting point for the new configuration file.

   For more information about the `-c`, `--create-full-config`, `-r`, and `-d` options, see Appendix A, "Image Tool Command-Line Option Reference."

   ---

2. Open the sample brief configuration file in a text editor, and do the following:

- Verify whether the default values for elements in the sample configuration file are appropriate for your environment, and change them if required.
- The sample brief configuration file does not include the following elements, but the Image Tool assumes certain default values for them when it assembles the virtual machine image. If the default values (described below) are not appropriate for your environment, add the elements explicitly in the configuration file and specify the required value.
  - `<jrockitve-binary-url>`: **Location of the Oracle JRockit Virtual Edition binary file** (`jrockitve.bin`)

    The default value for `<jrockitve-binary-url>` is assumed to be the directory that contains the configuration file. To specify a different location for the `jrockitve.bin` file, add the `<jrockitve-binary-url>` element.
  - `<java-filesystem-imports>`: **Location of the Java application to be virtualized**

    The default location of the Java application to be virtualized is assumed to be a subdirectory named `root` within the directory that contains the configuration XML file. For example, if the configuration XML file is in a directory named `myconfig`, the Java application to be virtualized is assumed to be in the `myconfig/root` directory. To change the location, add the `<java-filesystem-imports>` element.

    While creating a virtual machine image, the Image Tool copies, recursively, everything under the directories specified in the `<java-filesystem-imports>` element (if it is specified) or under the default `root` directory.
- Make sure that all the mandatory elements are specified in the configuration file.

For more information about the configuration elements and their attributes, see Appendix B, "Configuration File Element Reference."

3. After making the required changes in the configuration file, save it.

2.3 Assembling a Virtual Machine Image

To assemble a virtual machine image, run the following command:

```
java -jar jrockitve-imagetool.jar -a config.xml output_dir [hypervisor]
```

> **Note:** The default (and only supported) hypervisor is Oracle VM (`ovm`).

This command assembles a virtual machine image, which consists of two files – `system.img` and `vm.cfg`, in the *`output_dir`* directory.

The process of assembling a virtual machine image usually takes a few seconds. The actual time taken varies according to the size of the Java application that you are virtualizing.

You can now run the virtual machine on Oracle VM. For more information, see Chapter 3, "Creating Virtual Machines on Oracle VM."

For more information about the `-a` option, see Appendix A, "Image Tool Command-Line Option Reference."

2.4 About the File System of a Virtual Machine

The file system within a virtual machine that was created by using Oracle JRockit Virtual Edition is similar to that in most UNIX-like operating systems. It consists of a single root (`/`) directory.

The virtual disk (of the size defined in the configuration file) is mounted in the root directory. It contains the following files and directories.

- `/jrockitve`: This directory contains the Oracle JRockit JVM files, configuration files for the virtual machine, and files that provide the required kernel capabilities for Oracle JRockit Virtual Edition.
- The Java application files.

  The file structure of the Java application within the virtual machine file system matches the structures of the directories specified in the `<jrockitve-filesystem-imports>` element of the configuration file.
- `VERSION`: This file contains the version numbers of the components of Oracle JRockit Virtual Edition.

You can access the file system within a running virtual machine by using SSH protocol-based clients (SCP and SFTP). For more information, see Chapter 5, "Accessing the File System of a Running Virtual Machine by Using the SSH Protocol."

When a virtual machine is not running, you can access the file system by using the -f (--file) option of the Image Tool.

External clients and processes cannot access the local file system **within** a virtual machine when the virtual machine is running. They can, however, access shared file storage locations (such as an NFS location) that are defined by using the `-r vm_cfg add mount` operation of the Image Tool. For more information, see -r (--reconfigure) in Appendix A, "Image Tool Command-Line Option Reference."

3

Creating Virtual Machines on Oracle VM

This chapter provides an introduction to Oracle VM, and describes how to create virtual machines on Oracle VM by using a virtual machine image assembled by using Oracle JRockit Virtual Edition.

This chapter contains the following topics:

3.1 About Oracle VM

Oracle VM is a platform that provides a fully equipped environment to leverage the benefits of virtualization technology. Oracle VM enables you to create and run virtual machines.

The key components of Oracle VM are:

- **Oracle VM Server**, a self-contained virtualization environment designed to provide a lightweight, secure, server-based platform to run virtual machines

  For more information, see the *Oracle VM Server User's Guide* at `http://download.oracle.com/docs/cd/E15458_01/doc.22/e15444/toc.htm`.

- **Oracle VM Manager**, which enables you to manage Oracle VM servers, virtual machines, and resources

  For more information, see the *Oracle VM Manager User's Guide* at `http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/toc.htm`.

3.2 Creating Virtual Machines on Oracle VM

The process of creating virtual machines on Oracle VM consists of the following steps:

1. Copy the Virtual Machine Image to the Oracle VM Server
2. Create a Virtual Machine in Oracle VM

Note: Virtual machines created by using Oracle JRockit Virtual Edition do not support the plug-in console feature in Oracle VM Manager.

3.2.1 Copy the Virtual Machine Image to the Oracle VM Server

When you create a virtual machine image by using the Oracle JRockit Virtual Edition Image Tool, two files—`system.img` and `vm.cfg`—are created.

The `vm.cfg` file contains configuration parameters for the virtual machine, as shown in the following example:

```
name="wls-ve"
bootloader="/usr/bin/pygrub"
memory=512
disk=['tap:aio:/OVS/seed_pool/wls-ve/system.img,sda1,w']
vif=['']
on_crash="coredump-destroy"
```

The `disk` parameter indicates the directory on the Oracle VM server in which the `system.img` file should be located.

> **Note:** By default, the path of the `system.img` file is `/OVS/seed_pool/`*`vm-name`*`/system.img`, where *`vm-name`* is the name of the virtual machine.

Copy `vm.cfg` and `system.img` to the *`vm-name`* directory (on the Oracle VM server) indicated by the `disk` parameter in the `vm.cfg` file.

3.2.2 Create a Virtual Machine in Oracle VM

This section provides an overview of the process for creating virtual machines on Oracle VM. For details about each step in the process, see the Oracle VM documentation to which pointers are provided at relevant places throughout the section.

You can create a virtual machine in Oracle VM by using a virtual machine image or a virtual machine template. Oracle JRockit Virtual Edition creates virtual machine images in the same format as Oracle VM Templates.

To create a virtual machine in Oracle VM, perform the following steps:

Step 1: Import the Virtual Machine Image as an Oracle VM Template

Step 2: Approve the Imported Template

Step 3: Create the Virtual Machine

> **Note:** Before creating a virtual machine, you must have a server pool that contains a Virtual Machine Server. For more information about the server pool and the Virtual Machine Server, see *Oracle VM Manager User's Guide*.

Step 1: Import the Virtual Machine Image as an Oracle VM Template

Make sure that the virtual machine image files—`system.img` and `vm.cfg`—have been copied to the `/OVS/seed_pool/`*`vm-name`* directory on the Oracle VM server, as described in Section 3.2.1, and make sure that the configuration file is named vm.cfg.

Import the virtual machine image as a template, by following the procedure described in the "Selecting from the Server Pool" section of the *Oracle VM Manager User's Guide* at `http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/resources.htm#CHDFDGAC`.

Note: Alternatively, you can import the virtual machine image as an Oracle VM template by downloading the template from an HTTP or FTP server as described in the "Downloading from External Source" section of the *Oracle VM Manager User's Guide* at http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/resources.htm#CHDHFDAI.

Step 2: Approve the Imported Template

After importing the template, the status of the template is **`Pending`**. Only templates with the status **`Approved`** are available for creating virtual machines.

Approve the template by following the procedure described in the "Approving the Imported Virtual Machine Template" section of the *Oracle VM Manager User's Guide* at http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/resources.htm#BABIHBHB.

Note: Only users with the `Manager` or `Administrator` role can approve templates in Oracle VM Manager.

Step 3: Create the Virtual Machine

After approving a virtual machine template, you can use it to create a virtual machine, which inherits all the content and configuration from the template.

Create the virtual machine by following the procedure described in the "Creating a New Virtual Machine Based on Virtual Machine Template" section of the *Oracle VM Manager User's Guide* at http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/vm.htm#CACGFBFB.

You can now start and stop the virtual machine by using Oracle VM, as described in the "Starting and Shutting Down a Virtual Machine" section of the *Oracle VM Manager User's Guide* at http://download.oracle.com/docs/cd/E15458_01/doc.22/e15441/vm.htm#BABFCFCG.

4

Modifying Virtual Machine Images

You might need to modify a virtual machine image to change the memory available for the virtual machine, change the location of the log file, manage disks, mount file systems, and so on.

This chapter contains the following topics:

4.1 Modifying Virtual Machine Images by Using the Image Tool

> **Caution:** Before using the Image Tool to view or modify the configuration or file system of a virtual machine, shut down the virtual machine.
>
> Do not attempt to use the Image Tool to view or modify the configuration or file system of a running virtual machine.

To modify a virtual machine image by using the Image Tool, do one of the following:

- Modify the virtual machine image configuration directly by using the `--reconfigure` option of the Image Tool.

  For more information, see Section 4.1.1, "Reconfiguring Virtual Machine Images.")

- Modify the services (such as `sshd`) that are enabled in the virtual machine configuration by using the `--reconfigure-service` option of the Image Tool.

  For more information, see --reconfigure-service in Appendix A, "Image Tool Command-Line Option Reference.".

- Modify the file system within a virtual machine image by using the `--file` option of the Image Tool.

  For more information, see Section 4.1.2, "Modifying the File System Within a Virtual Machine Image by Using the Image Tool.")

- Extract the source components of the virtual machine image by using the `--disassemble` option of the Image Tool, make the required changes, and assemble (create) the virtual machine image again.

  For more information, see Section 4.1.3, "Disassembling and Re-Creating Virtual Machine Images."

4.1.1 Reconfiguring Virtual Machine Images

You can use the **-r (--reconfigure)** option of the Image Tool to modify the configuration settings of a virtual machine image directly, bypassing the configuration XML file.

> **Note:** At any point, you can extract an XML file reflecting the current configuration state of a virtual machine by using the `-d` option.

For detailed information about the `-r` option, its operations, and the configuration settings that you can modify by using the option, see **-r (--reconfigure)** in Appendix A, "Image Tool Command-Line Option Reference."

Examples

> **Caution:** Before using the **`-r`** option or any other Image Tool option that accesses the virtual machine image, you **must** shut down the virtual machine.

- To change the name of the virtual machine to `newname`, run the following command:

```
java -jar jrockitve-imagetool.jar -r vm.cfg set vm-name newname
```

- To add an NFS mount point named `mynfs` pointing to the `/export/mynfs` path on the `nfs.myhost.com` server with the user ID `501` and group ID `502`, run the following command:

```
java -jar jrockitve-imagetool.jar -r vm.cfg add mount nfs /mynfs nfs.myhost.com
/export/mynfs uid=501 gid=502
```

- To add a DNS server (with the IP address `10.172.22.5`, for example), run the following command:

```
java -jar jrockitve-imagetool.jar -r vm.cfg add network-dns-servers 10.172.22.5
```

- Consider a virtual machine image that currently has three DNS servers defined. If you run the `-r` option with the `get network-dns-servers` operation, the results are displayed in the lookup order, as shown in the following example:

```
#1 172.22.23.24
#2 172.22.23.25
#3 172.22.23.26
```

  If you want to remove the second entry in the DNS server table, run the following command:

```
java -jar jrockitve-imagetool.jar -r vm.cfg remove network-dns-servers 2
```

  The `get` operation now yields the following results:

```
#1 172.22.23.24
#2 172.22.23.26
```

4.1.2 Modifying the File System Within a Virtual Machine Image by Using the Image Tool

You can use the **-f (--file)** option of the Image Tool to copy files to and from the file system of a virtual machine image, view directory listings, move and delete files, create and delete directories, find files, and display the contents of text files.

Do not attempt to modify the file system of a virtual machine image unless you understand the file system. For more information, see Section 2.4, "About the File System of a Virtual Machine."

To modify the file system of a virtual machine image by using the Image Tool, **shut down the virtual machine**, and then run the following command:

```
java -jar jrockitve-imagetool.jar -f vm_cfg operation [parameters]
```

In this command, `vm_cfg` is the path and name of the configuration file (`vm.cfg`) corresponding to the virtual machine to be modified, `operation` is name of the modification task, and `parameters` represents the arguments that are required for the specified modification task.

Examples

> **Note:** For detailed information about the `-f` option, the operations it supports, the arguments it requires, and more examples, see **-f (--file)** in Appendix A, "Image Tool Command-Line Option Reference."

- To copy the contents of the `/jrockitve/log/jrockitve.log` file in a virtual machine image to the `/tmp/jrve.log` file in the local file system, run the following command:

  ```
  java -jar jrockitve-imagetool.jar -f vm.cfg get /jrockitve/log/jrockitve.log
  /tmp/jrve.log
  ```

- To copy a Java class file named `app.class` from the `/app/version2/` directory of the local system to the `/vmapp` directory of the virtual machine, run the following command:

  ```
  java -jar jrockitve-imagetool.jar -f vm.cfg put /app/version2/app.class /vmapp
  ```

4.1.3 Disassembling and Re-Creating Virtual Machine Images

You can use the **-d (--disassemble)** option of the Image Tool to disassemble a virtual machine image into its individual components.

After disassembling a virtual machine image, you can modify the source Java application or change the configuration settings defined in the configuration XML file, and then create a new virtual machine image.

> **Note:** The `-d` option is especially useful when you want to modify virtual machines that have been in production for a while during which several changes might have been made, and you now want to examine and modify the current configuration or the Java application.

To disassemble a virtual machine image, **shut down the virtual machine**, and then run the following command:

```
java -jar jrockitve-imagetool.jar -d vm_cfg output_directory
```

In this command, `vm_cfg` is the path and name of the configuration file of the virtual machine (`vm.cfg`) and `output_directory` is the directory to which you want the disassembled components of the virtual machine image to be copied.

For more information, see -d (--disassemble) in Appendix B, "Configuration File Element Reference."

4.2 Enabling Access from the Virtual Machine to External File Systems

You can configure the virtual machine created by using Oracle JRockit Virtual Edition to access a shared storage location that is external to the virtual machine. For example, you can specify an NFS mount point through which the Java application in the virtual machine (say, WebLogic Server) can access files that are shared at an external storage location.

You can configure mount points in the virtual machine by using the **`-r`** option of the Image Tool. For more information, see Appendix A, "Image Tool Command-Line Option Reference."

Note: File locking is not available in the current release. So the virtual image could, for example, read a file from a shared storage location while another process is concurrently writing to the same file, leading to problems.

4.3 Patching Oracle JRockit Virtual Edition

Oracle JRockit Virtual Edition provides a command-line option (**`-p`**) that enables you to replace the Oracle JRockit Virtual Edition binary (JVM, kernel, and services) within a virtual machine image with a new version of the file.

For information about the syntax and behavior of the -p option, see -p (--patch) in Appendix A, "Image Tool Command-Line Option Reference."

5

Accessing the File System of a Running Virtual Machine by Using the SSH Protocol

Oracle JRockit Virtual Edition enables you to transfer files to and from the file system of a **running** virtual machine by using the SSH protocols, SCP (secure copy) and SFTP (secure FTP).

An SSH service (`sshd`) is included in virtual machine images that are created by using Oracle JRockit Virtual Edition, but the service is not enabled by default. The SSH service supports password-based and public key authentication.

> **Note:** Oracle JRockit Virtual Edition does **not** provide shell access to virtual machines.

This chapter describes how to enable, configure, and use the SSH service to access the file system of a running virtual machine. It contains the following topics:

5.1 Enabling the SSH Service

To enable the SSH service, do the following:

1. Shut down the virtual machine.
2. Run the following command:

   ```
   java -jar jrockitve-imagetool.jar -r vm_cfg enable service sshd
   ```

   In this command, *`vm_cfg`* represents the path and name of the configuration file (`vm.cfg`) corresponding to the virtual machine.

To verify whether the SSH service is enabled, run the following command:

```
java -jar jrockitve-imagetool.jar -r vm_cfg get enabled-services
```

The following output is displayed:

```
sshd (An SSH2 implementation with SCP and SFTP support)
```

5.2 Configuring the SSH Service

You can configure the SSH service for password-based authentication and public key-based authentication.

5.2.1 Configuring the SSH Service for Password-Based Authentication

In this form of authentication, a user name and password must be used to log in to the SSH service in the virtual machine.

Prerequisite

Enable the SSH service as described in Section 5.1.

Procedure

To configure the SSH service for password-based authentication, do the following:

1. Shut down the virtual machine.
2. Add a user for the SSH service by running the following command:

   ```
   java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd add user
   user_name
   ```

   In this command, *vm_cfg* represents the path and name of the configuration file (`vm.cfg`) corresponding to the virtual machine, and *user_name* represents a user who can access the `sshd` service.
3. At the prompt, enter a password for the user and confirm the password.

5.2.2 Configuring the SSH Service for Public Key Authentication

Public key authentication is potentially more secure than password-based authentication, but it is more difficult to set up. You must generate a public-private key pair, add the public key to the `sshd` service in the virtual machine, and specify the private key while logging to the service.

Prerequisites

- Generate an SSH-2 RSA-compliant public-private key pair by using a key-generator tool such as PuTTY or the `ssh-keygen` command, and save the keys.
- Enable the SSH service as described in Section 5.1.

Procedure

To configure the SSH service for public key-based authentication do the following:

1. Shut down the virtual machine.
2. Add the public key to the `sshd` service by running the following command:

   ```
   java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd add key
   keyfile
   ```

   In this command, *vm_cfg* represents the path and name of the configuration file (`vm.cfg`) corresponding to the virtual machine, and *keyfile* represents the path and name of the file in which you saved the generated public key.

5.3 Logging in to the SSH Service

After enabling the SSH service, you can log in by using either a password or a private key depending on the type of authentication enabled.

> **Note:** The command syntaxes provided in this section are specific to Linux-based SFTP and SCP clients. For information about the syntaxes to be used in other such third-party clients, see the documentation for those clients.

5.3.1 Logging in to the SSH Service Using a Password

To log in to the SSH service by using password-based authentication, perform the following steps:

1. Start the virtual machine.
2. Run the following command from the machine that you are using to establish an SSH connection with the virtual machine:
   - By using SFTP: `sftp host`
   - By using SCP:
     - To copy files from the file system of the virtual machine to the local machine:

       ```
       scp host:path_in_virtual_machine_file-system path_in_local_file-system
       ```

     - To copy files from the local machine to the virtual machine file system:

       ```
       scp path_in_local_file-system host:path_in_virtual_machine_file-system
       ```

   In this command, *host* is the name or IP address of the virtual machine to which you want to connect.
3. At the prompt, enter the password.

   You can now use the standard SFTP or SCP commands to access and modify the file system of the running virtual machine.

   > **Note:** Some changes (for example, modifications to `jar` and `class` files that are already loaded by the JVM) require the virtual machine to be restarted to take effect.

   The following message is displayed on the virtual machine console:

   ```
   INFO: session created by machine using password
   ```

   In this message, `machine` is the name or IP address of the client machine from which you established the SSH connection to the virtual machine.

5.3.2 Logging in to the SSH Service Using a Private Key

To log in to the SSH service by using key-based authentication, perform the following steps:

1. Start the virtual machine.

2. Run the following command from the machine that you are using to establish an SSH connection with the virtual machine:

   - By using SFTP: `sftp -o IdentityFile=`*`privatekey_file user_name@host`*
   - By using SCP:
     - To copy files from the virtual machine file system to the local machine:

       `scp -i` *`privatekey_file user_name@host`*`:`*`path_in_virtual_machine_file-system path_in_local_file-system`*

     - To copy files from the local machine to the virtual machine file system:

       `scp -i` *`privatekey_file path_in_local_file-system user_name@host`*`:`*`path_in_virtual_machine_file-system`*

   In this command, *`user_name`* is the user name that you must use to start the SSH connection, *`host`* is the name or IP address of the virtual machine to which you want to connect, and `privatekey_file` is the path and name of the file (on the local machine) that contains the private key.

3. If you specified a passphrase while generating the keys, you must enter the passphrase at the prompt.

   You can now use the standard SFTP or SCP commands to access and modify the file system of the running virtual machine.

   Note: Some changes (for example, modifications to `jar` and `class` files that are already loaded by the JVM) require the virtual machine to be restarted to take effect.

6

Security in Oracle JRockit Virtual Edition

This chapter contains the following topics.

6.1 Security Features of Oracle JRockit Virtual Edition

Oracle JRockit Virtual Edition, by virtue of the following features, provides an intrinsically secure Java server platform.

- **Simple Configuration**: Configuring virtual machines created by using Oracle JRockit Virtual Edition is significantly simpler than configuring Java applications to run on standard operating systems. This simplicity minimizes the potential for configuration problems and human errors, which are the most common reasons for security breaches.
- **Small Code Size**: Virtual machines created by using Oracle JRockit Virtual Edition encapsulate only the essential operating system functions that Java applications require. The size of the operating system-level code in Oracle JRockit Virtual Edition is a fraction of the size of a standard operating system.
- **Java Sandbox Protection**: Virtual machines created by using Oracle JRockit Virtual Edition are protected from other applications by the proven Java sandbox model.
- **Singe-User Process**: Virtual machines created by using Oracle JRockit Virtual Edition run as single-user processes, which do not require or allow user access to the operating system in any form.

6.2 Security Guidelines

The following security guidelines are generic to any enterprise IT environment, not necessarily specific to virtualization.

- Do not store sensitive data on an NFS Server.

  Oracle JRockit Virtual Edition does not encrypt communication with NFS servers. So storing sensitive data on NFS servers can compromise the security of your system.

- Make sure that the Java application is secure.

Oracle JRockit Virtual Edition provides a secure run-time environment for the virtual machine that you create from the Java application, but the Java application has full access to its files. So make sure that the Java application is secure.

- Use a firewall to protect virtual machines (that are running on a local network) from external access.
- Secure the virtualization server so that unauthorized users cannot gain root access to the server. Oracle JRockit Virtual Edition cannot protect itself if unauthorized root access to the virtualiztion server is possible.
- Grant control and console access only to trusted users.

  Configure the virtual infrastructure such that only users that need to modify the run-time state of virtual machines are authorized to access the virtual machine console. Oracle JRockit Virtual Edition cannot protect itself from virtual machine shutdown and other virtual machine-related attacks if this policy is not maintained.
- Store the virtual machine files securely so that unauthorized users cannot access them.

6.3 Modifying SSL Certificates

Oracle JRockit Virtual Edition enables you to modify the `cacerts` certificates file, which represents a system-wide keystore with certificate authority (CA) certificates. The file is located in the `/jrockitve/jrockit/jre/lib/security` directory in the file system of the virtual machine image.

To add or remove SSL certificates, **shut down the virtual machine**, and then perform the following steps:

1. Copy the `cacerts` file from the file system of the virtual machine to a local directory by running the following Image Tool command:

   ```
   java -jar jrockitve-imagetool.jar -f vm_cfg get
   /jrockitve/jrockit/jre/lib/security/cacerts tmp/cacerts
   ```

   In this command, `vm_cfg` represents the path and name of the virtual machine image file (`vm.cfg`) and *`tmp`* represents the path of the directory in the local file system in which the `cacerts` file should be copied.
2. Modify the `cacerts` file as required by using the `keytool` utility.
3. Copy the modified `cacerts` file from the local directory to the file system of the virtual machine by running the following Image Tool command:

   ```
   java -jar jrockitve-imagetool.jar -f vm_cfg put tmp/cacerts
   /jrockitve/jrockit/jre/lib/security/cacerts
   ```

   In this command, `vm_cfg` represents the path and name of the virtual machine image file (`vm.cfg`) and *`tmp`* represents the path of the directory in the local file system in which the modified `cacerts` file is located.

For more information about the **`get`** and **`put`** operations of the -f (--file) command-line option of the Image Tool, see Appendix A, "Image Tool Command-Line Option Reference."

7

Diagnosing and Troubleshooting Problems

This chapter provides guidelines to help you diagnose and troubleshoot problems that might occur when you use Oracle JRockit Virtual Edition.

This chapter contains the following topics:

- Image Tool Problems
- Run-Time Problems
- Contacting Oracle for Help

> **Note:** **Run-time problems** are those that occur when you run virtual machines (created by using Oracle JRockit Virtual Edition) on Oracle VM.

7.1 Image Tool Problems

This section provides information to help you diagnose and solve problems that might occur when you use the Oracle JRockit Virtual Edition Image Tool.

This section contains the following topics:

- How to Diagnose and Solve Image Tool Problems
- Solutions to Specific Image-Tool Errors

7.1.1 How to Diagnose and Solve Image Tool Problems

This section describes a roadmap that you can follow to diagnose and solve Image Tool problems efficiently.

1. Make sure that you typed the command according to the syntax specified in Appendix A, "Image Tool Command-Line Option Reference."
2. Look for a solution to the problem in the error message that is displayed (if any) when you ran the command. For most errors that occur when you use the Image Tool, the error message provides a solution.

   For example, when you use the **`-c`** option to create a configuration XML file and specify an XML output file that already exists, the Image Tool displays the following error message and solution:

   ```
   Writing file "myconfig.xml" ...
   error: Failed to overwrite "\myconfig.xml".
   Use the "--force" option to override the warning.
   ```

3. If the error message text does not help you diagnose and solve the problem, run the command again after including the **`--log verbose`** option, to see detailed messages while the command executes.

   For more information about the logging options, see -l (--log).

4. Check whether Section 7.1.2, "Solutions to Specific Image-Tool Errors" contains the solution for the problem.

5. Contact Oracle for help, as described in Section 7.3, "Contacting Oracle for Help."

7.1.2 Solutions to Specific Image-Tool Errors

ERROR: Unable to access jar file jrockitve-imagetool.jar

This error occurs when you run any command of the Image Tool from a directory other than the one in which you installed Oracle JRockit Virtual Edition, without specifying the full path of `jrockitve-imagetool.jar`.

Solution: Do one of the following:

- Go to the directory in which you installed the Oracle JRockit Virtual Edition, and run the command from that directory.
- In the command line, specify the path (absolute or relative) of the `jrockitve-imagetool.jar` file.

ERROR: Cannot find default <jrockitve-binary-url> at "protocol:location"

Note: In this error message, `protocol:location` is the location (specified or default) at which the Image Tool expects to find the `jrockitve.bin` file.

The `protocol` could be `file:`, `http://`, `ftp://`, and so on. The `location` would vary accordingly.

This error occurs when you use the **-a (--assemble)** option, in either of the following conditions:

- The `jrockitve.bin` file is not available at the location specified in the `<jrockitve-binary-url>` element of the configuration XML file.
- The `<jrockitve-binary-url>` element is not specified in the configuration XML file, and the default location (the directory in which the configuration file resides) does not contain the `jrockitve.bin` file.

Solution: Open the configuration XML file in a text editor, and correct the path specified in the `<jrockitve-binary-url>` element. If the element does not exist, add it.

ERROR: Cannot find default <java-application-dir> at "directory_name"

This error occurs when you use the **-a (--assemble)** option, in either of the following conditions:

- The `<java-filesystem-imports>` element, which specifies the location of the Java application to be virtualized, is not specified in the configuration XML file, and the default location (a subdirectory named `root` within the directory in which the configuration XML file resides) does not exist.

- The files or directories specified in the `<java-filesystem-imports>` element do not exist.

Solution: Open the configuration XML file in a text editor and correct the values specified in the `<java-filesystem-imports>` element or add the element, as appropriate.

ERROR: Device or resource is busy. (File system is already mounted.)

This error occurs in the following situations:

- The Image Tool terminates unexpectedly.

  Solution: Shut down the virtual machine (if it is running), and then use the --repair option to check for and fix problems in the file system of the virtual machine image.

- You attempted to modify the image of a running virtual machine by using Image Tool options such as `-r` or `-f`.

  Solution: Shut down the virtual machine, and then perform the required modifications.

7.2 Run-Time Problems

This section provides information to help you diagnose and solve problems that might occur when you run virtual machines (created by using Oracle JRockit Virtual Edition) on Oracle VM.

This section contains the following topics:

7.2.1 How to Diagnose and Solve Run-Time Problems

This section describes a roadmap that you can follow to diagnose and solve Image Tool problems efficiently.

1. Verify whether you created the virtual machine as described in Chapter 3, "Creating Virtual Machines on Oracle VM."
2. Go to the Oracle VM Manager and check the status of the virtual machine.

   If the status is **`Error`**, view the details of the virtual machine and click the link to the log file. The *Oracle VM Manager User's Guide* ("Troubleshooting" section) provides information to help you troubleshoot virtual machine problems by using log messages.
3. Fix problems in the file system of the virtual machine by following the procedure described in Section 7.2.2, "Fixing Problems in the File System of a Virtual Machine."

4. Check whether Section 7.2.3, "Solutions to Specific Run-Time Problems" contains the solution to the problem.
5. Check whether Section 7.2.4, "Frequently Asked Questions: Run-Time" contains the answer to your question.
6. Contact Oracle for help, as described in Section 7.3, "Contacting Oracle for Help."

7.2.2 Fixing Problems in the File System of a Virtual Machine

The virtual machine might crash due to problems in its file system. The problems could, for example, be caused by interruptions while executing Image Tool options (such as **-f** and **-r**) that access the file system, and result in the following error message:

```
EBUSY - Device or resource busy! (Filesystem already mounted!)
```

When the virtual machine restarts, Oracle JRockit Virtual Edition checks the file system automatically.

If the virtual machine cannot be restarted, you can find and fix file system problems by running the following Image Tool command.

> **Note:** Repairing a file system of a virtual machine image could lead to loss of data. So, before attempting to repair the file system, back up the contents in virtual machine disk by copying the virtual machine image.

```
java -jar jrockitve-imagetool.jar --repair vm_cfg [prompt_type]
```

In this command, `vm_cfg` is the path and name of the virtual machine configuration file (`vm.cfg`), and `prompt_type` indicates whether problems found should be fixed automatically (`auto`, the default value), not fixed at all (`check`), or fixed only after the user specifically confirms that each problem should be fixed (`prompt`).

For more information, see --repair in Appendix A, "Image Tool Command-Line Option Reference."

7.2.3 Solutions to Specific Run-Time Problems

ERROR: Configured IP [...] in use by MAC

When the IP address specified for the server is already in use, the following error message is displayed:

```
00:00:01 [net WRN] Configured IP ip_address in use by MAC: mac_address
00:00:02 [net WRN] Network stack initialization FAILED: 98
```

Use the **-r** option to verify the IP address. Make sure that the IP address is correct and none of the running virtual machines use the same address. If the problem persists, contact your system administrator to obtain a new IP address.

For more information about using the **-r** option to reconfigure a virtual machine, see Appendix A, "Image Tool Command-Line Option Reference."

My Virtual Machine Is Freezing

If you know the IP address of the virtual machine, ping it.

- A failed ping indicates that the virtual machine is no longer running; it has crashed or hung (see "My Virtual Machine Has Crashed").

  Note that the ping can also fail if network security features, such as a firewall, are enabled in your system or network.

- A successful ping indicates that the network stack, scheduler, and other operating system functions are working.

  Try to determine where the virtual machine is freezing by reviewing the thread dump. For information about generating a thread dump, see "How do I take thread dumps?".

My Virtual Machine Has Crashed

When a virtual machine crashes, the Oracle JRockit JVM creates a text crash file in the current working directory of the virtual machine; it also writes the crash file to `stderr` and to the virtual console log.

Review the crash file to determine the cause of the crash. For more information, see Section 7.2.5, "Using Crash Files."

Error: 2, 'Invalid kernel', 'xc_dom_parse_elf_kernel: corrupted ELF image\n'

When a virtual machine starts, Oracle VM copies the Oracle JRockit Virtual Edition ELF (executable and linking format) kernel image to the `/var/run/xend/boot` directory, and then reads the image. If the `/var` disk partition is full (for example, on account of frequent of core dumps in the `/var/xen/dump` directory), Oracle VM cannot copy the ELF kernel image to the `/var/run/xend/boot` directory; so it displays the error message.

Make sure that the `/var` disk partition has sufficient space.

7.2.4 Frequently Asked Questions: Run-Time

Where can I find crash files?

See "Location of the Oracle JRockit JVM and Oracle JRockit Virtual Edition Crash Files".

How do I take thread dumps?

A thread dump is a snapshot of the state of all threads that are part of a virtual machine image.

To take thread dumps in Oracle JRockit Virtual Edition, do one of the following:

- Run the `print_threads` diagnostic command from Oracle JRockit Mission Control.
- Press `Ctrl+\` (or the `9` key) in the virtual machine console. This is equivalent to sending a SIGQUIT signal to the process in a Linux environment

If you cannot generate thread dumps, force the virtual machine to crash by using the **`xm dump-core`** command. Provide the core dump file to Oracle Support for further diagnosis and troubleshooting.

For more information about troubleshooting by using thread dumps, see the "Using Thread Dumps" section of the *Oracle JRockit Diagnostics Guide*, which is available at `http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html`

How do I send a Diagnostic Command?

Use Oracle JRockit Mission Control.

For more information, see Section 7.2.6, "Using Diagnostic Commands."

Where should I store the ctrlhandler.act file?

Save the `ctrlhandler.act` file in the current working directory of the file system of the virtual machine. For information about finding out the current working directory, see "How do I find out the current working directory of a virtual machine image?".

How do I shut down the virtual machine from the virtual console?

Press the `F5` (or `5`) key and then press `Ctrl+c` in the virtual machine console. This is equivalent to sending a SIGINT signal.

How do I send a SIGABRT signal?

Oracle JRockit Virtual Edition does not provide an option to send a SIGABRT signal.

Use the **`xm dump-core`** command to generate a core dump.

How do I find out the current working directory of a virtual machine image?

You can find out the current working directory by running the following command:

> **Caution:** You must shut down the virtual machine before running this command.

`java -jar` jrockitve-imagetool.jar `-r` *`vm_cfg`* `get working-dir`

In this command, `vm_cfg` is the path and name of the virtual machine configuration file (`vm.cfg`).

7.2.5 Using Crash Files

When the JVM crashes, Oracle JRockit Virtual Edition creates a text crash file (`.dump`) and a binary crash file. Note that if the Oracle JRockit Virtual Edition kernel (rather than the JVM) notices the crash first, then the crash files are not generated.

For more information about crash files and crash file sizing, see the "Understanding Crash Files" section of the *Oracle JRockit Diagnostics Guide*, which is available at `http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html`.

Location of the Oracle JRockit JVM and Oracle JRockit Virtual Edition Crash Files

- The name of the Oracle JRockit **text crash file** is `jrockit.4711.dump`. It is written to the current working directory in the file system of the virtual machine.

  For information about finding out the current working directory, see "How do I find out the current working directory of a virtual machine image?"

  > **Note:** Oracle JRockit Virtual Edition does not support using the environment variable `JROCKIT_DUMP_PATH` to specify a different location for the **text crash file**.

- The Oracle JRockit Virtual Edition **binary crash file** is written to the `/var/xen/dump` directory on the Oracle VM Server.

Enabling Creation of Crash Files on Oracle VM

The virtual machine configuration file (`vm.cfg`) contains a parameter `on_crash`, which defines the action to be performed in the event of a system crash, as shown in the following example.

```
name="wls-ve"
bootloader="/usr/bin/pygrub"
memory=256
disk=['tap:aio:/OVS/seed_pool/wls-ve/system.img,sda1,w']
vif=['']
on_crash="coredump-destroy"
```

For a virtual machine image created by using Oracle JRockit Virtual Edition, the possible values for the `on_crash` parameter are `restart`, `destroy`, `coredump-restart` and `coredump-destroy`.

> **Note:** For crash files to be written, sufficient disk space must exist in the `/var` partition on the Oracle VM Server.

Disabling Creation of Crash Files

- To disable creation of the text crash file, use the `-XXnoJrDump` option.
- To disable creation of the binary crash file, change the value of the `on_crash` parameter in the `vm.cfg` file.

> **Caution:** Changing the `on_crash` parameter in the `vm.cfg` file renders the product unsupported.

7.2.6 Using Diagnostic Commands

You can send diagnostic commands to a running virtual machine in one of the following ways:

- By using the Ctrl-Break handler from Oracle JRockit Mission Control.
- By using the JRockit Management Console in Oracle JRockit Mission Control.

> **Note:** The default Ctrl-Break handler (triggered by pressing **`Ctrl+\`**) is the `print_threads` command. However, if the `ctrlhandler.act` file is available in the current working directory, the diagnostic commands listed in the file are executed.

For more information about using the Ctrl-Break handler and JRockit Management Console, see the "Running Diagnostic Commands" section of the *Oracle JRockit Diagnostics Guide*, which is available at `http://download.oracle.com/docs/cd/E13150_01/jrockit_jvm/jrockit/webdocs/index.html`.

7.2.7 Viewing the Virtual Machine Log File

The log messages generated by Oracle JRockit Virtual Edition are useful for diagnosing and troubleshooting run-time problems.

The messages written to the standard output streams (`stdout` and `stderr`) are also written to a log file, which, by default, is `/jrockitve/log/jrockitve.log` within the file system of the virtual machine image. You can change the log file name and location by using the `-r vm_cfg set console-log-path` operation of the Image Tool.

Note: Oracle VM Server logs are in the `/var/log/xen` directory of the Oracle VM server. The log file is named `xend.log`. It contains only Oracle VM server-related messages.

To view the contents of the virtual machine log file (if it exists within the file system of the virtual machine), **shut down the virtual machine** and then run the following command:

```
java -jar jrockitve-imagetool.jar --get-log vm_cfg [output-file]
```

In this command, `vm_cfg` is the path and name of the virtual machine configuration file (`vm.cfg`), and `output-file` is the path and name of the file to which you want the contents of the log file to be written. If you do not specify the output file, the output of the command is printed to `stdout`.

7.3 Contacting Oracle for Help

If you have a service agreement with Oracle, you can contact Oracle Support (`http://support.oracle.com`) for help with Oracle JRockit Virtual Edition problems.

Steps to be Taken Before Contacting Oracle Support

Before contacting Oracle Support, do the following:

- Try all the appropriate diagnostics and troubleshooting guidelines described in this document (*Oracle Fusion Middleware User's Guide for Oracle JRockit Virtual Edition*).
- Check whether the problem (or a similar problem) has been discussed in the Oracle JRockit Virtual Edition forum at `http://forums.oracle.com/`.

  If the information available on the forum is not sufficient to help you solve the problem, post a question on the forum. Other Oracle JRockit Virtual Edition users on the forum might respond to your question.
- Document the environment and the actions performed just before you encountered the problem.
- Where applicable, try to restore the original state of the system and reproduce the problem using the documented steps. This helps to determine whether the problem is reproducible or an intermittent issue.
- If the issue can be reproduced, try to narrow down the steps for reproducing the problem. Problems that can be reproduced by small test cases are typically easier to diagnose when compared with large test cases.

Narrowing down the steps for reproducing problems enables Oracle Support to provide solutions for potential problems faster.

> **Note:** When you send files (`.dump`, `.core`, and so on) to Oracle Support, remember to provide the MD5 checksum value for each file, so that Oracle Support personnel can verify the integrity of the files before using them for troubleshooting the problem.

Information to be Provided When Contacting Oracle Support

When you contact Oracle for support, provide the following information.

- A brief description of the problem
- The version number of Oracle JRockit Virtual Edition

  To find out the version number, run the following command:

  ```
  java -jar jrockitve-imagetool.jar -v [vm_cfg]
  ```

  In this command, `vm_cfg` is the path and name of the virtual machine configuration file (`vm.cfg`).
- For problems with the Image Tool, provide the following information:
  - The name of the command-line option for which you need help
  - Debug log for the command

    You can generate debug log messages for the Image Tool option for which you need help from Oracle, by running the following command:

    ```
    java -jar jrockitve-imagetool.jar option --log verbose
    ```

    `option` is the Image Tool option (`-c`, `-f`, and so on) for which you want to view verbose messages.

    This command displays detailed messages on the screen. Copy the log messages that are displayed on the screen to a text file.
- For run-time problems, provide the following:
  - Virtual machine log file

    For information about finding, viewing, and extracting the virtual machine log file, see Section 7.2.7, "Viewing the Virtual Machine Log File."
  - Thread dump

    For information about creating thread dumps, see "How do I take thread dumps?"
  - Text crash file

    For more information about viewing crash files, see Section 7.2.5, "Using Crash Files."

A

Image Tool Command-Line Option Reference

This appendix provides alphabetically sorted reference information about the command-line options that the Oracle JRockit Virtual Edition Image Tool supports.

You can type the command-line options in short form (if available) or in long form, as shown in the following examples:

- **Short form**: `java -jar jrockitve-imagetool.jar` **`-h`**
- **Long form**: `java -jar jrockitve-imagetool.jar` **`--help`**

> **Note:** When specifying local directory paths, local file names, and Java arguments containing spaces, enclose the values in double quotation marks.
>
> For example, to create a configuration file named `config 1.xml` by using the `-c` option, treat the space in the file name as follows:
>
> ```
> java -jar jrockitve-imagetool.jar -c "config 1.xml"
> ```

Documentation Conventions for Command-Line Syntax

The following conventions are used in this document to show the syntax for command-line options:

Convention	Meaning	Example
Plain text	Type the text as is.	`option`
Italicized text	Placeholder for mandatory parameter; substitute with appropriate text.	*`parameter`*
Text in square brackets	Optional argument.	`[argument]`
Asterisk (*) after text in square brackets	Optional argument; you can specify multiple instances.	`[argument]*`
Plus symbol (+) after text	Mandatory argument; you can specify multiple instances.	`argument+`
Vertical bar symbol (|) between text	Optional values: specify one of them.	`opt1|opt2`

The Image Tool supports the following command-line options.

Caution: Before using any option of the Image Tool to view or modify the configuration or file system of a virtual machine, shut down the virtual machine.

Do not attempt to use the Image Tool to view or modify the configuration or file system of a running virtual machine.

- -a (--assemble)
- -c (--create-config)
- --create-full-config
- -d (--disassemble)
- -f (--file)
- --force
- --get-log
- -h (--help)
- -l (--log)
- -p (--patch)
- -r (--reconfigure)
- --reconfigure-service
- --repair
- -v (--version)

-a (--assemble)

The `-a` option creates (assembles) a hypervisor-specific virtual machine image from a Java application, by using the configuration parameters specified in an XML file.

> **Note:** The XML file that you use to create a virtual machine image is not retained as an XML file anywhere in the file system of the virtual machine image. During the assembly process, the configuration settings defined in the input XML file are stored within the virtual machine image.
>
> Subsequently, if you reconfigure the virtual machine by using the -r (--reconfigure) option, the configuration changes are not reflected in the original XML file that you used to assemble the virtual machine image.
>
> At any point, you can extract the current configuration settings of a virtual machine and rebuild an XML file that reflects the configuration state of the virtual machine, by using the -d (--disassemble) option. You can also `get` the configuration settings by using the **`-r`** `vm.cfg get config` option.

Syntax

```
java -jar jrockitve-imagetool.jar -a config_file.xml output_dir [hypervisor]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description	Mandatory / Optional	Default Value
`config.xml`	The path and name of the XML file that contains the configuration parameters for the virtual machine	Mandatory	
`output_dir`	The directory in which the virtual machine image should be created	Mandatory	
`hypervisor`	The hypervisor for which the virtual machine image should be created	Optional	`ovm` (Oracle VM) **Note**: Oracle VM is the only supported hypervisor.

Example

```
java -jar jrockitve-imagetool.jar -a myconfig.xml app
```

This command creates a virtual machine image that can be run on Oracle VM, by using the configuration parameters specified in the `myconfig.xml` file. The resulting virtual machine image is created in the `app` directory.

Related Options

-c (--create-config)

-d (--disassemble)

-r (--reconfigure)

-c (--create-config)

The `-c` option generates a **brief** sample configuration file (in XML format) containing only those configuration parameters that are essential for creating virtual machine images. You can use this file as a template to create the final configuration file for your virtual machine.

> **Note:** You can use the --create-full-config option to create a detailed sample configuration file.

Syntax

```
java -jar jrockitve-imagetool.jar -c [config_file.xml] [vm_name]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description	Mandatory / Optional	Default Value / Behavior
config_file.xml	The name of the sample brief configuration file to be created	Optional	If you do not specify the file name, the virtual machine configuration is printed (in XML format) to `stdout`.
vm_name	Name of the virtual machine	Optional	`default-vm` The name can contain only alphanumeric characters, periods (`.`), dashes (`-`), and underscores (_).

Example

```
java -jar jrockitve-imagetool.jar -c myconfig.xml my-vm
```

This command creates a sample brief configuration file named `myconfig.xml`, as shown in Appendix C.1, "Sample Brief Configuration File." Note that the value of the `<vm-name>` element is set to `my-vm`.

Related Topics

--create-full-config

-d (--disassemble)

-r (--reconfigure)

Appendix B, "Configuration File Element Reference"

--create-full-config

The `--create-full-config` option generates a **detailed** sample configuration file (in XML format). You can use this file as a template to create the final configuration file for your virtual machine.

Syntax

```
java -jar jrockitve-imagetool.jar --create-full-config [config_file.xml] [vm_name]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description	Mandatory / Optional	Default Value / Behavior
config_file.xml	The name of the sample detailed configuration file	Optional	If you do not specify the file name, the virtual machine configuration is printed (in XML format) to `stdout`.
vm_name	Name of the virtual machine	Optional	`default-vm` The name can contain only alphanumeric characters, periods (`.`), dashes (`-`), and underscores (`_`).

Example

```
java -jar jrockitve-imagetool.jar --create-full-config config.xml
```

This command creates a sample detailed configuration file named `config.xml`, as shown in Appendix C.2, "Sample Detailed Configuration File."

Related Topics

-c (--create-config)

-d (--disassemble)

-r (--reconfigure)

Appendix B, "Configuration File Element Reference"

-d (--disassemble)

The `-d` option disassembles a virtual machine image into the following components:

- A directory named `root`, containing the Java application in its current on-disk state
- The `jrockitve.bin` file
- A configuration XML file that the Image Tool builds by extracting configuration settings stored in a virtual machine image
- A log file containing the contents of all the console log files (if the log files are within the file system of the virtual machine), concatenated in the old-to-new order

Syntax

```
java -jar jrockitve-imagetool.jar -d vm_cfg output_dir
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
vm_cfg	The path and name of the configuration file (`vm.cfg`) corresponding to the virtual machine image that you want to disassemble
output_dir	The directory in which you want the components of the disassembled virtual machine image to be placed

Example

```
java -jar jrockitve-imagetool.jar -d app\vm.cfg dis_app
```

This command disassembles the virtual machine image that exists in the `app` directory and places the disassembled components in the `dis_app` directory.

Related Topics

-a (--assemble)

Disassembling and Re-Creating Virtual Machine Images

-f (--file)

The `-f` option enables you to perform the following tasks in the file system of a virtual machine image:

- Copy files from the file system of a virtual machine image to the local file system and vice versa.
- View directory listings.
- Move and rename files.
- Create and delete directories.
- Find files.
- Display the contents of text files.

> **Caution:** Unless you are familiar with the file system of a virtual machine image, do not use the `-f` option for modifying the file system. For more information, see Section 2.4, "About the File System of a Virtual Machine."

Syntax

```
java -jar jrockitve-imagetool.jar -f vm_cfg operation [parameter]*
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

When specifying directory paths in the file system of a virtual machine, use slashes (/) as path delimiters. A slash (/) at the beginning of the path represents the root directory. A period (.) at the beginning of the path represents the current directory. Two periods (..) indicate the parent directory.

For directory paths in the local file system, use the appropriate delimiter: slash (/) on UNIX-like operating systems and backslash (\) on Windows.

Arguments

Argument	Description
`vm_cfg`	Path and name of the configuration file corresponding to the virtual machine file system that you want to view or modify
`operation`	The file operation (see Table A–1) to be performed
`parameter`	The parameters required for the specified file operation

The parameters that you can specify vary depending on the operation. Table A–1, lists the parameters for each operation.

Table A–1 Operations and Parameters of the -f Option

Operation and Parameters	Purpose
get *path_in_virtual_machine path_in_local_filesystem*	Copy virtual machine files to the local file system.
rget *path_in_virtual_machine path_in_local_filesystem*	Copy virtual machine files recursively to the local file system.
put *path_in_local_filesystem path_in_virtual_machine*	Copy files from the local file system to the file system of a virtual machine image.
rput *path_in_local_filesystem path_in_virtual_machine*	Copy files recursively from the local file system to the file system of a virtual machine image.
ls [*file_pattern*]*	List files and directories.
rls [*file_pattern*]*	List files and directories recursively.
mv *from_file_pattern to_file_pattern*	Move or rename files.
mkdir *directory_name*+	Create a directory.
rm *file_pattern*+	Delete files.
rrm *file_pattern*+	Delete files recursively.
rmdir *directory_pattern*+	Delete directories.
cat *file_pattern*+	Display the contents of text files.
find *pattern*	Search for files matching the specified pattern.

Note: When specifying local directory paths and file names that contain spaces or wildcards, enclose the parameter in double quotation marks.

For example, to copy all the `.prop` files from a local directory named "`my app`" to the root directory in the file system of the virtual machine, use double quotation marks as shown below:

```
java -jar jrockitve-imagetool.jar -f vm_cfg put "my app\*.prop" ./
```

Examples

In all of these examples, the virtual machine configuration file (`vm.cfg`) is assumed to reside in the `app` directory.

- **get**

  ```
  java -jar jrockitve-imagetool.jar -f app/vm.cfg get /jrockitve/log/* /tmp
  ```

  This command copies all the files in the `/jrockitve/log` directory in the file system of a virtual machine image to the `/tmp` directory of the local file system.

- **put**

  ```
  java -jar jrockitve-imagetool.jar -f app/vm.cfg put new/app.class app
  ```

  This command copies a Java class file named `app.class` from the `new` directory in the local file system to the `/app` directory in the file system of the virtual machine image.

- **ls**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg ls /jrockitve/log
```

This command lists the contents of `/jrockitve/log` directory in the file system of the virtual machine image.

- **mv**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg mv app/app.class new.class
```

This command renames the `app/app.class` file in the file system of the virtual machine image to `new.class` and moves the renamed file to the root (`/`) directory.

- **mkdir**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg mkdir app/data
```

This command creates a directory named `data` in the `/app` directory in the file system of the virtual machine image.

- **rm**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg rm app/app.class
```

This command deletes the `app.class` file from the `/app` directory in the file system of the virtual machine image.

- **rmdir**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg rmdir app/data
```

This command deletes the `/app/data` directory from the file system of the virtual machine image. If the directory is not empty, the Image Tool displays an error message.

- **cat**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg cat
/jrockitve/log/jrockitve.log
```

This command displays the contents of the `/jrockitve/log/jrockitve.log` file in the file system of a virtual machine image.

- **find**

```
java -jar jrockitve-imagetool.jar -f app/vm.cfg find log
```

This command searches the file system of a virtual machine image for files and directories that have `log` in the name, and displays the results on the screen.

Related Topics

Modifying the File System Within a Virtual Machine Image by Using the Image Tool

--force

You can include the `--force` option with other Image Tool options such as `-a`, `-c`, and `-d`, to override warnings that occur while executing those options.

For example, when you use the `-d` option to disassemble a virtual machine image, if the specified output directory is not empty, the command exits with a warning message. You can force the Image Tool to write to the nonempty directory by including `--force`.

Syntax

```
java -jar jrockitve-imagetool.jar option --force
```

You can include `--force` either before or after the options it accompanies.

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
option	The Image Tool option for which you want to override warnings

Example

When creating (assembling) a virtual machine image, if you specify a nonempty directory (say, `app`) as the output directory, the Image Tool exits with the following message:

```
Assembling the image ...
error: Cannot overwrite directory "app".
```

To force the Image Tool to put a virtual machine image in the nonempty directory, include `--force` with the `-a` option, as follows:

```
java -jar jrockitve-imagetool.jar -a myconfig.xml app --force
```

or

```
java -jar jrockitve-imagetool.jar --force -a myconfig.xml app
```

--get-log

The `--get-log` option prints the contents of the virtual machine console log files, if they exist within the file system of the virtual machine. When multiple log files exist (as a result of log-file rotation), the contents are concatenated in the old-to-new order.

Notes:

- The default location of the console log messages of the virtual machine is `/jrockitve/log/jrockitve.log` in the file system of the virtual machine. You can specify a different location by using the `-r vm.cfg set console-log-path` command.

  You can find out the current location of the console log files by using the `-r vm.cfg get console-log-path` command.
- If the log file is stored outside the file system of the virtual machine (say, in an NFS location), the `--get-log` option does not work.

Syntax

```
java -jar jrockitve-imagetool.jar --get-log vm_cfg [output_file]
```

Note: For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description	Mandatory / Optional
`vm_cfg`	Path and name of the configuration file corresponding to the virtual machine file system that you want to view or modify	Mandatory
`output_file`	The path and name of the file to which you want the contents of the log file to be written	Optional If you do not specify the output file, the command prints the contents of the log files to `stdout`.

Example

```
java -jar jrockitve-imagetool.jar --get-log app/vm.cfg /logs/100215.txt
```

This command copies the contents of the log files to the `/logs/100215.txt` file.

Related Topics

-r (--reconfigure)

-h (--help)

The `-h` option displays help for the specified Image Tool option.

Syntax

```
java -jar jrockitve-imagetool.jar -h [option]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
`option`	The Image Tool option for which help should be displayed

Example

```
java -jar jrockitve-imagetool.jar -h -a
```

This command displays help for the `-a` option.

> **Note:** You can view help for a specific operation of an option. For example, for help on the `set config` operation of the `-r` option, run the following command:
>
> ```
> java -jar jrockitve-imagetool.jar -h -r set config
> ```

-l (--log)

You can include `-l` (lowercase letter `L`) in an Image Tool command, to specify the types of messages that should be printed for the command.

Syntax

```
java -jar jrockitve-imagetool.jar option -l level
```

You can include `--log` either before or after the options it accompanies.

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
`option`	The option for which you want messages to be printed
`level`	The types of log messages that should be printed

Table A–2 lists the possible log levels and the output stream (`stdout` or `stderr`) for each message type at various log levels.

Table A–2 Output Streams for Message Types at Various Log Levels

Log Level	Error Messages	Warning Messages	Information Messages	Confirmation Messages
`quiet`	`stderr`	None	None	None
`brief` (default)	`stderr`	`stderr`	`stdout`	`stdout`
`verbose`	`stderr`	`stderr`	`stdout`	`stdout`

Example

```
java -jar jrockitve-imagetool.jar -r app/vm.cfg get vm-name -l verbose
```

This command causes the following log messages to be displayed while the Image Tool retrieves the name of the virtual machine.

```
[INFO  image.tool            ] Synchronizing the image ...
[INFO  image.tool            ] Done
default-vm
```

When you run the same command with the `brief` or `quiet` log level instead, only the virtual machine name is printed.

-p (--patch)

You can use the `-p` option to replace the Oracle JRockit Virtual Edition binary (JVM, the kernel, and services) within the virtual machine with a new version.

> **Note:** The `-p` option does not modify the Java application within the virtual machine, the configuration of the virtual machine, and the virtual machine log files.
>
> - To change the configuration of the virtual machine, use the -r (--reconfigure) option.
> - To modify the Java application, use the -f (--file) option.

Syntax

```
java -jar jrockitve-imagetool.jar -p vm_cfg [path/]jrockitve.bin
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
`vm_cfg`	Path and name of the configuration file corresponding to the virtual machine file system that you want to patch
`path`	The path of the directory that contains the new version of the `jrockitve.bin` file

Example

```
java -jar jrockitve-imagetool.jar -p app/vm.cfg /patches/v2/jrockitve.bin
```

This command replaces the Oracle JRockit Virtual Edition binary within the virtual machine with the `jrockitve.bin` file that exists in the `/patches/v2` directory on the local file system.

If the patching fails for any reason (for example, due to insufficient disk space), the virtual machine image is reverted to its state before the patching was attempted, and an error message is displayed.

After running the command, you can check whether the command was successful by using the -v (--version) option to view the version of the new Oracle JRockit Virtual Edition binary.

For information about the limitations of the `-p` option, see Appendix D, "Known Issues."

-r (--reconfigure)

The `-r` option enables you to view, add, change, and remove configuration settings for a virtual machine by modifying the virtual machine configuration directly.

> **Note:** If required, you can extract a configuration XML file corresponding to the modified virtual machine configuration by using the **`-r`** `vm.cfg get config` option.

Syntax

```
java -jar jrockitve-imagetool.jar -r vm_cfg operation field [parameter]*
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
vm_cfg	Path and name of the configuration file (`vm.cfg`) of the virtual machine to be reconfigured.
operation	The operation to be performed on the specified field: ▪ `get`: Display the current configuration settings of the specified field. ▪ `set`: Set or change the value of the specified field. ▪ `add`: Add the specified field to the configuration. ▪ `remove`: Remove the specified field from the configuration. ▪ `enable`: Enable the specified service. ▪ `disable`: Disable the specified service. Table A–3 shows the operations that are supported for each configurable field.
field	The field to be reconfigured. Table A–3 lists the values that you can specify for this argument.
parameter	Parameters for the specified field. The values that you should specify for this argument vary depending on the field to be reconfigured and the operation you want to perform. **Note**: When you run `-r set` for a field without specifying any parameter, all the parameters relevant for operations supported for that field are displayed. For more information, see Detailed Syntax for Operations of --reconfigure.

Table A–3 lists the fields that you can reconfigure and the reconfiguration operations that you can perform. Most fields correspond to either an element or an attribute in the configuration XML file.

Table A–3 Fields and Operations of the -r Option

Field	Supported Operations	Affected XML Element Name (Attribute)
config	get, set	jrockitve-config (mandatory)

Table A–3 (Cont.) Fields and Operations of the -r Option

Field	Supported Operations	Affected XML Element Name (Attribute)
console-log	get, remove	None
console-log-path	get, set	console-log-path (optional)
cpus	get, set	jrockitve-config (mandatory) (attribute: cpus)
disk-size	set	disks (mandatory) (attribute: size)
disks	get	disks (mandatory)
enabled-services	get	services (optional)
free-disk-size	get	None
installed-services	get	services (optional)
java-arguments	get, set	java-arguments (mandatory)
locale	get, set	locale (optional)
locale-encoding	set	encoding (optional)
locale-timezone	set	timezone (optional)
memory	get, set	jrockitve-config (mandatory) (attribute: memory)
mount	add, remove, set	mounts (mandatory)
mounts	get	mounts (mandatory)
network-dns-host	add, remove	dns (optional)
network-dns-hosts	get, remove	
network-dns-name	add, get, remove, set	lookup-order (optional) (attribute: suffix)
network-dns-names	get	lookup-order (optional) (attribute: suffix)
network-dns-server	add, get, remove, set	server-order (optional) (attribute: ip)
network-dns-servers	get	server-order (optional) (attribute: ip)
network-hostname	get, remove, set	network (mandatory) (attribute: hostname)
network-nic	add, get, remove, set	nics (mandatory)
network-nics	get	nics (mandatory)
service	disable, enable	service (optional)
service-argument	add, remove	arguments (optional)
service-arguments	get	arguments (optional)
vm-name	get, set	vm-name (mandatory)
working-dir	get, set	working-dir (optional)

Detailed Syntax for Operations of --reconfigure

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

config

```
get config [destination.xml]
set config source.xml
```

View or change the configuration of a virtual machine.

Parameter	Description
destination	Name of the XML file in which the extracted configuration information should be stored. If you do not specify this parameter, the command displays the configuration details in XML format on the screen.
source	Name of the XML file that contains the configuration settings to be defined for the virtual machine.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get config /work/config.xml`

  This command extracts the current configuration of the virtual machine and copies the configuration, in XML format, to the `/work/config.xml` file on the local system.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set config /new/config.xml`

  This command replaces the configuration of the virtual machine with the settings defined in the `/new/config.xml` file on the local system.

console-log

```
get console-log
remove console-log
```

View or remove the log file of a virtual machine.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get console-log`

  This command displays the contents of the virtual machine log file. The output is the same as that of the **--get-log** command.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove console-log`

  This command deletes the virtual machine log file.

console-log-path

get console-log-path
set console-log-path *logfile_path-and-name*

View or change the directory and file in which log messages are stored. The default log file is `/jrockitve/log/jrockitve.log`.

Parameter	Description
logfile_path-and-name	The path and name of the **file** to which console log messages should be written. Do not specify a directory as the value of this parameter.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get console-log-path`

  This command displays the path and name of the virtual machine log file.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set console-log-path /jrockitve/log/myvm.log`

  This command sets `/jrockitve/log/myvm.log` as the virtual machine log file.

cpus

get cpus
set cpus *number*

View or change the number of CPUs available to a virtual machine.

The number specified must be greater than zero. You can specify a maximum of eight virtual CPUs.

> **Note:** In the current release, Oracle recommends that you configure a single CPU, because configuring the virtual machine for multiple virtual CPUs affects performance.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get cpus`

  This command displays the number of CPUs currently defined for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set cpus 2`

  This command changes the number of CPUs defined for the virtual machine to 2.

disk-size

set disk-size *index size*

Set the hard disk space available to a virtual machine.

Parameter	Description
index	Index number of the disk for which you want to set the size **Note**: Only one file-based disk is supported in the current release.
size	Disk size. Specify a numerical value followed by one of the following units (**not case sensitive**) ■ `K` or `KB`: Kilobytes ■ `M` or `MB`: Megabytes ■ `G` or `GB`: Gigabytes ■ `T` or `TB`: Terabytes If you do not specify any unit, megabytes is assumed. If you leave a space between the size and the unit, enclose the number and the unit in quotes as shown in the example that follows.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg set disk-size 1 "200 MB"`

This command defines 200 MB as the size of the disk represented by index number 1.

> **Note:** When you change `disk-size`, the virtual machine image is resized (shrunk or expanded) depending on the new size specified. If the image cannot be expanded to the specified size due to insufficient disk space or if the new size specified is lower than the current size of the image, the command returns error messages.

disks

`get` `disks`

View the settings for the disk defined for a virtual machine.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get disks`

The command displays the index number, ID, type, and size of each disk defined for the virtual machine, in the following format:

```
#1 id=root type=file 976 MB
```

enabled-services

`get` `enabled-services`

View the names of the installed services that are currently enabled.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get enabled-services`

The command displays the names and descriptions of the services that are currently enabled, as shown in the following example output.

```
sshd (An SSH2 implementation with SCP and SFTP support)
```

free-disk-size

```
get free-disk-size id
```

View the free disk space on the specified disk ID.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get free-disk-size root`

This command displays the amount of free space available on the disk with the ID `root`.

installed-services

```
get installed-services
```

View the names of the services that are currently installed in a virtual machine.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get installed-services`

The command displays the names of the services that are currently installed and a brief description of each service, as shown in the following example output.

```
jmxstat (JRockitVE kernel statistics MBean)
sshd (An SSH2 implementation with SCP and SFTP support)
sysstat (JRockitVE kernel sysstat statistics)
```

java-arguments

```
get java-arguments
set java-arguments "arguments"
```

View or change Java arguments.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get java-arguments`

  This command displays the Java command-line arguments defined currently for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set java-arguments "HelloWorld"`

  This command sets `HelloWorld` as the Java command-line argument for the virtual machine.

locale

```
get locale
set locale locale [timezone] [charset]
```

View or change the locale for the virtual machine.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get locale`

This command displays the locale information for the virtual machine: language, time zone, and character encoding.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set locale en_US America/Denver Cp1252`

  This command changes the locale settings for the virtual machine to the specified values.

locale-timezone

set locale-timezone *new_timezone*

Set the time zone for the locale.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg set locale-timezone America/Phoenix`

This command changes the timezone of the virtual machine to the specified value.

locale-encoding

set locale-encoding *new_charset*

Set the encoding character set for the locale.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg set locale-encoding iso-8859-1`

This command changes the character encoding of the virtual machine to the specified value.

memory

get memory
set memory *size*

View or change the memory available to a virtual machine.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get memory`

  This command displays the memory size that is currently defined for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set memory 512M`

  This command changes the memory size of the virtual machine to the specified value.

  You can specify the size in the following units (**not case sensitive**):

  - `K` or `KB`: Kilobytes
  - `M` or `MB`: Megabytes
  - `G` or `GB`: Gigabytes
  - `T` or `TB`: Terabytes

If you do not specify any unit, megabytes is assumed.

mount

```
add mount type mount-point options...
set mount index type mount-point options...
remove mount index
```

Add, remove, or configure a mount point.

Parameter	Description
index	Index number of the mount point that you want view, change, remove. You can find out the index number by using the `get mounts` operation.
type	The type of mount that you want to add or configure. The following mount types are supported: ■ `nfs` ■ `native` **Note**: Only one native mount (the root disk of the virtual machine) is supported in the current release.
mount-point	The directory, within the virtual image, at which the external file system should be mounted.
options	The options depend on the mount type. ■ **nfs**: *server server-path option1 option2 ... optionN* ■ **native**: *device option1 option2 ... optionN*

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add mount nfs /home nfs.host.com /export/home uid=501 gid=502`

  This command adds an NFS mount point at the `/home` directory within the virtual machine, pointing to the `/export/home` directory on the `nfs.host.com` server, with `501` as the user ID and `502` as the group ID.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set mount 2 nfs /home nfs.newhost.com /export/home uid=501 gid=502`

  This command changes the settings of the NFS mount point represented by index number 2, to the specified values.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove mount 2`

  This command removes the NFS mount point represented by index number 2.

mounts

```
get mounts
```

View mount points.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get mounts`

The command displays the index number, mount type, and other parameters of all the mount points currently defined for the virtual machine, as shown in the following example output.

```
#1 native /
#2 nfs /log logs.mycompany.com /logger/logs,uid=513,gid=502
```

The first line in the example output shows details of the native mount pointing to the root (`/`) directory of the virtual machine file system. The second line shows the directory `/logger/logs` on the `logs.mycompany.com` server mounted as an NFS mount point at the `/log` directory in the file system of the virtual machine.

network-dns-host

add network-dns-host *ip-address host*
remove network-dns-host *ip-address host*

Add or remove a static host entry in the DNS hosts table.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add network-dns-host 192.168.66.2 myhost`

  This command adds the specified IP address-hostname pair to the DNS table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-dns-host 192.168.66.2 myhost`

  This command removes the specified IP address-hostname pair from the DNS .

network-dns-hosts

get network-dns-hosts
remove network-dns-hosts *ip-address*

View all static host entries in the DNS hosts table or remove entries corresponding to the specified IP address.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-dns-hosts`

  This command displays the entries in the DNS table of the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-dns-hosts 192.168.66.2`

  This command removes the DNS entry corresponding to the specified IP address.

network-dns-name

add network-dns-name *domain-name*
get network-dns-name *index*
remove network-dns-name *index*
set network-dns-name *index domain-name*

Add, view, remove, or change a DNS domain name.

Note: The DNS domain names are tried in the order in which they are defined in the lookup table. Make sure that the primary domain names are defined first.

Parameter	Description
index	Index number of the entry (in the DNS server name lookup table) that you want to view, change, or remove. For example, if you run `get network-dns-name` for a virtual machine for which two DNS server names are defined, the output of the command would be as shown below: **#1** `one.example.com` **#2** `two.example.com` The number shown at the beginning of each line in the output indicates the index number of the entry in the DNS server name lookup table.
domain-name	Domain name for the DNS server entry to be changed or added (example: `mycompany.com`).

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add network-dns-name example.com`

  This command adds the specified domain name to the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-dns-name 1`

  This command displays the domain name corresponding to the first entry in the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set network-dns-name 1 two.example.com`

  This command changes the domain name for the first entry in the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-dns-name 1`

  This command removes the first entry in the DNS server name lookup table.

network-dns-names

`get network-dns-names`

View DNS domain names.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-dns-names`

The command displays the entries in the DNS server name lookup table.

network-dns-server

`add network-dns-server` *`ip-address`*

```
get network-dns-server index
remove network-dns-server index
set network-dns-server index ip-address
```

Add, view, remove, or change a DNS IP address.

> **Note:** The DNS servers are queried in the order in which they are defined in the lookup table. Make sure that the primary DNS servers are defined first.

Parameter	Description
index	Index number of the entry (in the DNS server lookup table) that you want to view, change, or remove. For example, if you run `get network-dns-server` for a virtual machine for which two DNS servers are defined, the output of the command would be as shown below: **#1** 172.22.23.24 **#2** 172.22.23.25 The number shown at the beginning of each line in the output is the index number of the entry in the DNS server lookup table.
ip-address	IP address for the DNS server to be changed or added.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add network-dns-server 172.22.23.26`

  This command adds the specified IP address to the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-dns-server 1`

  This command displays the IP address corresponding to the first entry in the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set network-dns-server 1 172.22.23.27`

  This command changes the IP address for the first entry in the DNS server name lookup table.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-dns-server 1`

  This command removes the first entry in the DNS server name lookup table.

network-dns-servers

```
get network-dns-servers
```

View DNS server IP addresses.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-dns-servers`

The command displays the entries in the DNS server name lookup table.

network-hostname

```
get network-hostname
remove network-hostname
set network-hostname hostname
```

View, remove, or change the network host name.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-hostname`

  This command displays the network hostname of the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-hostname`

  This command removes the network hostname of the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set network-hostname myhost.com`

  This command sets the network hostname of the virtual machine to the specified value.

network-nic

```
add network-nic network nic-type
add network-nic network nic-type ip-address netmask gateway mac
add network-nic nic-type
add network-nic nic-type ip-address
add network-nic nic-type ip-address netmask gateway mac
get network-nic index
set network-nic index network nic-type
set network-nic index network nic-type ip-address netmask gateway mac
set network-nic index nic-type
set network-nic index nic-type ip-address
set network-nic index nic-type ip-address netmask gateway mac
remove network-nic index
```

Add, view, remove, or change a network interface card (NIC).

Parameter	Description
index	Index number of the entry (in the NICs table) to be displayed or changed. **Note**: Only one NIC is supported in the current release.
network	Name for the NIC entry to be changed or added.
nic-type	Network type: `nat` or `bridged`.
ip-address	IP address for the NIC entry to be changed or added (required only if you want to use a static IP address). If you do not specify the IP address, the NIC attempts to use DHCP to find its configuration.
netmask	Net mask for the NIC entry to be changed or added (required only if you want to use a static IP address).

Parameter	Description
gateway	Gateway for the NIC entry to be changed or added (required only if you want to use a static IP address).
mac	MAC address for the NIC entry to be changed or added (required only if you want to use a static IP address)

Note: Any parameters that you do not specify (`ip-address`, `netmask`, or `gateway`) will be determined by using DHCP.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-nic 1`

  This command displays the network name (if it is defined) and the NIC type of the first NIC (index=`1`) defined for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set network-nic 1 nat`

  This command sets NAT as the type of the NIC.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove network-nic`

  This command removes the NIC.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add network-nic 1 nat`

  This command adds a NIC of type NAT.

network-nics

```
get network-nics
```

View settings of network interface cards (NICs).

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get network-nics`

The command displays the index number, network name, and NIC type of the NICs defined for the virtual machine.

service

```
enable service service-name
disable service service-name
```

Disable or enable a service.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg enable service sshd`

  This command enables the `sshd` service for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg disable service sshd`

  This command disables the `sshd` service for the virtual machine.

service-argument

```
add service-argument service-name argument
remove service-argument service-name index
```

Add or remove an argument for a service that is enabled in a virtual machine.

> **Note:** If an argument contains spaces or other special characters, prefix each space and special character with an escape character: backslash (\) on UNIX-like systems and caret (^) on Windows.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg add service-argument sshd port=4711`

  This command adds the argument `port=4711` for the `sshd` service.

  > **Note:** Some services export their arguments through the service-specific `--reconfigure-service` command. For example, to view the arguments that you can add for the `sshd` service, run the following command:
  >
  > `java -jar jrockitve-imagetool.jar` **`--reconfigure-service`** *`vm_cfg`* **`sshd get valid-arguments`**
  >
  > The output of this command is a list of arguments that the `sshd` service exports, as shown below.
  >
  > ```
  > port=<#>
  > log=<quiet|brief|verbose|debug>
  > no_auth
  > ```

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg remove service-argument sshd 1`

  This command removes the first argument that is currently configured for the `sshd` service.

service-arguments

```
get service-arguments service-name
```

View the names of arguments specified for a service.

Example: `java -jar jrockitve-imagetool.jar -r app/vm.cfg get service-arguments sshd`

The command displays all the arguments that are currently defined for the `sshd` service.

vm-name

```
get vm-name
set vm-name name
```

View or change the name of a virtual machine.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get vm-name`

  This command displays the name of the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set vm-name myvm`

  This command changes the name of the virtual machine to `myvm`.

working-dir

```
get working-dir
set working-dir path
```

View or change the name of the current working directory for a virtual machine. Relative paths are assumed to be in relation to the root (/) directory.

Examples

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg get working-dir`

  This command displays the current working directory for the virtual machine.

- `java -jar jrockitve-imagetool.jar -r app/vm.cfg set working-dir /app`

  This command changes the working directory of the virtual machine to `/app`.

Related Topics

Appendix B, "Configuration File Element Reference"

Section 4.1.3, "Disassembling and Re-Creating Virtual Machine Images"

--reconfigure-service

You can use the `--reconfigure-service` option to configure services that are enabled in the virtual machine.

Syntax

```
java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg service-name
operation field [parameter]*...]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
vm_cfg	Path and name of the configuration file (`vm.cfg`) of the virtual machine to be reconfigured.
service-name	The name of the enabled service that you want to reconfigure. You can view the names of the services that are currently enabled in the virtual machine, by using the following command: `java -jar jrockitve-imagetool.jar -r vm_cfg get enabled-services`

operation, field	The fields that you can reconfigure for a service and the operations that you can perform on the field depend on the functionality that is exposed by the service.
	You can view a list of the reconfigurable fields, by running the `--reconfigure-service` option without specifying any operation.
	Example:
	`java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd`
	This command displays the fields that you can reconfigure for the `sshd` service, and the reconfiguration operations that you can perform, as follows.
	`Operations for service "sshd":`
	`[get, remove] host-identity`
	`[get] host-uuid`
	`[add, remove] key`
	`[get] keys`
	`[add, remove] user`
	`[get] users`
	`[get] valid-arguments`
	In this example, `get`, `remove`, and `add` are the operations that can be performed on fields such as `host-identity`, `host-uuid`, and so on.

Argument	Description

parameter	The parameter that you must specify depends on the field you want to reconfigure and the operation you want to perform.
	
	You can view the parameters required for each field-operation combination, by running the `--reconfigure-service` option without specifying any parameter.
	
	Examples:
	
	■ To view the parameter to be specified for adding a user to the `sshd` service, run the following command:
	
	`java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd add user`
	
	This command displays the required parameter, as follows.
	
	`add key <username>`
	
	■ To view the parameter to be specified for adding a key to the `sshd` service, run the following command:
	
	`java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd add key`
	
	This command displays the required parameter, as follows.
	
	`add key <key-file>`

Example

```
java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd add user
jrveuser
```

This command adds the user `jrveuser` to the `sshd` service and prompts you to enter the password for the user.

--repair

The `--repair` option checks the file systems for problems.

Syntax

```
java -jar jrockitve-imagetool.jar --repair vm_cfg [repair_type]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
vm_cfg	Path and name of the configuration file (`vm.cfg`) of the virtual machine for which the file system is to be checked.
repair_type	The possible values are: ▪ `prompt`: Checks for problems and, for each problem found, prompts the user to confirm whether the problem should be fixed. ▪ `auto` (default value): Checks for and fixes problems automatically. ▪ `check`: Checks for problems but does not fix them.

Example

```
java -jar jrockitve-imagetool.jar --repair app/vm.cfg prompt
```

This command checks for problems in the file system of the virtual machine in the `app` directory. For each problem, the command prompts the user to confirm whether the problem should be fixed.

-v (--version)

You can use the `-v` option to find out the version number of the Oracle JRockit Virtual Edition Image Tool and an Oracle JRockit Virtual Edition binary.

Syntax

```
java -jar jrockitve-imagetool.jar -v [vm_cfg]
```

> **Note:** For information about the syntax conventions, see Documentation Conventions for Command-Line Syntax.

Arguments

Argument	Description
jrockitve_image	Path and name of the virtual machine configuration file (`vm.cfg`)

Example

```
java -jar jrockitve-imagetool.jar -v jrockitve.bin
```

This command displays the version numbers of Oracle JRockit Virtual Edition and associated components, as follows:

```
Oracle JRockit Virtual Edition ImageTool version: 5.0-128420

jrockitve.bin:
jrockitve.name=Oracle JRockit Virtual Edition
jrockitve.version=11.1.1.3.0-49-128710
jrockitve.kernel.name=JRockitVE Kernel
jrockitve.kernel.version=6.1.0.0-59-128379
jrockitve.utils.name=JRockitVE Utils
jrockitve.utils.version=3.0-66-128420
jrockitve.jvm.name=BEA JRockit(R)
jrockitve.jvm.version=R27.6.6-28_o-125824-1.6.0_17-20091214-2104-linux-ia32
```

B

Configuration File Element Reference

This appendix provides information about the elements of the XML file that specifies configuration settings for virtual machines created by using Oracle JRockit Virtual Edition.

The **root** element of the configuration XML file is `jrockitve-imagetool-config`, which specifies the following attributes:

- The version of the Image Tool to be used to create a virtual machine image
- The XML schema that the Image Tool uses for validating the elements in the configuration file

The following is the high-level element hierarchy in the configuration XML file. The mandatory elements are highlighted in bold text. For more information about an element and its attributes, click the element name in the hierarchy.

```
jrockitve-imagetool-config (mandatory)
    java-filesystem-imports (optional)
    jrockitve-binary-url (optional)
    jrockitve-config (mandatory)
        storage (mandatory)
            disks (mandatory)
            mounts (mandatory)
        services (optional)
        vm-name (mandatory)
        working-dir (optional)
        java-arguments (mandatory)
        network (mandatory)
            dns (optional)
            nics (mandatory)
            hostname (optional)
        locale-data (optional)
        console-log-path (optional)
```

Appendix C provides samples of brief and detailed configuration XML files that you can generate by using the Image Tool.

B.1 java-filesystem-imports (optional)

This element contains zero or more `copy` elements, which specify the locations from which the Image Tool (when it assembles the virtual machine image) should copy Java application files to the file system of the virtual machine image.

If you do not specify this element, the resulting virtual machine image will not start. However, you can `put` the Java application files into the virtual machine image later by using the -f (--file) command-line option.

Example

```
<jrockitve-filesystem-imports>
   <copy from="src/*" todir="/app"/>
   <copy from="src/app1" tofile="/app/app1.dat"/>
   <copy from="src/*"/>
</jrockitve-filesystem-imports>
```

B.1.1 copy (optional)

You can specify multiple **`copy`** elements, each of which specifies a local source directory and either a destination directory or destination file in the virtual machine image.

When you create the virtual machine image by using the -a (--assemble) option, the Image Tool copies, recursively, the files from each specified source directory to the corresponding destination.

Any links (including symbolic links) and path references are copied intact; so make sure that your Java application does not contain absolute paths and links to directories outside the directories specified by `java-filesystem-imports`.

Note: If you do not specify any **`copy`** element, the Image Tool copies, recursively, the contents of the directory named `root` (in the directory that contains the configuration file) to the root (`/`) directory in the file system of the virtual machine. If a directory named `root` does not exist in the directory that contains the configuration file, the Image Tool still builds the virtual machine image, but you cannot run the virtual machine until you add a **`copy`** element.

Attributes

Name	Description	Mandatory / Optional	Default Value
from	The directory or file in the local file system that should be copied to the virtual machine image. The path is relative to the XML configuration file.	Mandatory	
todir	The absolute path of the directory in the file system of the virtual machine image to which the files or directories specified in the `from` attribute should be copied.	Optional	The root directory (`/`)
tofile	The absolute path and name of the file (in the file system of the virtual machine image) to which the file specified in the `from` attribute should be copied.	Optional	The file specified in the `from` attribute is copied, with the same name, to the root directory (`/`)

Wild cards are supported for all the attributes.

Note: When specifying directory paths, use the appropriate path delimiters – slash (`/`) on Linux and on UNIX-like systems; backslash (`\`) on Windows – depending on the machine on which you are assembling the virtual machine image.

Examples

- `<copy from="src/*" todir="/app"/>`

  The Image Tool copies, recursively, all the directories and files from the `src` directory in the local file system to the `/app` directory in the file system of the virtual machine image.

- `<copy from="src/app1" tofile="/app/app1.dat"/>`

  The Image Tool copies the `app1` file located in the `src` directory in the local file system to the `/app/app1.dat` file in the file system of the virtual machine image.

- `<copy from="src/*"/>`

  The Image Tool copies, recursively, all the directories and files from the `src` directory in the local file system to the root (`/`) directory in the file system of the virtual machine image.

B.2 jrockitve-binary-url (optional)

This element specifies the location from which the Image Tool should download the Oracle JRockit Virtual Edition binary (`jrockitve.bin`). Typically, the value of this element would be the directory in which you installed Oracle JRockit Virtual Edition, but you can specify any location in the network or on the Internet.

Depending on the location of the `jrockitve.bin` file, you must specify the appropriate protocol prefix (`file:`, `http://`, `ftp://`, and so on).

If you do not specify this element, when the Image Tool assembles the application, it looks for the `jrockitve.bin` file in the directory in which the configuration XML file resides.

Example

```
<jrockitve-binary-url>file:../jrockitve.bin</jrockitve-binary-url>
```

This example specifies that the `jrockitve.bin` file is in the parent directory of the directory that contains the configuration XML file.

B.3 jrockitve-config (mandatory)

This element and its child elements define the configuration settings of the virtual machine.

Attributes

Name	Description	Mandatory / Optional
memory	Maximum memory available to the virtual machine.	Mandatory
cpus	Number of CPUs available to the virtual machine. You can configure a maximum of eight virtual CPUs. **Note**: In the current release, Oracle recommends that you configure a single CPU, because configuring the virtual machine for multiple virtual CPUs affects performance.	Mandatory

You can specify the following units (**not case sensitive**) for the **`memory`** attribute:

- K or KB: Kilobytes
- M or MB: Megabytes
- G or GB: Gigabytes
- T or TB: Terabytes

If you do not specify the unit, the Image Tool assumes that the values are specified in megabytes.

Example

```
<jrockitve-config memory="256 MB" cpus=2>
```

Child Elements

- storage (mandatory)
- services (optional)
- vm-name (mandatory)
- working-dir (optional)
- java-arguments (mandatory)
- network (mandatory)
- locale-data (optional)
- console-log-path (optional)

B.3.1 storage (mandatory)

The `storage` element is a container for configurations of disks and mounts.

Child Elements

- disks (mandatory)
- mounts (mandatory)

Example

```
<storage>
   <disks>
      <disk id="root" size="512 MB"/>
    </disks>
    <mounts>
       <mount>
          <mount-point>/</mount-point>
          <disk>root</disk>
       </mount>
       <mount type="nfs">
          <mount-point>/test1</mount-point>
          <server>nfs.mycompany.com</server>
          <server-path>/export/test1</server-path>
          <options>
             <option>uid=513</option>
             <option>gid=503</option>
          </options>
       </mount>
    </mounts>
</storage>
```

B.3.1.1 disks (mandatory)

This element is a container for the **`disk`** element.

B.3.1.1.1 disk (mandatory)

This element defines the properties of the root disk.

If you change the disk size, the virtual machine image is resized (shrunk or expanded) to the new size specified. If the image cannot be expanded to the specified size due to insufficient disk space, an error message is displayed.

If the Java application to be converted into a virtual machine image contains more data than the disk size specified, when the Image Tool assembles the virtual machine image, it overrides (but does not change) the disk size specified in the configuration XML file and proceeds with the assembly.

Attributes

Name	Description	Mandatory / Optional
id	A string that identifies the disk	Mandatory
size	An integer value that specifies the size of the disk	Mandatory

You can specify the following units (**not case sensitive**) for the **`size`** attribute:

- `K` or `KB`: Kilobytes
- `M` or `MB`: Megabytes
- `G` or `GB`: Gigabytes
- `T` or `TB`: Terabytes

If you do not specify the unit, the Image Tool assumes that the values are specified in megabytes.

Example

```
<disk id="root" size="512 MB"/>
```

B.3.1.2 mounts (mandatory)

This element is a container for mounts defined by the `mount` element.

B.3.1.2.1 mount (mandatory)

This element defines the settings for mount points.

Attribute

Name	Description	Mandatory / Optional	Default Value
type	The mount type: `nfs` or `native`	Optional	native

The configuration file should contain one (and only one) `native` mount that maps the native disk defined for the virtual machine to the root (`/`) directory of the virtual machine.

There can be any number of `nfs` mounts.

Child Elements

- **mount-point (mandatory)**

  This element specifies the absolute path in the file system of the virtual machine to which the native disk and the NFS locations defined for the virtual machine should be mapped.

  Note: You must specify the path in the POSIX format by using slashes (**/**) as delimiters.

  A slash (**/**) at the beginning of the path represents the root directory. A period (**.**) at the beginning of the path represents the current directory. Two periods (**..**) indicate the parent directory.

- **disk (mandatory)**

  This element is mandatory (and allowed) only for mounts of type `native`. It allows you to map the native disk defined for the virtual machine to the root (`/`) directory of the virtual machine.

  The value of this element is the `id` specified for the native `disk` within the `disks` element.

- **server (mandatory)**

  This element is mandatory (and allowed) only for mounts of type `nfs`. It specifies the server that contains the NFS location. It is a string representing a server name or an IP address.

- **server-path (optional)**

  This element is mandatory (and allowed) only for mounts of type `nfs`. It specifies the directory path of the NFS location.

- **options (optional)**

  This element is allowed only for mounts of type `nfs`. It contains one or more **`option`** elements that specify configuration settings for NFS mount points: `uid`, `gid`, and so on.

Example

```
<mounts>
   <mount type="native">
      <mount-point>/</mount-point>
      <disk>root</disk>
   </mount>
   <mount type="nfs">
      <mount-point>/test1</mount-point>
      <server>nfs.mycompany.com</server>
      <server-path>/export/test1</server-path>
      <options>
         <option>uid=513</option>
         <option>gid=503</option>
      </options>
   </mount>
</mounts>
```

B.3.2 services (optional)

This element is a container for one or more `service` elements, which specify the properties of the services that are enabled for the virtual machine.

Example

```
<services>
   <service name="sshd"/>
      <arguments>
         <argument>port=4711</argument>
         <argument>no_auth</argument>
      </arguments>
   </service>
</services>
```

B.3.2.1 service (optional)

This element contains specifies the name of the enabled service, and is a container for the `arguments` element.

B.3.2.2 arguments (optional)

This element is a container for one or more `argument` elements, which specify service-specific arguments.

You can find out what service arguments can be configured for a service by using the --reconfigure-service command-line option of the Image Tool. For example, to find out the service arguments that can be configured for the `sshd` service, run the following command:

```
java -jar jrockitve-imagetool.jar --reconfigure-service vm_cfg sshd get
valid-arguments
```

This command might produce output that resembles the following:

```
port=<#>
log=<quiet|brief|verbose|debug>
no_auth
```

B.3.3 vm-name (mandatory)

This element specifies the name of the virtual machine as it would appear to Oracle VM. The name can contain only the following characters: A–Z, a–z, 0–9, period (.), hyphen (-), and underscore (_).

If you do not specify this element, when the Image Tool assembles the virtual machine image, it uses the name `default-vm`.

Example

```
<vm-name>my_vm</vm-name>
```

B.3.4 working-dir (optional)

This element specifies the directory in the file system inside a virtual machine image that should serve as the working directory for the JRockit JVM (the default location in which crash files, for example, are created).

Note: You must specify the path in the POSIX format by using slashes (/) as delimiters.

The default working directory is the root (/) directory in the file system of the virtual machine.

B.3.5 java-arguments (mandatory)

This element specifies the Java arguments for the virtual machine and the Java application.

Note: You must specify the path in the POSIX format by using slashes (/) as delimiters.

A slash (/) at the beginning of the path represents the root directory of the virtual machine. A period (.) at the beginning of the path represents the current directory. Two periods (..) indicate the parent directory.

To specify the Java home directory of the JDK in the file system inside a virtual machine image (`/jrockitve/jrockit`) as part of a Java argument, you can use the `$JAVA_HOME` variable. For example, in an argument that points to the `/jrockitve/jrockit/lib/tools.jar` file, you can specify the path as `$JAVA_HOME/lib/tools.jar`.

Example

```
<java-arguments>-cp $JAVA_HOME/lib/tools.jar:. -Xmx128m -jar App.jar arg0
</java-arguments>
```

B.3.6 network (mandatory)

This element specifies the configuration for network interface cards (NICs) and DNS servers.

Child Elements

- dns (optional)
- nics (mandatory)
- hostname (optional)

Example: Detailed Network Configuration

```
<network>
   <dns>
      <static-hosts>
         <hosts ip="192.168.1.34">
            <host>test</host>
         </hosts>
         <hosts ip="10.0.1.56">
            <host>thefiftysix.mycompany.com</host>
            <host>the56.mycompany.com</host>
         </hosts>
      </static-hosts>
      <server-order>
```

```
         <server ip="172.22.17.100"/>
         <server ip="192.168.1.2"/>
      </server-order>
      <lookup-order>
         <name suffix="us.mycompany.com"/>
         <name suffix="in.mycompany.com"/>
      </lookup-order>
   </dns>
   <nics>
      <nic network="eth0" type="nat">
         <ip>172.23.22.22</ip>
         <netmask>255.255.255.0</netmask>
         <gateway>172.23.22.1</gateway>
         <mac>12:ab:34:cd:56:ef</mac>
      </nic>
   </nics>
   <hostname>example</hostname>
</network>
```

Example: Minimum Network Configuration

The minimum network configuration example specifies a single network interface card (NIC) that uses DHCP over a bridged network.

```
<network>
   <nics>
      <nic/>
   </nics>
</network>
```

B.3.6.1 dns (optional)

This element is a container for DNS information in the configuration file. It should be used when the DNS information needs to be passed to Oracle JRockit Virtual Edition.

> **Note:** This element is mandatory only if you do not want to use DHCP for DNS configuration.

Child Elements

- **static-hosts (optional)**

  This element allows you to add static DNS entries (similar to the `/etc/hosts` file on UNIX-like systems, for example).

  It contains a list of host names specified by one or more `hosts` elements. More than one host name can be mapped to one IP address that is defined by the `ip` attribute of the `hosts` element.

- **server-order (optional)**

  This element contains a list of DNS server IP addresses specified by one or more `server` elements. The order of the IP addresses in the list indicates the lookup order.

- **lookup-order (optional)**

  This element contains a list of DNS server names specified by one or more `name` elements. The order of the DNS server names in the list indicates the lookup order.

Note: The DNS domain names and servers are tried in the order in which they are defined in the lookup table. Make sure that the primary domain names and primary servers are defined first.

Example

```
<dns>
   <static-hosts>
      <hosts ip="192.168.1.34">
         <host>test</host>
      </hosts>
      <hosts ip="10.0.1.56">
         <host>us.mycompany.com</host>
         <host>in.mycompany.com</host>
      </hosts>
   </static-hosts>
   <server-order>
      <server ip="172.22.17.100"/>
      <server ip="192.168.1.2"/>
   </server-order>
   <lookup-order>
      <name suffix="abc.com"/>
      <name suffix="xyz.com"/>
   </lookup-order>
</dns>
```

B.3.6.2 nics (mandatory)

This element is a container for the `nic` element.

Example

```
<nics>
   <nic network="eth0" type="bridged">
      <ip>172.123.122.122</ip>
      <netmask>255.255.255.0</netmask>
      <gateway>172.123.122.1</gateway>
      <mac>12:ab:34:cd:56:ef</mac>
    </nic>
</nics>
```

B.3.6.2.1 nic (mandatory)

This element specifies the configuration of the network interface card (NIC) available for the virtual machine.

Attributes	Description	Mandatory / Optional	Default Value
network	Name to identify the NIC	Optional	NA
type	Mode that the NIC must use to connect to the physical network from the virtual machine Possible values: ■ `bridged`: Connect to the physical network through the physical NIC on the machine in which the virtual machine is running ■ `nat`: Network address translation mode	Optional	bridged

Child Elements

- **ip (optional)**

  This element specifies the IP address of the NIC. It is mandatory only for static IP configuration.

  > **Note:** You can specify a static IP address, and leave `netmask` and `gateway` undefined (that is, to be determined by using DHCP).

- **netmask (optional)**

  This element specifies the netmask address (in dot-separated decimal octet format) of the NIC. It is mandatory only if you do not want to use DHCP for netmask configuration.

- **gateway (optional)**

  This element specifies the gateway address (in dot-separated decimal octet format) of the NIC. It is mandatory only if you do not want to use DHCP for gateway configuration.

- **mac (optional)**

  This element specifies the MAC address (in colon-separated hexadecimal octet format) of the NIC. If you do not specify this element, Oracle VM assigns a MAC address.

Example: Detailed Configuration

```
<nic network="eth0" type="bridged">
    <ip>172.123.122.122</ip>
    <netmask>255.255.255.0</netmask>
    <gateway>172.123.122.1</gateway>
    <mac>12:ab:34:cd:56:ef</mac>
</nic>
```

Example: Minimum Configuration

```
</nic>
```

The minimum configuration example specifies a single NIC (without an identifying name) that uses DHCP over a bridged network.

> **Note:** In the current release, only one NIC is supported.

B.3.6.3 hostname (optional)

This element specifies the host name of the virtual machine.

Example

```
<hostname>example</hostname>
```

B.3.7 locale-data (optional)

This element specifies the locale information (language, time zone, and character encoding) for the virtual machine. If you do not specify this element, the virtual machine uses the locale settings of the machine on which it was created (assembled).

Child Elements

- **locale** (optional)

  This element specifies the name of the locale.

- **timezone** (optional)

  This element specifies the time zone.

- **encoding** (optional)

  This element specifies the character encoding to be used for the locale.

Example

```
<locale-data>
   <locale>en_UK</locale>
   <timezone>Europe/London</timezone>
   <encoding>ISO-8859-1</encoding>
</locale-data>
```

B.3.8 console-log-path (optional)

This element specifies the directory path and the name of the file in which the virtual console log messages should be stored. The console log contains the output of the Java application and Oracle JRockit Virtual Edition.

The log file location specified by using this element might contain several files as a result of log-file rotation. Oracle JRockit Virtual Edition rotates log files as follows: Every time the virtual machine starts, the existing contents of the main log file are moved to *log_file*`.old.1`, the contents of *log_file*`.old.1` are moved to *log_file*`.old.2`, and so on, up to *log_file*`.old.9`.

By default, virtual console log messages are written to `/jrockitve/log/jrockitve.log` in the file system of the virtual machine. Change the log location only when absolutely essential to use a different location.

Note: You must specify the path in the POSIX format by using slashes (**/**) as delimiters.

A slash (**/**) at the beginning of the path represents the root directory. A period (**.**) at the beginning of the path represents the current directory. Two periods (**..**) indicate the parent directory.

Example

```
<console-log-path>/test/mylog.log</console-log-path>
```

C

Sample Configuration Files

This appendix provides the contents of the following sample configuration files, which you can generate by using the Oracle JRockit Virtual Edition Image Tool.

- Sample Brief Configuration File
- Sample Detailed Configuration File

C.1 Sample Brief Configuration File

This section shows the contents of the sample brief configuration file, which you can generate by using the **-c (--create-config)** option of the Image Tool.

```
<jrockitve-imagetool-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="jrockitve-imagetool-config.xsd" version="5.0">
  <jrockitve-config memory="256 MB" cpus="1">
    <storage>
      <disks>
        <disk id="root" size="256 MB"/>
      </disks>
      <mounts>
        <mount>
          <mount-point>/</mount-point>
          <disk>root</disk>
        </mount>
      </mounts>
    </storage>
    <vm-name>default-vm</vm-name>
    <java-arguments>HelloWorld</java-arguments>
    <network>
      <nics>
        <nic type="bridged"/>
      </nics>
    </network>
  </jrockitve-config>
</jrockitve-imagetool-config>
```

C.2 Sample Detailed Configuration File

This section shows the contents of the sample detailed configuration file, which you can generate by using the **--create-full-config** option of the Image Tool.

```
<jrockitve-imagetool-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="jrockitve-imagetool-config.xsd" version="5.0">
  <jrockitve-filesystem-imports>
    <copy from="/installs/*install3*/*"/>
```

```
    <copy from="/share/temp/data/*" todir="/data"/>
    <copy from="/tmp/my.dat" tofile="/data/THEVERSION"/>
  </jrockitve-filesystem-imports>
  <jrockitve-config memory="1 GB" cpus="1">
    <storage>
      <disks>
        <disk id="root" size="256 MB"/>
      </disks>
      <mounts>
        <mount>
          <mount-point>/</mount-point>
          <disk>root</disk>
        </mount>
        <mount type="nfs">
          <mount-point>/test</mount-point>
          <options>
            <option>uid=513</option>
            <option>gid=503</option>
          </options>
          <server-path>/temp/user/testdir</server-path>
          <server>us.mycompany.com</server>
        </mount>
      </mounts>
    </storage>
    <vm-name>default-vm</vm-name>
    <working-dir>/app/myworkdir/</working-dir>
    <java-arguments>-cp $JAVA_HOME/lib/tools.jar:. -Xmx128m -jar MyApp.jar arg0
arg1 arg2</java-arguments>
    <kernel-arguments>
      <entry key="logLog" value="all"/>
      <entry key="netTcpTtl" value="20"/>
    </kernel-arguments>
    <console-log-path>/test/mylog.log</console-log-path>
    <network>
      <dns>
        <static-hosts>
          <hosts ip="192.168.1.34">
            <host>myhost</host>
          </hosts>
          <hosts ip="10.0.1.56">
            <host>one.myhost1.com</host>
            <host>two.myhost1.com</host>
          </hosts>
        </static-hosts>
        <server-order>
          <server ip="172.22.17.100"/>
          <server ip="192.168.1.2"/>
        </server-order>
        <lookup-order>
          <name suffix="us.host.com"/>
          <name suffix="in.host.com"/>
        </lookup-order>
      </dns>
      <nics>
        <nic network="eth0" type="nat">
          <ip>172.23.22.22</ip>
          <netmask>255.255.255.0</netmask>
          <mac>12:ab:34:cd:56:ef</mac>
        </nic>
      </nics>
```

```
      <hostname>example</hostname>
    </network>
    <locale-data>
      <locale>en_UK</locale>
      <timezone>Europe/London</timezone>
      <encoding>ISO-8859-1</encoding>
    </locale-data>
  </jrockitve-config>
  <jrockitve-binary-url>ftp://host.com/jrockitve.bin</jrockitve-binary-url>
</jrockitve-imagetool-config>
```

D

Known Issues

This appendix describes the following known issues in Oracle JRockit Virtual Edition 11.1.1.3.0, and provides workarounds (if available) for the issues.

D.1 Cannot Specify Multiple NICs

Oracle JRockit Virtual Edition currently supports only one network interface card (NIC) for virtual machines.

For information about how to configure the NIC for a virtual machine, see the description of the `network` configuration element in Appendix B, "Configuration File Element Reference."

D.2 Image Tool Cannot Handle Windows Short Path Names

The Image Tool returns an error if a parameter that you specify contains a Windows short path name.

For example, when you run the **`-f`** option to copy a file from the local (Windows) file system to the file system of a virtual machine image, if the path specified for the local file system contains a short name, an error message is displayed.

Example

```
java -jar jrockitve-imagetool.jar -f vm.cfg put "c:\Docume~1\prop.txt" /app/
```

This command returns the following error:

```
No match for "c:\Docume~1\prop.txt"
```

Workaround

Use long path names, as shown in the following example.

```
java -jar jrockitve-imagetool.jar -f vm.cfg put "C:\Documents and
Settings\prop.txt" /app/
```

D.3 Suboptimal Disk I/O Performance in Certain Cases

Oracle JRockit Virtual Edition, like all operating systems, maintains an in-memory disk cache to improve disk I/O performance. Under certain conditions, this in-memory cache has to be flushed to guarantee a consistent disk state. The current release (11.1.1.3.0) of Oracle JRockit Virtual Edition flushes the in-memory disk cache in some cases where it is not strictly necessary to do so. This can result in suboptimal performance in some cases — for example, deletion of a large number of files.

D.4 Performance Loss When Using Multiple Virtual CPUs

In a virtual machine that is configured with multiple virtual CPUs, in certain circumstances (for example, when there is a lot of synchronization between threads), performance might be affected.

In the current release, Oracle recommends that you configure a single virtual CPU. Avoid configuring the virtual machine with more CPUs than are essential for the application, because each extra CPU causes additional overhead, with the largest increase in overhead occurring when moving from one CPU to two CPUs.

D.5 File Locking Not Available for NFS Access

While the virtual image reads a file from an NFS shared storage location another process could write to the same file, leading to problems. File locking is not available in the current release.

D.6 Patched Virtual Machine Might Not Work in Certain Cases

After using the -p (--patch) option, the virtual machine will not work if the kernel or a service in the new Oracle JRockit Virtual Edition binary contains arguments or command parameters that are different from those used in the original virtual machine.

To solve this problem, do the following:

1. Disassemble the virtual machine image by using the -d (--disassemble) option of the Image Tool.
2. Make sure that the `<jrockitve-binary-url>` element in the configuration file specifies the correct location of the new Oracle JRockit Virtual Edition binary (`jrockitve.bin`).
3. Assemble the virtual machine image afresh by using the -a (--assemble) option of the Image Tool.

For more information, see Section 4.1.3, "Disassembling and Re-Creating Virtual Machine Images."