Oracle® VM VirtualBox

User Manual for Release 6.0

ORACLE®

Table of Contents

Preface

The *Oracle VM VirtualBox User Manual* provides an introduction to using Oracle VM VirtualBox. The manual provides information on how to install Oracle VM VirtualBox and use it to create and configure virtual machines.

Audience

This document is intended for both new and existing users of Oracle VM VirtualBox. It is assumed that readers are familiar with Web technologies and have a general understanding of Windows and UNIX platforms.

Related Documents

The documentation for this product is available at:

https://docs.oracle.com/en/virtualization/virtualbox/index.html

Conventions

The following text conventions are used in this document:

- **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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Chapter 1 First Steps

Welcome to Oracle VM VirtualBox.

Oracle VM VirtualBox is a cross-platform virtualization application. What does that mean? For one thing, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac OS X, Linux, or Oracle Solaris operating systems (OSes). Secondly, it extends the capabilities of your existing computer so that it can run multiple OSes, inside multiple virtual machines, at the same time. As an example, you can run Windows and Linux on your Mac, run Windows Server 2016 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. You can install and run as many virtual machines as you like. The only practical limits are disk space and memory.

Oracle VM VirtualBox is deceptively simple yet also very powerful. It can run everywhere from small embedded systems or desktop class machines all the way up to datacenter deployments and even Cloud environments.

The following screenshot shows how Oracle VM VirtualBox, installed on an Apple Mac OS X computer, is running Windows Server 2016 in a virtual machine window.

Figure 1.1 Windows Server 2016 Virtual Machine, Displayed on a Mac OS X Host
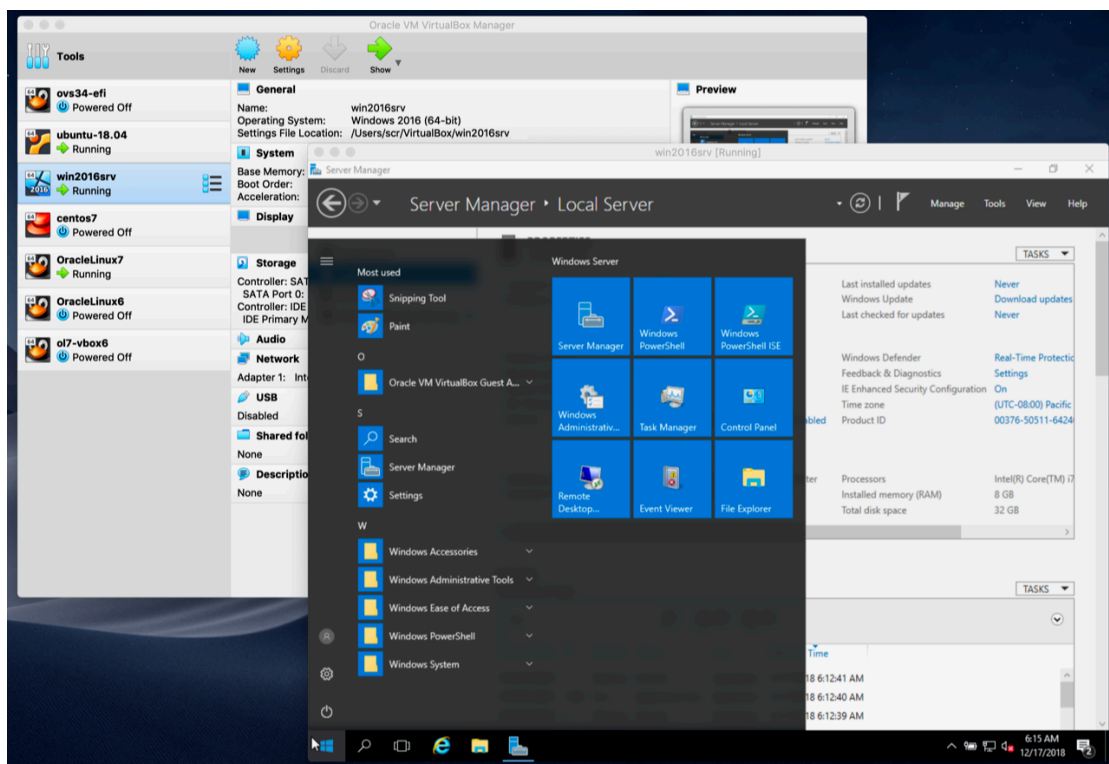
In this User Manual, we will begin simply with a quick introduction to virtualization and how to get your first virtual machine running with the easy-to-use Oracle VM VirtualBox graphical user interface. Subsequent chapters will go into much more detail covering more powerful tools and features, but fortunately, it is not necessary to read the entire User Manual before you can use Oracle VM VirtualBox.

You can find a summary of Oracle VM VirtualBox's capabilities in Section 1.3, "Features Overview". For existing Oracle VM VirtualBox users who just want to find out what is new in this release, see the Change Log.

1.1. Why is Virtualization Useful?

The techniques and features that Oracle VM VirtualBox provides are useful in the following scenarios:

- **Running multiple operating systems simultaneously.** Oracle VM VirtualBox enables you to run more than one OS at a time. This way, you can run software written for one OS on another, such as Windows software on Linux or a Mac, without having to reboot to use it. Since you can configure what kinds of *virtual* hardware should be presented to each such OS, you can install an old OS such as DOS or OS/2 even if your real computer's hardware is no longer supported by that OS.

- **Easier software installations.** Software vendors can use virtual machines to ship entire software configurations. For example, installing a complete mail server solution on a real machine can be a tedious task. With Oracle VM VirtualBox, such a complex setup, often called an *appliance*, can be packed into a virtual machine. Installing and running a mail server becomes as easy as importing such an appliance into Oracle VM VirtualBox.

- **Testing and disaster recovery.** Once installed, a virtual machine and its virtual hard disks can be considered a *container* that can be arbitrarily frozen, woken up, copied, backed up, and transported between hosts.

  On top of that, with the use of another Oracle VM VirtualBox feature called *snapshots*, one can save a particular state of a virtual machine and revert back to that state, if necessary. This way, one can freely experiment with a computing environment. If something goes wrong, such as prolems after installing software or infecting the guest with a virus, you can easily switch back to a previous snapshot and avoid the need of frequent backups and restores.

  Any number of snapshots can be created, allowing you to travel back and forward in virtual machine time. You can delete snapshots while a VM is running to reclaim disk space.

- **Infrastructure consolidation.** Virtualization can significantly reduce hardware and electricity costs. Most of the time, computers today only use a fraction of their potential power and run with low average system loads. A lot of hardware resources as well as electricity is thereby wasted. So, instead of running many such physical computers that are only partially used, one can pack many virtual machines onto a few powerful hosts and balance the loads between them.

1.2. Some Terminology

When dealing with virtualization, and also for understanding the following chapters of this documentation, it helps to acquaint oneself with a bit of crucial terminology, especially the following terms:

- **Host operating system (host OS).** This is the OS of the physical computer on which Oracle VM VirtualBox was installed. There are versions of Oracle VM VirtualBox for Windows, Mac OS X, Linux, and Oracle Solaris hosts. See Section 1.4, "Supported Host Operating Systems".

  Most of the time, this manual discusses all Oracle VM VirtualBox versions together. There may be platform-specific differences which we will point out where appropriate.

- **Guest operating system (guest OS).** This is the OS that is running inside the virtual machine. Theoretically, Oracle VM VirtualBox can run any x86 OS such as DOS, Windows, OS/2, FreeBSD, and OpenBSD. But to achieve near-native performance of the guest code on your machine, we had to go through a lot of optimizations that are specific to certain OSes. So while your favorite OS *may* run as a guest, we officially support and optimize for a select few, which include the most common OSes.

  See Section 3.1, "Supported Guest Operating Systems".

- **Virtual machine (VM).** This is the special environment that Oracle VM VirtualBox creates for your guest OS while it is running. In other words, you run your guest OS *in* a VM. Normally, a VM is shown as a window on your computer's desktop. Depending on which of the various frontends of Oracle VM VirtualBox you use, the VM might be shown in full screen mode or remotely on another computer.

  Internally, Oracle VM VirtualBox treats a VM as a set of parameters that specify its behavior. Some parameters describe hardware settings, such as the amount of memory and number of CPUs assigned. Other parameters describe the state information, such as whether the VM is running or saved.

  You can view these VM settings in the VirtualBox Manager window, the **Settings** dialog, and by running the `VBoxManage` command. See Chapter 7, *VBoxManage*.

- **Guest Additions.** This refers to special software packages which are shipped with Oracle VM VirtualBox but designed to be installed *inside* a VM to improve performance of the guest OS and to add extra features. See Chapter 4, *Guest Additions*.

1.3. Features Overview

The following is a brief outline of Oracle VM VirtualBox's main features:

- **Portability.** Oracle VM VirtualBox runs on a large number of 64-bit host OSes. See Section 1.4, "Supported Host Operating Systems".

  Oracle VM VirtualBox is a so-called *hosted* hypervisor, sometimes referred to as a *type 2* hypervisor. Whereas a *bare-metal* or *type 1* hypervisor would run directly on the hardware, Oracle VM VirtualBox requires an existing OS to be installed. It can thus run alongside existing applications on that host.

  To a very large degree, Oracle VM VirtualBox is functionally identical on all of the host platforms, and the same file and image formats are used. This enables you to run virtual machines created on one host on another host with a different host OS. For example, you can create a virtual machine on Windows and then run it under Linux.

  In addition, virtual machines can easily be imported and exported using the Open Virtualization Format (OVF), an industry standard created for this purpose. You can even import OVFs that were created with a different virtualization software. See Section 1.14, "Importing and Exporting Virtual Machines".

- **No hardware virtualization required.** For many scenarios, Oracle VM VirtualBox does not require the processor features built into current hardware, such as Intel VT-x or AMD-V. As opposed to many other virtualization solutions, you can therefore use Oracle VM VirtualBox even on older hardware where these features are not present. See Hardware vs. Software Virtualization.

- **Guest Additions: shared folders, seamless windows, 3D virtualization.** The Oracle VM VirtualBox Guest Additions are software packages which can be installed *inside* of supported guest systems to improve their performance and to provide additional integration and communication with the host system. After installing the Guest Additions, a virtual machine will support automatic adjustment of video resolutions, seamless windows, accelerated 3D graphics and more. See Chapter 4, *Guest Additions*.

  In particular, Guest Additions provide for *shared folders*, which let you access files on the host system from within a guest machine. See Section 4.3, "Shared Folders".

- **Great hardware support.** Among other features, Oracle VM VirtualBox supports the following:

  - **Guest multiprocessing (SMP).** Oracle VM VirtualBox can present up to 32 virtual CPUs to each virtual machine, irrespective of how many CPU cores are physically present on your host.

  - **USB device support.** Oracle VM VirtualBox implements a virtual USB controller and enables you to connect arbitrary USB devices to your virtual machines without having to install device-specific

drivers on the host. USB support is not limited to certain device categories. See Section 3.11.1, "USB Settings".

- **Hardware compatibility.** Oracle VM VirtualBox virtualizes a vast array of virtual devices, among them many devices that are typically provided by other virtualization platforms. That includes IDE, SCSI, and SATA hard disk controllers, several virtual network cards and sound cards, virtual serial and parallel ports and an Input/Output Advanced Programmable Interrupt Controller (I/O APIC), which is found in many computer systems. This enables easy cloning of disk images from real machines and importing of third-party virtual machines into Oracle VM VirtualBox.

- **Full ACPI support.** The Advanced Configuration and Power Interface (ACPI) is fully supported by Oracle VM VirtualBox. This enables easy cloning of disk images from real machines or third-party virtual machines into Oracle VM VirtualBox. With its unique *ACPI power status support*, Oracle VM VirtualBox can even report to ACPI-aware guest OSes the power status of the host. For mobile systems running on battery, the guest can thus enable energy saving and notify the user of the remaining power, for example in full screen modes.

- **Multiscreen resolutions.** Oracle VM VirtualBox virtual machines support screen resolutions many times that of a physical screen, allowing them to be spread over a large number of screens attached to the host system.

- **Built-in iSCSI support.** This unique feature enables you to connect a virtual machine directly to an iSCSI storage server without going through the host system. The VM accesses the iSCSI target directly without the extra overhead that is required for virtualizing hard disks in container files. See Section 5.10, "iSCSI Servers".

- **PXE Network boot.** The integrated virtual network cards of Oracle VM VirtualBox fully support remote booting using the Preboot Execution Environment (PXE).

- **Multigeneration branched snapshots.** Oracle VM VirtualBox can save arbitrary snapshots of the state of the virtual machine. You can go back in time and revert the virtual machine to any such snapshot and start an alternative VM configuration from there, effectively creating a whole snapshot tree. See Section 1.10, "Snapshots". You can create and delete snapshots while the virtual machine is running.

- **VM groups.** Oracle VM VirtualBox provides a groups feature that enables the user to organize and control virtual machines collectively, as well as individually. In addition to basic groups, it is also possible for any VM to be in more than one group, and for groups to be nested in a hierarchy. This means you can have groups of groups. In general, the operations that can be performed on groups are the same as those that can be applied to individual VMs: Start, Pause, Reset, Close (Save state, Send Shutdown, Poweroff), Discard Saved State, Show in File System, Sort.

- **Clean architecture and unprecedented modularity.** Oracle VM VirtualBox has an extremely modular design with well-defined internal programming interfaces and a clean separation of client and server code. This makes it easy to control it from several interfaces at once. For example, you can start a VM simply by clicking on a button in the Oracle VM VirtualBox graphical user interface and then control that machine from the command line, or even remotely. See Section 1.16, "Alternative Front-Ends".

  Due to its modular architecture, Oracle VM VirtualBox can also expose its full functionality and configurability through a comprehensive **software development kit (SDK),** which enables integration of Oracle VM VirtualBox with other software systems. See Oracle VM VirtualBox Programming Interfaces.

- **Remote machine display.** The VirtualBox Remote Desktop Extension (VRDE) enables high-performance remote access to any running virtual machine. This extension supports the Remote

Desktop Protocol (RDP) originally built into Microsoft Windows, with special additions for full client USB support.

The VRDE does not rely on the RDP server that is built into Microsoft Windows. Instead, the VRDE is plugged directly into the virtualization layer. As a result, it works with guest OSes other than Windows, even in text mode, and does not require application support in the virtual machine either. The VRDE is described in detail in Remote Display (VRDP Support).

On top of this special capacity, Oracle VM VirtualBox offers you more unique features:

- **Extensible RDP authentication.** Oracle VM VirtualBox already supports Winlogon on Windows and PAM on Linux for RDP authentication. In addition, it includes an easy-to-use SDK which enables you to create arbitrary interfaces for other methods of authentication. See RDP Authentication.

- **USB over RDP.** Using RDP virtual channel support, Oracle VM VirtualBox also enables you to connect arbitrary USB devices locally to a virtual machine which is running remotely on an Oracle VM VirtualBox RDP server. See Remote USB.

1.4. Supported Host Operating Systems

Currently, Oracle VM VirtualBox runs on the following host OSes:

- **Windows hosts (64-bit):**

  - Windows 7

  - Windows 8

  - Windows 8.1

  - Windows 10 RTM (1507) build 10240

  - Windows 10 November Update (1511) build 10586

  - Windows 10 Anniversary Update (1607) build 14393

  - Windows 10 Creators Update (1703) build 15063

  - Windows 10 Fall Creators Update (1709) build 16299

  - Windows 10 April 2018 Update (1803) build 17134

  - Windows 10 October 2018 Update (1809) build 17763

  - Windows Server 2008 R2

  - Windows Server 2012

  - Windows Server 2012 R2

  - Windows Server 2016

  - Windows Server 2019

- **Mac OS X hosts (64-bit):**

  - 10.12 (Sierra)

- 10.13 (High Sierra)

- 10.14 (Mojave)

Intel hardware is required. See also Known Limitations.

- **Linux hosts (64-bit).** Includes the following:

  - Ubuntu 16.04 LTS, 18.04 LTS and 18.10

  - Debian GNU/Linux 9 ("Stretch")

  - Oracle Linux 6 and 7

  - Red Hat Enterprise Linux 6 and 7

  - Fedora 28 and 29

  - Gentoo Linux

  - SUSE Linux Enterprise server 12 and 15

  - openSUSE Leap 42.3 and 15.0

  It should be possible to use Oracle VM VirtualBox on most systems based on Linux kernel 2.6 or 3.x using either the Oracle VM VirtualBox installer or by doing a manual installation. See Section 2.3, "Installing on Linux Hosts". However, the formally tested and supported Linux distributions are those for which we offer a dedicated package.

  Note that Linux 2.4-based host OSes are no longer supported.

- **Oracle Solaris hosts (64-bit only).** The following versions are supported with the restrictions listed in Known Limitations:

  - Oracle Solaris 11

Note that the above list is informal. Oracle support for customers who have a support contract is limited to a subset of the listed host OSes. Also, any feature which is marked as *experimental* is not supported. Feedback and suggestions about such features are welcome.

1.4.1. Host CPU Requirements

SSE2 (Streaming SIMD Extensions 2) support is required for host CPUs.

1.5. Installing Oracle VM VirtualBox and Extension Packs

Oracle VM VirtualBox comes in many different packages, and installation depends on your host OS. If you have installed software before, installation should be straightforward. On each host platform, Oracle VM VirtualBox uses the installation method that is most common and easy to use. If you run into trouble or have special requirements, see Chapter 2, *Installation Details* for details about the various installation methods.

Oracle VM VirtualBox is split into the following components:

- **Base package.** The base package consists of all open source components and is licensed under the GNU General Public License V2.

- **Extension packs.** Additional extension packs can be downloaded which extend the functionality of the Oracle VM VirtualBox base package. Currently, Oracle provides a single extension pack, available from: http://www.virtualbox.org. The extension pack provides the following added functionality:

  1. The virtual USB 2.0 (EHCI) device. See Section 3.11.1, "USB Settings".

  2. The virtual USB 3.0 (xHCI) device. See Section 3.11.1, "USB Settings".

  3. VirtualBox Remote Desktop Protocol (VRDP) support. See Remote Display (VRDP Support).

  4. Host webcam passthrough. See Webcam Passthrough.

  5. Intel PXE boot ROM.

  6. Experimental support for PCI passthrough on Linux hosts. See PCI Passthrough.

  7. Disk image encryption with AES algorithm. See Encryption of Disk Images.

  Oracle VM VirtualBox extension packages have a `.vbox-extpack` file name extension. To install an extension, simply double-click on the package file and a **Network Operations Manager** window is shown to guide you through the required steps.

  To view the extension packs that are currently installed, start the VirtualBox Manager, as shown in Section 1.6, "Starting Oracle VM VirtualBox". From the **File** menu, select **Preferences**. In the window that displays, go to the **Extensions** category. This shows you the extensions which are currently installed, and enables you to remove a package or add a new package.

  Alternatively, you can use the `VBoxManage` command line. See Section 7.41, "VBoxManage extpack".

1.6. Starting Oracle VM VirtualBox

After installation, you can start Oracle VM VirtualBox as follows:

- On a Windows host, in the **Programs** menu, click on the item in the **VirtualBox** group. On some Windows platforms, you can also enter `VirtualBox` in the search box of the **Start** menu.

- On a Mac OS X host, in the Finder, double-click on the **VirtualBox** item in the Applications folder. You may want to drag this item onto your Dock.

- On a Linux or Oracle Solaris host, depending on your desktop environment, an Oracle VM VirtualBox item may have been placed in either the System or System Tools group of your **Applications** menu. Alternatively, you can enter `VirtualBox` in a terminal window.

When you start Oracle VM VirtualBox for the first time, a window like the following is displayed:

Figure 1.2 VirtualBox Manager Window, After Initial Startup
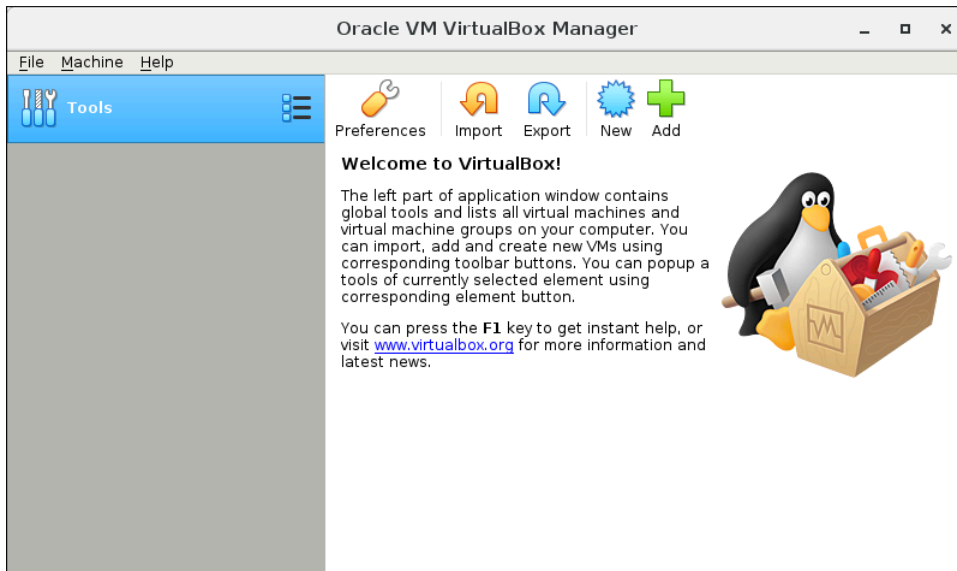
This window is called the **VirtualBox Manager**. The left pane will later list all your virtual machines. Since you have not yet created any virtual machines, this list is empty. The **Tools** button provides access to user tools, such as the Virtual Media Manager.

The pane on the right displays the properties of the currently selected virtual machine. Since you do not have any machines yet, the pane displays a welcome message.

The buttons on the right pane are used to create and work with VMs.

The following figure gives an idea of what Oracle VM VirtualBox might look like after you have created some VMs.

Figure 1.3 VirtualBox Manager Window, After Creating Virtual Machines
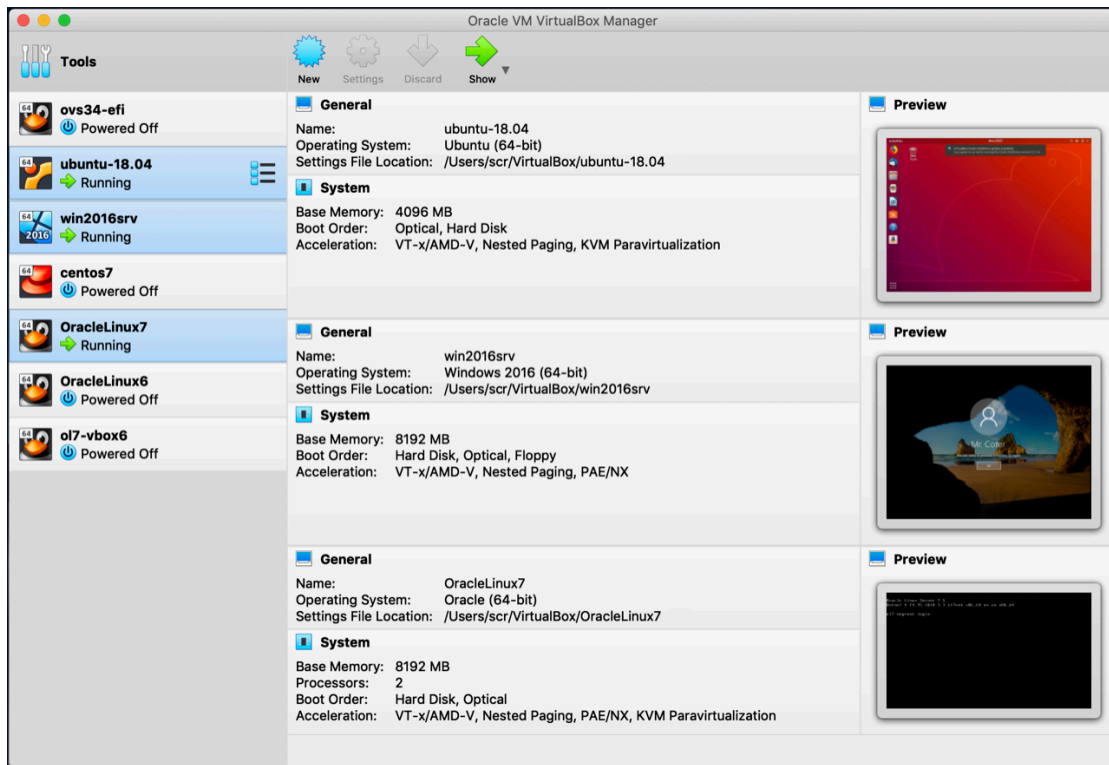
1.7. Creating Your First Virtual Machine

Click **New** in the VirtualBox Manager window. A wizard is shown, to guide you through setting up a new virtual machine (VM).

Figure 1.4 Creating a New Virtual Machine: Name and Operating System

On the following pages, the wizard will ask you for the bare minimum of information that is needed to create a VM, in particular:

1. The **Name** of the VM you choose is shown in the machine list of the VirtualBox Manager window and is also used for the VM's files on disk.

    Be sure to assign each VM an informative name that describes the OS and software running on the VM. For example, `Windows 10 with Visio`.

2. The **Machine Folder** is the location where VMs are stored on your computer. The default folder location is shown.

3. For **Operating System Type**, select the OS that you want to install. The supported OSes are grouped. If you want to install something very unusual that is not listed, select **Other**. Depending on your selection, Oracle VM VirtualBox will enable or disable certain VM settings that your guest OS may require. This is particularly important for 64-bit guests. See Section 3.1.2, "64-bit Guests". It is therefore recommended to always set it to the correct value.

4. On the next page, select the **Memory (RAM)** that Oracle VM VirtualBox should allocate every time the virtual machine is started. The amount of memory given here will be taken away from your host machine and presented to the guest OS, which will report this size as the virtual computer's installed RAM.

    > **Caution**
    >
    > Choose this setting carefully. The memory you give to the VM will not be available to your host OS while the VM is running, so do not specify more than you can spare. For example, if your host machine has 1 GB of RAM and you enter 512 MB as the amount of RAM for a particular virtual machine, while that VM is running, you will only have 512 MB left for all the other software on your host. If you run two VMs at the same time, even more memory will be allocated for the second VM, which may not even be able to start if that memory is not available. On the other hand, you should specify as much as your guest OS and your applications will require to run properly.

    A Windows XP guest will require at least a few hundred MB of RAM to run properly, and Windows Vista will not install with less than 512 MB. If you want to run graphics-intensive applications in your VM, you may require even more RAM.

    As a rule of thumb, if you have at least 1 GB of RAM in the host computer, you can safely allocate 512 MB to each VM. In any case, ensure that the host OS has at least 256 to 512 MB of RAM remaining. If insufficient RAM remains, the system might excessively swap memory to the hard disk, which effectively brings the host system to a standstill.

    As with the other settings, you can change this setting later, after you have created the VM.

5. Next, you must specify a **Virtual Hard Disk** for your VM.

    There are many and potentially complicated ways in which Oracle VM VirtualBox can provide hard disk space to a VM, see Chapter 5, *Virtual Storage*, but the most common way is to use a large image file on your physical hard disk, whose contents Oracle VM VirtualBox presents to your VM as if it were a

complete hard disk. This file then represents an entire hard disk, so you can even copy it to another host and use it with another Oracle VM VirtualBox installation.

The wizard displays the following window:

Figure 1.5 Creating a New Virtual Machine: Hard Disk

At this screen, you have the following options:

- To create a new, empty virtual hard disk, click the **Create** button.

- You can pick an *existing* disk image file.

  The drop-down list presented in the window lists all disk images which are currently remembered by Oracle VM VirtualBox. These disk images are currently attached to a virtual machine, or have been attached to a virtual machine.

  Alternatively, click on the small **folder icon** next to the drop-down list. In the displayed file dialog, you can click **Add** to select any disk image file on your host disk.

If you are using Oracle VM VirtualBox for the first time, you will want to create a new disk image. Click the **Create** button.

This displays another window, the **Create Virtual Hard Disk Wizard** wizard. This wizard helps you to create a new disk image file in the new virtual machine's folder.

Oracle VM VirtualBox supports the following types of image files:

- A **dynamically allocated file** only grows in size when the guest actually stores data on its virtual hard disk. Therefore, this file is small initially. As the drive is filled with data, the file grows to the specified size.

- A **fixed-size file** immediately occupies the file specified, even if only a fraction of that virtual hard disk space is actually in use. While occupying much more space, a fixed-size file incurs less overhead and is therefore slightly faster than a dynamically allocated file.

For details about the differences, see Section 5.2, "Disk Image Files (VDI, VMDK, VHD, HDD)".

To prevent your physical hard disk (host OS) from filling up, Oracle VM VirtualBox limits the size of the image file. But the image file must be large enough to hold the contents of the guest OS and the applications you want to install. For a Windows or Linux guest, you will probably need several gigabytes for any serious use. The limit of the image file size can be changed later, see Section 7.22, "VBoxManage modifymedium".

Figure 1.6 Creating a New Virtual Machine: File Location and Size

After having selected or created your image file, click **Next** to go to the next page.

6. Click **Create**, to create your new virtual machine. The virtual machine is displayed in the list on the left side of the VirtualBox Manager window, with the name that you entered initially.

Note

After becoming familiar with the use of wizards, consider using the Expert Mode available in some wizards. Where available, this is selectable using a button, and speeds up the process of using wizards.

1.8. Running Your Virtual Machine

To start a virtual machine, you have several options:

• Double-click on the VM's entry in the list in the VirtualBox Manager window.

• Select the VM's entry in the list in the VirtualBox Manager window, and click **Start** at the top of the window.

- Go to the `VirtualBox VMs` folder in your system user's home directory. Find the subdirectory of the machine you want to start and double-click on the machine settings file. This file has a `.vbox` file extension.

Starting a virtual machine displays a new window, and the virtual machine which you selected will boot up. Everything which would normally be seen on the virtual system's monitor is shown in the window. See the screenshot image in Chapter 1, *First Steps*.

In general, you can use the virtual machine as you would use a real computer. There are couple of points worth mentioning however.

1.8.1. Starting a New VM for the First Time

When a VM is started for the first time, the **First Start Wizard**, is displayed. This wizard helps you to select an installation medium. Since the VM is created empty, it would otherwise behave just like a real computer with no OS installed. It will do nothing and display an error message that no bootable OS was found.

For this reason, the wizard helps you to select a medium to install an OS from.

- If you have physical CD or DVD media from which you want to install your guest OS, such as a Windows installation CD or DVD, put the media into your host's CD or DVD drive.

  In the wizard's drop-down list of installation media, select **Host Drive** with the correct drive letter. In the case of a Linux host, choose a device file. This will allow your VM to access the media in your host drive, and you can proceed to install from there.

- If you have downloaded installation media from the Internet in the form of an ISO image file such as with a Linux distribution, you would normally burn this file to an empty CD or DVD and proceed as described above. With Oracle VM VirtualBox however, you can skip this step and mount the ISO file directly. Oracle VM VirtualBox will then present this file as a CD or DVD-ROM drive to the virtual machine, much like it does with virtual hard disk images.

  In this case, the wizard's drop-down list contains a list of installation media that were previously used with Oracle VM VirtualBox.

  If your medium is not in the list, especially if you are using Oracle VM VirtualBox for the first time, click the small folder icon next to the drop-down list to display a standard file dialog. Here you can pick an image file on your host disks.

After completing the choices in the wizard, you will be able to install your OS.

1.8.2. Capturing and Releasing Keyboard and Mouse

Oracle VM VirtualBox provides a virtual USB tablet device to new virtual machines through which mouse events are communicated to the guest OS. If you are running a modern guest OS that can handle such devices, mouse support may work out of the box without the mouse being *captured* as described below. See Section 3.5.1, "Motherboard Tab".

Otherwise, if the virtual machine detects only standard PS/2 mouse and keyboard devices, since the OS in the virtual machine does not know that it is not running on a real computer, it expects to have exclusive control over your keyboard and mouse. But unless you are running the VM in full screen mode, your VM needs to share keyboard and mouse with other applications and possibly other VMs on your host.

After installing a guest OS and before you install the Guest Additions, described later, either your VM or the rest of your computer can "own" the keyboard and the mouse. Both cannot own the keyboard and mouse at the same time. You will see a *second* mouse pointer which is always confined to the limits of the VM window. You activate the VM by clicking inside it.

To return ownership of keyboard and mouse to your host OS, Oracle VM VirtualBox reserves a special key on your keyboard: the *Host key*. By default, this is the *right Ctrl key* on your keyboard. On a Mac host, the default Host key is the left Command key. You can change this default in the Oracle VM VirtualBox Global Settings. See Section 1.15, "Global Settings". The current setting for the Host key is always displayed at the bottom right of your VM window.

Figure 1.7 Host Key Setting on the Virtual Machine Task Bar

This means the following:

* Your **keyboard** is owned by the VM if the VM window on your host desktop has the keyboard focus. If you have many windows open in your guest OS, the window that has the focus in your VM is used. This means that if you want to enter text within your VM, click on the title bar of your VM window first.

  To release keyboard ownership, press the Host key. As explained above, this is typically the right Ctrl key.

  Note that while the VM owns the keyboard, some key sequences, such as Alt+Tab, will no longer be seen by the host, but will go to the guest instead. After you press the Host key to reenable the host keyboard, all key presses will go through the host again, so that sequences such as Alt+Tab will no longer reach the guest. For technical reasons it may not be possible for the VM to get all keyboard input even when it does own the keyboard. Examples of this are the Ctrl+Alt+Del sequence on Windows hosts or single keys grabbed by other applications on X11 hosts such as the GNOME desktop Locate Pointer feature.

* Your **mouse** is owned by the VM only after you have clicked in the VM window. The host mouse pointer will disappear, and your mouse will drive the guest's pointer instead of your normal mouse pointer.

  Note that mouse ownership is independent of that of the keyboard. Even after you have clicked on a titlebar to be able to enter text into the VM window, your mouse is not necessarily owned by the VM yet.

  To release ownership of your mouse by the VM, press the Host key.

As this behavior is inconvenient, Oracle VM VirtualBox provides a set of tools and device drivers for guest systems called the Oracle VM VirtualBox Guest Additions. These tools make VM keyboard and mouse operations much more seamless. Most importantly, the Guest Additions suppress the second "guest" mouse pointer and make your host mouse pointer work directly in the guest. See Chapter 4, *Guest Additions*.

1.8.3. Typing Special Characters

Some OSes expect certain key combinations to initiate certain procedures. The key combinations that you type into a VM might target the host OS, the Oracle VM VirtualBox software, or the guest OS. The recipient of these keypresses depends on a number of factors, including the key combination itself.

* Host OSes reserve certain key combinations for themselves. For example, you cannot use the **Ctrl +Alt+Delete** combination to reboot the guest OS in your VM because this key combination is usually hard-wired into the host OS. So, even though both the Windows and Linux OSes intercept this key combination, only the host OS would be rebooted.

  On Linux and Oracle Solaris hosts, which use the X Window System, the key combination **Ctrl+Alt +Backspace** normally resets the X server and restarts the entire graphical user interface. As the X

server intercepts this combination, pressing it will usually restart your *host* graphical user interface and kill all running programs, including Oracle VM VirtualBox, in the process.

On Linux hosts supporting virtual terminals, the key combination **Ctrl+Alt+Fx**, where Fx is one of the function keys from F1 to F12, normally enables you to switch between virtual terminals. As with **Ctrl+Alt +Delete**, these combinations are intercepted by the host OS and therefore always switch terminals on the *host*.

If, instead, you want to send these key combinations to the *guest* OS in the virtual machine, you will need to use one of the following methods:

- Use the items in the **Input**, **Keyboard** menu of the virtual machine window. This menu includes the settings **Insert Ctrl+Alt+Delete** and **Insert Ctrl+Alt+Backspace**. However, the latter setting affects only Linux guests or Oracle Solaris guests.

  This menu also includes an option for inserting the Host key combination.

- Use special key combinations with the Host key, which is normally the right Control key. Oracle VM VirtualBox then translates the following key combinations for the VM:

  - **Host key + Del** sends **Ctrl+Alt+Del** to reboot the guest OS.

  - **Host key + Backspace** sends **Ctrl+Alt+Backspace** to restart the graphical user interface of a Linux or Oracle Solaris guest.

  - **Host key + Function key**. For example, use this key combination to simulate **Ctrl+Alt+Fx** to switch between virtual terminals in a Linux guest.

- For some other keyboard combinations such as **Alt+Tab** to switch between open windows, Oracle VM VirtualBox enables you to configure whether these combinations will affect the host or the guest, if a virtual machine currently has the focus. This is a global setting for all virtual machines and can be found under **File**, **Preferences**, **Input**.

- A soft keyboard can be used to input key combinations in the guest. See Section 1.17, "Soft Keyboard".

1.8.4. Changing Removable Media

While a virtual machine is running, you can change removable media in the **Devices** menu of the VM's window. Here you can select in detail what Oracle VM VirtualBox presents to your VM as a CD, DVD, or floppy drive.

The settings are the same as those available for the VM in the **Settings** dialog of the Oracle VM VirtualBox main window. But as the **Settings** dialog is disabled while the VM is in the Running or Saved state, the **Devices** menu saves you from having to shut down and restart the VM every time you want to change media.

Using the **Devices** menu, you can attach the host drive to the guest or select a floppy or DVD image, as described in Section 3.7, "Storage Settings".

The **Devices** menu also includes an option for creating a virtual ISO (VISO) from selected files on the host.

1.8.5. Resizing the Machine's Window

You can resize the VM's window while that VM is running. When you do, the window is scaled as follows:

1. If you have **scaled mode** enabled, then the virtual machine's screen will be scaled to the size of the window. This can be useful if you have many machines running and want to have a look at one of them

while it is running in the background. Alternatively, it might be useful to enlarge a window if the VM's output screen is very small, for example because you are running an old OS in it.

To enable scaled mode, press **Host key + C**, or select **Scaled Mode** from the **View** menu in the VM window. To leave scaled mode, press **Host key + C** again.

The aspect ratio of the guest screen is preserved when resizing the window. To ignore the aspect ratio, press **Shift** during the resize operation.

See Known Limitations for additional remarks.

2. If you have the Guest Additions installed and they support automatic **resizing**, the Guest Additions will automatically adjust the screen resolution of the guest OS. For example, if you are running a Windows guest with a resolution of 1024x768 pixels and you then resize the VM window to make it 100 pixels wider, the Guest Additions will change the Windows display resolution to 1124x768.

    See Chapter 4, *Guest Additions*.

3. Otherwise, if the window is bigger than the VM's screen, the screen will be centered. If it is smaller, then scroll bars will be added to the machine window.

1.8.6. Saving the State of the Machine

When you click on the **Close** button of your virtual machine window, at the top right of the window, just like you would close any other window on your system, Oracle VM VirtualBox asks you whether you want to save or power off the VM. As a shortcut, you can also press **Host key + Q**.

Figure 1.8 Closing Down a Virtual Machine
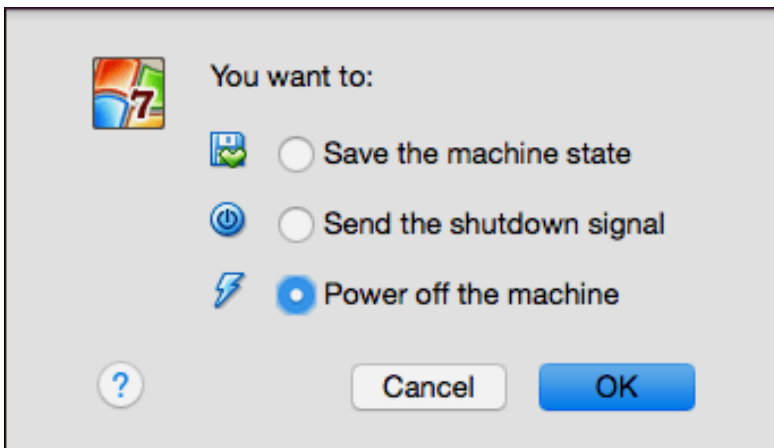
The difference between the three options is crucial. They mean the following:

*   **Save the machine state:** With this option, Oracle VM VirtualBox *freezes* the virtual machine by completely saving its state to your local disk.

    When you start the VM again later, you will find that the VM continues exactly where it was left off. All your programs will still be open, and your computer resumes operation. Saving the state of a virtual machine is thus in some ways similar to suspending a laptop computer by closing its lid.

*   **Send the shutdown signal.** This will send an ACPI shutdown signal to the virtual machine, which has the same effect as if you had pressed the power button on a real computer. This should trigger a proper shutdown mechanism from within the VM.

- **Power off the machine:** With this option, Oracle VM VirtualBox also stops running the virtual machine, but *without* saving its state.

> **Warning**
>
> This is equivalent to pulling the power plug on a real computer without shutting it down properly. If you start the machine again after powering it off, your OS will have to reboot completely and may begin a lengthy check of its virtual system disks. As a result, this should not normally be done, since it can potentially cause data loss or an inconsistent state of the guest system on disk.

As an exception, if your virtual machine has any snapshots, see Section 1.10, "Snapshots", you can use this option to quickly **restore the current snapshot** of the virtual machine. In that case, powering off the machine will not disrupt its state, but any changes made since that snapshot was taken will be lost.

The **Discard** button in the VirtualBox Manager window discards a virtual machine's saved state. This has the same effect as powering it off, and the same warnings apply.

1.9. Using VM Groups

VM groups enable the user to create ad hoc groups of VMs, and to manage and perform functions on them collectively, as well as individually.

The following figure shows VM groups displayed in VirtualBox Manager.

Figure 1.9 Groups of Virtual Machines

The following features are available for groups:

- Create a group using the VirtualBox Manager. Do one of the following:

  - Drag one VM on top of another VM.

  - Select multiple VMs and select **Group** from the right-click menu.

- Create and manage a group using the command line. Do one of the following:

  - Create a group and assign a VM. For example:

    ```
    VBoxManage modifyvm "vm01" --groups "/TestGroup"
    ```

    This command creates a group "TestGroup" and attaches the VM "vm01" to that group.

  - Detach a VM from the group, and delete the group if empty. For example:

    ```
    VBoxManage modifyvm "vm01" --groups ""
    ```

    This command detaches all groups from the VM "vm01" and deletes the empty group.

- Create multiple groups. For example:

  ```
  VBoxManage modifyvm "vm01" --groups "/TestGroup,/TestGroup2"
  ```

  This command creates the groups "TestGroup" and "TestGroup2", if they do not exist, and attaches the VM "vm01" to both of them.

- Create nested groups, having a group hierarchy. For example:

  ```
  VBoxManage modifyvm "vm01" --groups "/TestGroup/TestGroup2"
  ```

  This command attaches the VM "vm01" to the subgroup "TestGroup2" of the "TestGroup" group.

- The following is a summary of group commands: Start, Pause, Reset, Close (save state, send shutdown signal, poweroff), Discard Saved State, Show in File System, Sort.

1.10. Snapshots

With snapshots, you can save a particular state of a virtual machine for later use. At any later time, you can revert to that state, even though you may have changed the VM considerably since then. A snapshot of a virtual machine is thus similar to a machine in Saved state, but there can be many of them, and these saved states are preserved.

To see the snapshots of a virtual machine, click on the machine name in VirtualBox Manager. Then click the **List** icon next to the machine name, and select **Snapshots**. Until you take a snapshot of the machine, the list of snapshots will be empty except for the **Current State** item, which represents the "now" point in the lifetime of the virtual machine.

1.10.1. Taking, Restoring, and Deleting Snapshots

There are three operations related to snapshots, as follows:

1. **Take a snapshot**. This makes a copy of the machine's current state, to which you can go back at any given time later.

   - If your VM is running, select **Take Snapshot** from the **Machine** pull-down menu of the VM window.

   - If your VM is in either the Saved or the Powered Off state, as displayed next to the VM name in the Oracle VM VirtualBox main window, click the **List** icon next to the machine name and select **Snapshots**. The snapshots window is shown. Do one of the following:

     - Click the **Take** icon.

     - Right-click on the **Current State** item in the list and select **Take**.

In either case, a window is displayed prompting you for a snapshot name. This name is purely for reference purposes to help you remember the state of the snapshot. For example, a useful name would be "Fresh installation from scratch, no Guest Additions", or "Service Pack 3 just installed". You can also add a longer text in the **Description** field.

Your new snapshot will then appear in the snapshots list. Underneath your new snapshot, you will see an item called **Current State**, signifying that the current state of your VM is a variation based on the snapshot you took earlier. If you later take another snapshot, you will see that they are displayed in sequence, and that each subsequent snapshot is derived from an earlier one.

Figure 1.10 Snapshots List For a Virtual Machine
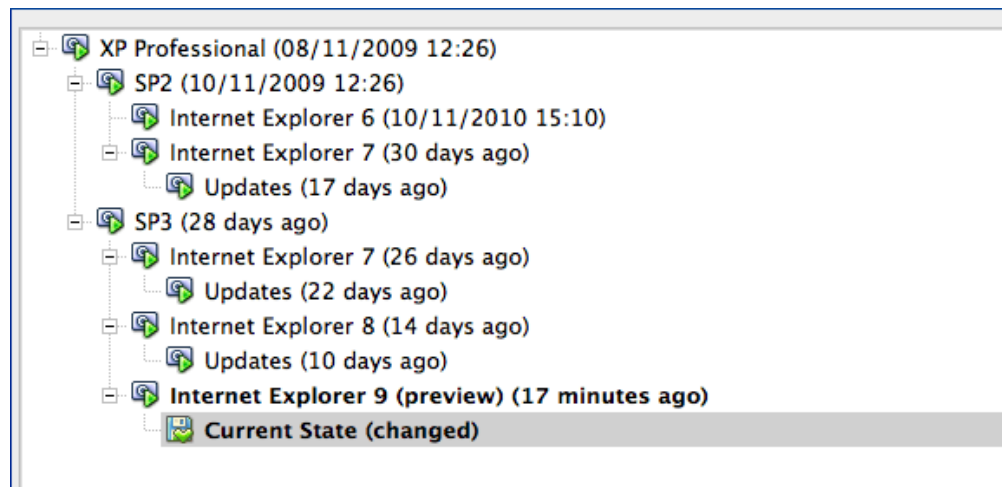
Oracle VM VirtualBox imposes no limits on the number of snapshots you can take. The only practical limitation is disk space on your host. Each snapshot stores the state of the virtual machine and thus occupies some disk space. See Section 1.10.2, "Snapshot Contents" for details on what is stored in a snapshot.

2. **Restore a snapshot**. In the list of snapshots, right-click on any snapshot you have taken and select **Restore**. By restoring a snapshot, you go back or forward in time. The current state of the machine is lost, and the machine is restored to the exact state it was in when the snapshot was taken.

> **Note**
>
> Restoring a snapshot will affect the virtual hard drives that are connected to your VM, as the entire state of the virtual hard drive will be reverted as well. This means also that all files that have been created since the snapshot and all other file changes *will be lost.* In order to prevent such data loss while still making use of the snapshot feature, it is possible to add a second hard drive in *write-through* mode using the `VBoxManage` interface and use it to store your data. As write-through hard drives are *not* included in snapshots, they remain unaltered when a machine is reverted. See Section 5.4, "Special Image Write Modes".

To avoid losing the current state when restoring a snapshot, you can create a new snapshot before the restore operation.

By restoring an earlier snapshot and taking more snapshots from there, it is even possible to create a kind of alternate reality and to switch between these different histories of the virtual machine. This can result in a whole tree of virtual machine snapshots, as shown in the screenshot above.

3. **Delete a snapshot**. This does not affect the state of the virtual machine, but only releases the files on disk that Oracle VM VirtualBox used to store the snapshot data, thus freeing disk space. To delete a snapshot, right-click on the snapshot name in the snapshots tree and select **Delete**. Snapshots can be deleted even while a machine is running.

> **Note**
>
> Whereas taking and restoring snapshots are fairly quick operations, deleting a snapshot can take a considerable amount of time since large amounts of data may need to be copied between several disk image files. Temporary disk files may also need large amounts of disk space while the operation is in progress.

There are some situations which cannot be handled while a VM is running, and you will get an appropriate message that you need to perform this snapshot deletion when the VM is shut down.

1.10.2. Snapshot Contents

Think of a snapshot as a point in time that you have preserved. More formally, a snapshot consists of the following:

- The snapshot contains a complete copy of the VM settings, including the hardware configuration, so that when you restore a snapshot, the VM settings are restored as well. For example, if you changed the hard disk configuration or the VM's system settings, that change is undone when you restore the snapshot.

  The copy of the settings is stored in the machine configuration, an XML text file, and thus occupies very little space.

- The complete state of all the virtual disks attached to the machine is preserved. Going back to a snapshot means that all changes that had been made to the machine's disks, file by file and bit by bit, will be undone as well. Files that were since created will disappear, files that were deleted will be restored, changes to files will be reverted.

  Strictly speaking, this is only true for virtual hard disks in "normal" mode. You can configure disks to behave differently with snapshots, see Section 5.4, "Special Image Write Modes". In technical terms, it is not the virtual disk itself that is restored when a snapshot is restored. Instead, when a snapshot is taken, Oracle VM VirtualBox creates differencing images which contain only the changes since the snapshot were taken. When the snapshot is restored, Oracle VM VirtualBox throws away that differencing image, thus going back to the previous state. This is both faster and uses less disk space. For the details, which can be complex, see Section 5.5, "Differencing Images".

  Creating the differencing image as such does not occupy much space on the host disk initially, since the differencing image will initially be empty and grow dynamically later with each write operation to the disk. The longer you use the machine after having created the snapshot, however, the more the differencing image will grow in size.

- If you took a snapshot while the machine was running, the memory state of the machine is also saved in the snapshot. This is in the same way that memory can be saved when you close a VM window. When you restore such a snapshot, execution resumes at exactly the point when the snapshot was taken.

  The memory state file can be as large as the memory size of the VM and will therefore occupy considerable disk space.

1.11. Virtual Machine Configuration

When you select a virtual machine from the list in the VirtualBox Manager window, you will see a summary of that machine's settings on the right.

Clicking on **Settings** displays a window, where you can configure many of the properties of the selected VM. But be careful when changing VM settings. It is possible to change all VM settings after installing a guest OS, but certain changes might prevent a guest OS from functioning correctly if done after installation.

> **Note**
>
> The **Settings** button is disabled while a VM is either in the Running or Saved state. This is because the **Settings** dialog enables you to change fundamental characteristics of the virtual machine that is created for your guest OS. For example, the guest OS may not perform well if half of its memory is taken away. As a result, if the **Settings** button is disabled, shut down the current VM first.

Oracle VM VirtualBox provides a wide range of parameters that can be changed for a virtual machine. The various settings that can be changed in the **Settings** window are described in detail in Chapter 3, *Configuring Virtual Machines*. Even more parameters are available when using the `VBoxManage` command line interface. See Chapter 7, *VBoxManage*.

1.12. Removing and Moving Virtual Machines

You can remove a VM from Oracle VM VirtualBox or move the VM and its associated files, such as disk images, to another location on the host.

- **Removing a VM.** To remove a VM, right-click on the VM in the VirtualBox Manager's machine list and select **Remove**.

  The confirmation dialog enables you to specify whether to only remove the VM from the list of machines or to remove the files associated with the VM.

  Note that the **Remove** menu item is disabled while a VM is running.

- **Moving a VM.** To move a VM to a new location on the host, right-click on the VM in the VirtualBox Manager's machine list and select **Move**.

  The file dialog prompts you to specify a new location for the VM.

  When you move a VM, Oracle VM VirtualBox configuration files are updated automatically to use the new location on the host.

  Note that the **Move** menu item is disabled while a VM is running.

  You can also use the `VBoxManage movevm` command to move a VM. See Section 7.9, "VBoxManage movevm".

For information about removing or moving a disk image file from Oracle VM VirtualBox, see Section 5.3, "The Virtual Media Manager".

1.13. Cloning Virtual Machines

You can create a full copy or a linked copy of an existing VM. This copy is called a *clone*. You might use a cloned VM to experiment with a VM configuration, to test different guest OS levels, or to back up a VM.

The **Clone Virtual Machine** wizard guides you through the cloning process.

Figure 1.11 The Clone Virtual Machine Wizard
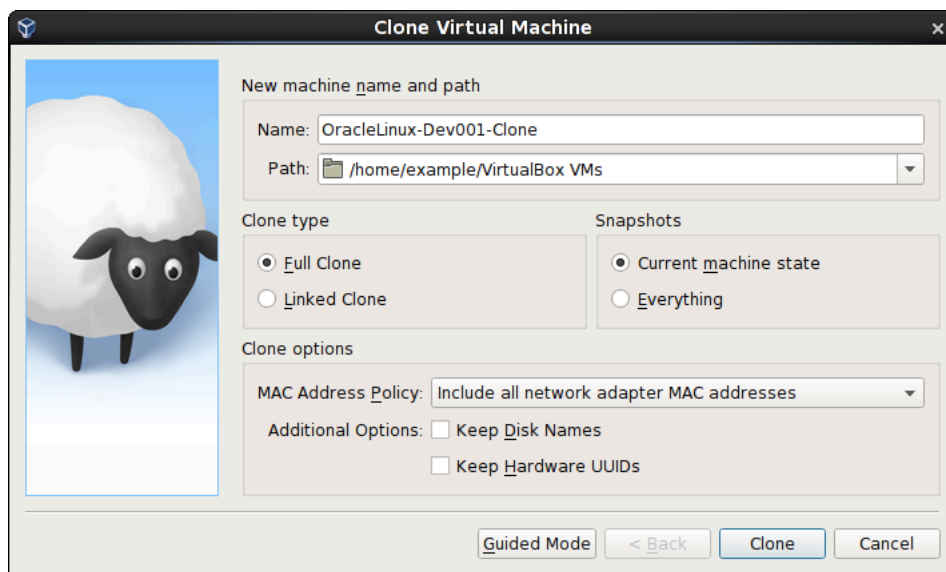
Start the wizard by clicking **Clone** in the right-click menu of the VirtualBox Manager's machine list or in the **Snapshots** view of the selected VM.

Specify a new **Name** for the clone. You can choose a **Path** for the cloned virtual machine, otherwise Oracle VM VirtualBox uses the default machines folder.

The **Clone Type** option specifies whether to create a clone linked to the source VM or to create a fully independent clone:

- **Full Clone:** Copies all dependent disk images to the new VM folder. A full clone can operate fully without the source VM.

- **Linked Clone:** Creates new differencing disk images based on the source VM disk images. If you select the current state of the source VM as the clone point, Oracle VM VirtualBox creates a new snapshot.

The **Snapshots** option specifies whether to create a clone of the current machine state only or of everything.

- **Everything:** Clones the current machine state and all its snapshots.

- **Current Machine State and All Children:**. Clones a VM snapshot and all its child snapshots.

The following clone options are available:

- **MAC Address Policy:** Specifies how to retain network card MAC addresses when cloning the VM.

  For example, the **Generate New MAC Addresses For All Network Adapters** value assigns a new MAC address to each network card during cloning. This is the default setting. This is the best option when both the source VM and the cloned VM must operate on the same network. Other values enable you to retain the existing MAC addresses in the cloned VM.

- **Keep Disk Names:** Retains the disk image names when cloning the VM.

- **Keep Hardware UUIDs:** Retains the hardware universally unique identifiers (UUIDs) when cloning the VM.

The duration of the clone operation depends on the size and number of attached disk images. In addition, the clone operation saves all the differencing disk images of a snapshot.

Note that the **Clone** menu item is disabled while a machine is running.

You can also use the `VBoxManage clonevm` command to clone a VM. See Section 7.40, "VBoxManage clonevm".

1.14. Importing and Exporting Virtual Machines

Oracle VM VirtualBox can import and export virtual machines in the following formats:

- **Open Virtualization Format (OVF).** This is the industry-standard format. See Section 1.14.1, "About the OVF Format".

- **Cloud service formats.** Exporting to and importing from cloud services such as Oracle Cloud Infrastructure is supported. See Section 1.14.4, "Exporting an Appliance to Oracle Cloud Infrastructure".

1.14.1. About the OVF Format

OVF is a cross-platform standard supported by many virtualization products which enables the creation of ready-made virtual machines that can then be imported into a hypervisor such as Oracle VM VirtualBox. Oracle VM VirtualBox makes OVF import and export easy to do, using the VirtualBox Manager window or the command-line interface.

Using OVF enables packaging of *virtual appliances*. These are disk images, together with configuration settings that can be distributed easily. This way one can offer complete ready-to-use software packages, including OSes with applications, that need no configuration or installation except for importing into Oracle VM VirtualBox.

> **Note**
>
> The OVF standard is complex, and support in Oracle VM VirtualBox is an ongoing process. In particular, no guarantee is made that Oracle VM VirtualBox supports all appliances created by other virtualization software. For a list of known limitations, see Known Limitations.

Appliances in OVF format can appear in the following variants:

- They can come in several files, as one or several disk images, typically in the widely-used VMDK format. See Section 5.2, "Disk Image Files (VDI, VMDK, VHD, HDD)". They also include a textual description file in an XML dialect with an `.ovf` extension. These files must then reside in the same directory for Oracle VM VirtualBox to be able to import them.

- Alternatively, the above files can be packed together into a single archive file, typically with an `.ova` extension. Such archive files use a variant of the TAR archive format and can therefore be unpacked outside of Oracle VM VirtualBox with any utility that can unpack standard TAR files.

> **Note**
>
> OVF cannot describe snapshots that were taken for a virtual machine. As a result, when you export a virtual machine that has snapshots, only the current state of the machine will be exported. The disk images in the export will have a *flattened* state identical to the current state of the virtual machine.

1.14.2. Importing an Appliance in OVF Format

The following steps show how to import an appliance in OVF format.

1. Double-click on the OVF or OVA file.

   Oracle VM VirtualBox creates file type associations automatically for any OVF and OVA files on your host OS.

2. Select **File**, **Import Appliance** from the VirtualBox Manager window.

   From the file dialog, go to the file with either the `.ovf` or the `.ova` file extension.

   Click **Import** to open the **Appliance Settings** screen.

   Figure 1.12 Appliance Settings Screen for Import Appliance
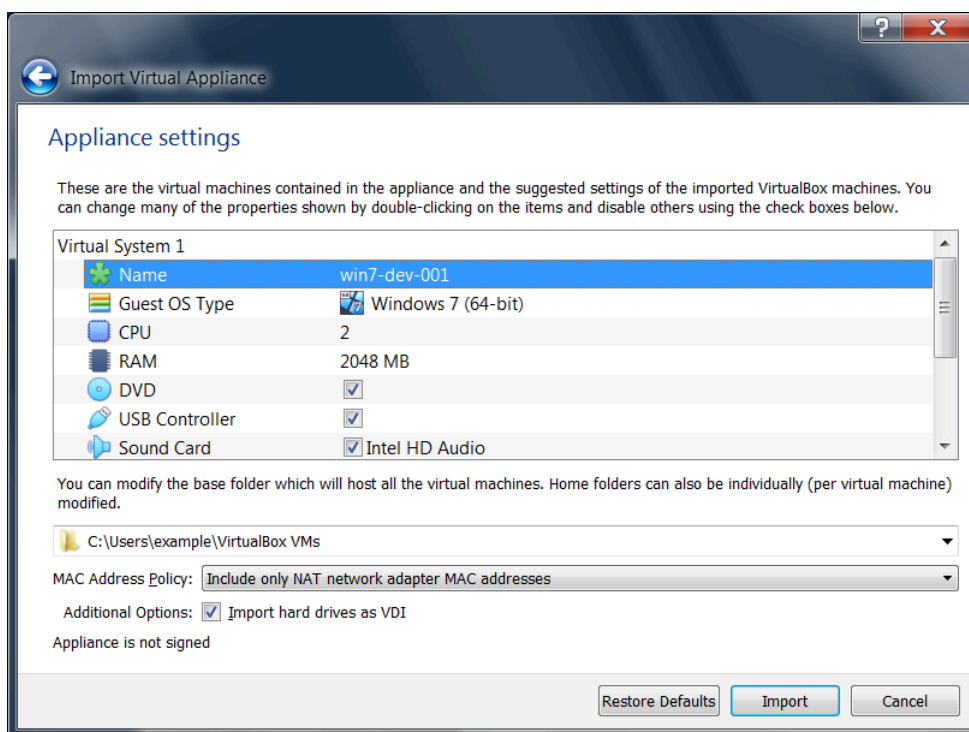
This screen shows the VMs described in the OVF or OVA file and enables you to change the VM settings.

By default, membership of VM groups is preserved on import for VMs that were initially exported from Oracle VM VirtualBox. You can change this behavior by using the **Primary Group** setting for the VM.

The following global settings apply to all of the VMs that you import:

• **Base Folder:** Specifies the directory on the host in which to store the imported VMs.

   If an appliance has multiple VMs, you can specify a different directory for each VM by editing the **Base Folder** setting for the VM.

• **MAC Address Policy:** Reinitializes the MAC addresses of network cards in your VMs prior to import, by default. You can override the default behavior and preserve the MAC addresses on import.

- **Import Hard Drives as VDI:** Imports hard drives in the VDI format rather than in the default VMDK format.

3. Click **Import** to import the appliance.

   Oracle VM VirtualBox copies the disk images and creates local VMs with the settings described on the **Appliance Settings** screen. The imported VMs are shown in the list of VMs in VirtualBox Manager.

   Because disk images are large, the VMDK images that are included with virtual appliances are shipped in a compressed format that cannot be used directly by VMs. So, the images are first unpacked and copied, which might take several minutes.

You can use the `VBoxManage import` command to import an appliance. See Section 7.10, "VBoxManage import".

1.14.3. Exporting an Appliance in OVF Format

The following steps show how to export an appliance in OVF format.

1. Select **File**, **Export Appliance** to open the **Export Virtual Appliance** wizard.

   From the initial window, you can combine several VMs into an OVF appliance.

   Select one or more VMs to export, and click **Next**.

2. The **Appliance Settings** screen enables you to select the following settings:

   - **Format:** Selects the **Open Virtualization Format** value for the output files.

     The **Oracle Cloud Infrastructure** value exports the appliance to Oracle Cloud Infrastructure. See Section 1.14.4, "Exporting an Appliance to Oracle Cloud Infrastructure".

   - **File:** Selects the location in which to store the exported files.

   - **MAC Address Policy:** Specifies whether to retain or reassign network card MAC addresses on export.

   - **Write Manifest File:** Enables you to include a manifest file in the exported archive file.

   - **Include ISO Image Files:** Enables you to include ISO image files in the exported archive file.

3. Click **Next** to show the **Virtual System Settings** screen.

   You can edit settings for the virtual appliance. For example, you can change the name of the virtual appliance or add product information, such as vendor details or license text.

   Double-click the appropriate field to change its value.

4. Click **Export** to begin the export process. Note that this operation might take several minutes.

You can use the `VBoxManage export` command to export an appliance. See Section 7.11, "VBoxManage export".

1.14.4. Exporting an Appliance to Oracle Cloud Infrastructure

Oracle VM VirtualBox supports the export of VMs to an Oracle Cloud Infrastructure service.

Before you can export a VM to Oracle Cloud Infrastructure, ensure that you perform the following configuration steps:

- For a custom Linux image, prepare the VM as described here:

  https://docs.cloud.oracle.com/iaas/Content/Compute/Tasks/preparingcustomimage.htm

- Generate an API signing key pair that is used for API requests to Oracle Cloud Infrastructure.

  - The key pair is usually installed in the `.oci` folder in your home directory. For example, `~/.oci` on a Linux system.

  - Upload the public key of the key pair to the cloud service.

  For step-by-step instructions for creating and uploading an API signing key for Oracle Cloud Infrastructure, see:

  https://docs.cloud.oracle.com/iaas/Content/API/Concepts/apisigningkey.htm#How

- Create a profile for your cloud account.

  The cloud profile contains resource identifiers for your cloud account, such as your user OCID, and the fingerprint for your public key. You can create a cloud profile in the following ways:

  - Automatically by using the **Cloud Profile Manager**. See Section 1.14.6, "The Cloud Profile Manager".

  - Manually by creating an `oci_config` file in your Oracle VM VirtualBox global configuration directory. For example, this is `$HOME/.config/VirtualBox/oci_config` on a Linux host.

  - Manually by creating a `config` file in your Oracle Cloud Infrastructure configuration directory. For example, this is `$HOME/.oci/config` on a Linux host.

    This is the same file that is used by the Oracle Cloud Infrastructure command line interface.

    Oracle VM VirtualBox automatically uses the `config` file if no cloud profile file is present in your global configuration directory. Alternatively, you can import this file manually into the Cloud Profile Manager.

  For more information about the cloud profile settings used by Oracle Cloud Infrastructure see:

  https://docs.cloud.oracle.com/iaas/Content/API/Concepts/sdkconfig.htm

- Ensure that the subnets that are used by source VMs are available in the target compartment on the cloud service.

Perform the following steps to export a VM to Oracle Cloud Infrastructure:

1. Select **File**, **Export Appliance** to open the **Export Virtual Appliance** wizard.

   Select a VM to export and click **Next** to open the **Appliance Settings** screen.

2. From the **Format** drop-down list, select **Oracle Cloud Infrastructure**.

   In the **Account** drop-down list, select your Oracle Cloud Infrastructure account.

   You can set up Oracle Cloud Infrastructure accounts by using the Cloud Profile Manager.

   The list after the **Account** field shows the profile settings for your cloud account.

Figure 1.13 Appliance Settings Screen, Showing Cloud Profile Settings
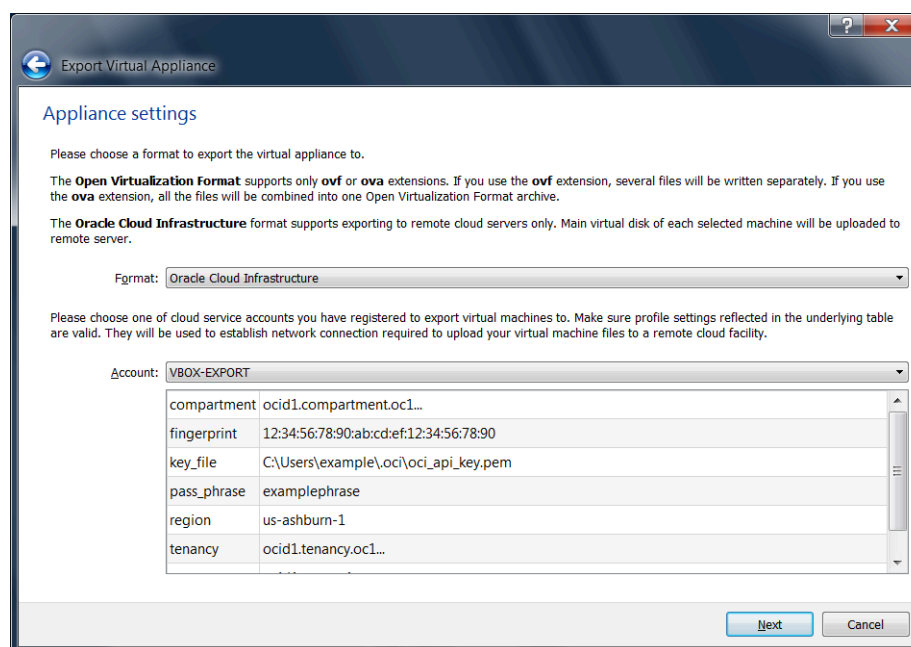
Click **Next** to make an API request to the Oracle Cloud Infrastructure service and open the **Virtual System Settings** screen.

3. Optionally edit settings used for the virtual machine on Oracle Cloud Infrastructure.

   For example, you can edit the Disk Size and Shape used for the VM instance.

   Click **Export** to export the VMs to the cloud service.

   The VMs are uploaded to Oracle Cloud Infrastructure.

   Instances are created for the uploaded VMs.

   By default, the VM instance is started after upload to Oracle Cloud Infrastructure.

4. Monitor the export process by using the Oracle Cloud Infrastructure Console.

You can also use the `VBoxManage export` command to export a VM to Oracle Cloud Infrastructure. See Section 7.11.2, "Export to Oracle Cloud Infrastructure".

1.14.5. Importing an Instance from Oracle Cloud Infrastructure

Oracle VM VirtualBox supports the import of cloud instances from an Oracle Cloud Infrastructure service.

Before you can import an instance from Oracle Cloud Infrastructure, ensure that you perform the following configuration steps:

- Generate an API signing key pair that is used for API requests to Oracle Cloud Infrastructure.

  - The key pair is usually installed in the `.oci` folder in your home directory. For example, `~/.oci` on a Linux system.

  - Upload the public key of the key pair to the cloud service.

For step-by-step instructions for creating and uploading an API signing key for Oracle Cloud Infrastructure, see:

https://docs.cloud.oracle.com/iaas/Content/API/Concepts/apisigningkey.htm#How

- Create a profile for your cloud account.

  The cloud profile contains resource identifiers for your cloud account, such as your user OCID, and the fingerprint for your public key. You can create a cloud profile in the following ways:

  - Automatically by using the **Cloud Profile Manager**. See Section 1.14.6, "The Cloud Profile Manager".

  - Manually by creating an `oci_config` file in your Oracle VM VirtualBox global configuration directory. For example, this is `$HOME/.config/VirtualBox/oci_config` on a Linux host.

  - Manually by creating a `config` file in your Oracle Cloud Infrastructure configuration directory. For example, this is `$HOME/.oci/config` on a Linux host.

    This is the same file that is used by the Oracle Cloud Infrastructure command line interface.

    Oracle VM VirtualBox automatically uses the `config` file if no cloud profile file is present in your global configuration directory. Alternatively, you can import this file manually into the Cloud Profile Manager.

  For more information about the cloud profile settings used by Oracle Cloud Infrastructure see:

  https://docs.cloud.oracle.com/iaas/Content/API/Concepts/sdkconfig.htm

Perform the following steps to import an instance from Oracle Cloud Infrastructure:

1. Select **File**, **Import Appliance** to open the **Import Virtual Appliance** wizard.

   In the **Source** drop-down list, select **Oracle Cloud Infrastructure**.

   In the **Account** drop-down list, select your Oracle Cloud Infrastructure account.

   You can set up Oracle Cloud Infrastructure accounts by using the Cloud Profile Manager.

   The list after the **Account** field shows the profile settings for your cloud account.

   Choose the required cloud instance from the list in the **Machines** field.

   Click **Next** to make an API request to the Oracle Cloud Infrastructure service and open the **Appliance Settings** screen.

2. Optionally edit settings for the new local virtual machine.

   For example, you can edit the VM name, number of CPUs, RAM memory, OS type, and OS version.

   Click **Import** to import the instance from the cloud service.

3. Monitor the import process by using the Oracle Cloud Infrastructure Console.

The following list describes the sequence of events when you import an instance from Oracle Cloud Infrastructure.

- A custom image is created from the boot volume of the instance.

- The custom image is exported to an Oracle Cloud Infrastructure object and is stored using Object Storage in the bucket specified by the user.

- The Oracle Cloud Infrastructure object is downloaded to the local host. The object is a TAR archive which contains a boot volume of the instance in QCOW2 format and a JSON file containing metadata related to the instance.

- The boot volume of the instance is extracted from the archive and a new VMDK image is created by converting the boot volume into the VMDK format. The VMDK image is registered with Oracle VM VirtualBox.

- A new VM is created using the VMDK image for the cloud instance.

  By default, the new VM is not started after import from Oracle Cloud Infrastructure.

- The downloaded TAR archive is deleted after a successful import.

You can also use the `VBoxManage import` command to import an instance from Oracle Cloud Infrastructure. See Section 7.10.2, "Import from Oracle Cloud Infrastructure".

1.14.6. The Cloud Profile Manager

The Cloud Profile Manager is a component of Oracle VM VirtualBox that enables you to create, edit, and manage cloud profiles for your cloud service accounts.

Figure 1.14 The Cloud Profile Manager
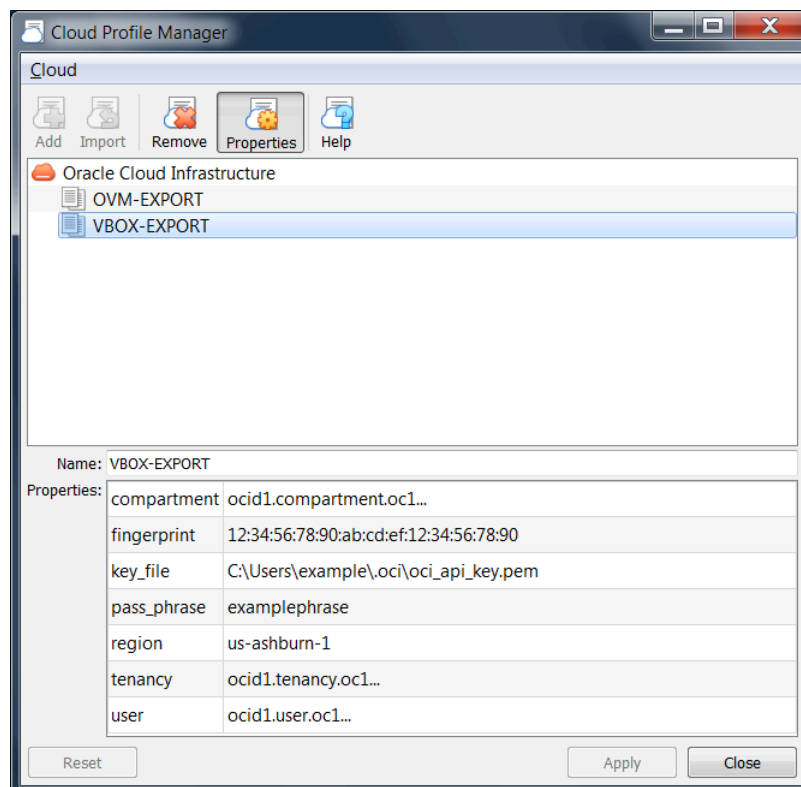
To open the Cloud Profile Manager select **File**, **Cloud Profile Manager** from the VirtualBox Manager window.

Use the Cloud Profile Manager to create a new cloud profile automatically. Or, create a cloud profile by importing settings from your Oracle Cloud Infrastructure configuration file into the Cloud Profile Manager.

Perform the following steps to create a new cloud profile:

1. Click the **Add** icon and specify a **Name** for the profile.

2. Click **Properties** and specify the following property values for the profile:

   • Compartment OCID

   • Fingerprint of the public key

   • Location of the private key on the client device

   • (Optional) Passphrase for the private key, if the key is encrypted

   • Region OCID

   • Tenancy OCID

   • User OCID

   Some of these are settings for your Oracle Cloud Infrastructure account, which you can view from the Oracle Cloud Infrastructure Console.

3. Click **Apply** to save your changes.

   The cloud profile settings are saved in the `oci_config` file in your Oracle VM VirtualBox global settings directory.

Perform the following steps to import an existing Oracle Cloud Infrastructure configuration file:

1. Ensure that a `config` file is present in your Oracle Cloud Infrastructure configuration directory. For example, this is `$HOME/.oci/config` on a Linux host.

2. Click the **Import** icon to open a dialog that prompts you to import cloud profiles from external files.

   Warning

   This action overwrites any cloud profiles that are in your Oracle VM VirtualBox global settings directory.

3. Click **Import**.

   Your cloud profile settings are saved to the `oci_config` file in your Oracle VM VirtualBox global settings directory.

4. Click **Properties** to show the cloud profile settings.

   Double-click on the appropriate field to change the value.

5. Click **Apply** to save your changes.

1.15. Global Settings

The **Global Settings** dialog can be displayed using the **File** menu, by clicking the **Preferences** item. This dialog offers a selection of settings, most of which apply to all virtual machines of the current user. The **Extensions** option applies to the entire system.

The following settings are available:

- **General.** Enables the user to specify the default folder or directory for VM files, and the VRDP Authentication Library.

- **Input.** Enables the user to specify the Host key. This is the key that toggles whether the cursor is in the focus of the VM or the Host OS windows, see Section 1.8.2, "Capturing and Releasing Keyboard and Mouse". The Host key is also used to trigger certain VM actions, see Section 1.8.3, "Typing Special Characters".

- **Update.** Enables the user to specify various settings for Automatic Updates.

- **Language.** Enables the user to specify the GUI language.

- **Display.** Enables the user to specify the screen resolution, and its width and height. A default scale factor can be specified for all guest screens.

- **Network.** Enables the user to configure the details of Host Only Networks.

- **Extensions.** Enables the user to list and manage the installed extension packages.

- **Proxy.** Enables the user to configure a HTTP Proxy Server.

1.16. Alternative Front-Ends

As briefly mentioned in Section 1.3, "Features Overview", Oracle VM VirtualBox has a very flexible internal design that enables you to use multiple interfaces to control the same virtual machines. For example, you can start a virtual machine with the VirtualBox Manager window and then stop it from the command line. With Oracle VM VirtualBox's support for the Remote Desktop Protocol (RDP), you can even run virtual machines remotely on a headless server and have all the graphical output redirected over the network.

The following front-ends are shipped in the standard Oracle VM VirtualBox package:

- **VirtualBox.** This is the VirtualBox Manager, a graphical user interface that uses the Qt toolkit. This interface is described throughout this manual. While this is the simplest and easiest front-end to use, some of the more advanced Oracle VM VirtualBox features are not included.

- **VBoxManage.** A command-line interface for automated and detailed control of every aspect of Oracle VM VirtualBox. See Chapter 7, *VBoxManage*.

- **VBoxHeadless.** A front-end that produces no visible output on the host at all, but can act as a RDP server if the VirtualBox Remote Desktop Extension (VRDE) is installed and enabled for the VM. As opposed to the other graphical interfaces, the headless front-end requires no graphics support. This is useful, for example, if you want to host your virtual machines on a headless Linux server that has no X Window system installed. See VBoxHeadless, the Remote Desktop Server.

If the above front-ends still do not satisfy your particular needs, it is possible to create yet another front-end to the complex virtualization engine that is the core of Oracle VM VirtualBox, as the Oracle VM VirtualBox core neatly exposes all of its features in a clean API. See Oracle VM VirtualBox Programming Interfaces.

1.17. Soft Keyboard

Oracle VM VirtualBox provides a *soft keyboard* that enables you to input keyboard characters on the guest. A soft keyboard is an on-screen keyboard that can be used as an alternative to a physical keyboard. See Section 1.17.1, "Using the Soft Keyboard" for details of how to use the soft keyboard.

Caution

For best results, ensure that the keyboard layout configured on the guest OS matches the keyboard layout used by the soft keyboard. Oracle VM VirtualBox does not do this automatically.

Figure 1.15 Soft Keyboard in a Guest Virtual Machine
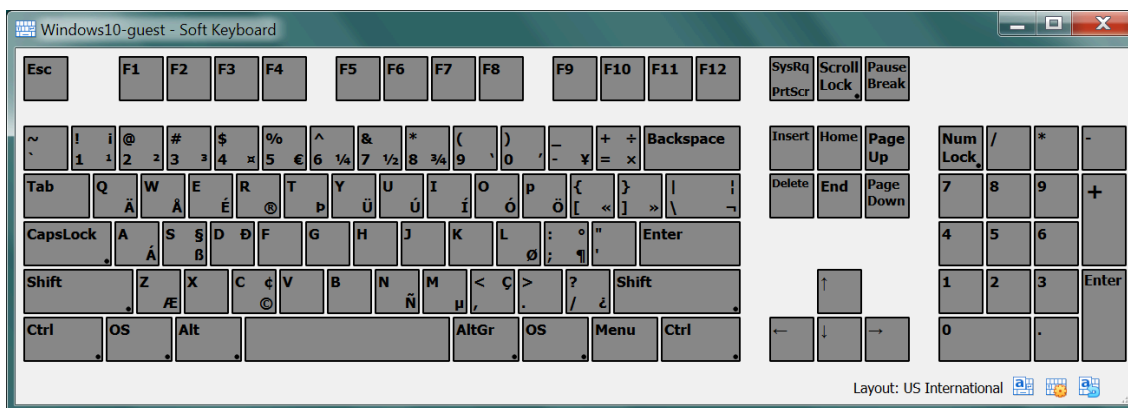
The soft keyboard can be used in the following scenarios:

- When the physical keyboard on the host is not the same as the keyboard layout configured on the guest. For example, if the guest is configured to use an international keyboard, but the host keyboard is US English.

- To send special key combinations to the guest. Note that some common key combinations are also available in the **Input**, **Keyboard** menu of the guest VM window. See Section 1.8.3, "Typing Special Characters".

- For guests in kiosk mode, where a physical keyboard is not present.

- When using nested virtualization, the soft keyboard provides a method of sending key presses to a guest.

By default, the soft keyboard includes some common international keyboard layouts. You can copy and modify these to meet your own requirements. See Section 1.17.2, "Creating a Custom Keyboard Layout".

1.17.1. Using the Soft Keyboard

1. Display the soft keyboard.

   In the guest VM window, select **Input**, **Keyboard**, **Soft Keyboard**.

2. Select the required keyboard layout.

   The name of the current keyboard layout is displayed in the task bar of the soft keyboard window. This is the previous keyboard layout that was used.

   Click the **Layout List** icon in the task bar of the soft keyboard window. The **Layout List** window is displayed.

   Select the required keyboard layout from the entries in the **Layout List** window.

   The keyboard display graphic is updated to show the available input keys.

3. Use the soft keyboard to enter keyboard characters on the guest.

   - Modifier keys such as Shift, Ctrl, and Alt are available on the soft keyboard. Click once to select the modifier key, click twice to lock the modifier key.

     The **Reset the Keyboard and Release All Keys** icon can be used to release all pressed modifier keys, both on the host and the guest.

   - To change the look of the soft keyboard, click the **Settings** icon in the task bar. You can change colors used in the keyboard graphic, and can hide or show sections of the keyboard, such as the NumPad keys.

1.17.2. Creating a Custom Keyboard Layout

You can use one of the supplied default keyboard layouts as the starting point to create a custom keyboard layout.

> **Note**
>
> To permananently save a custom keyboard layout, you must save it to file. Otherwise, any changes you make are discarded when you close down the **Soft Keyboard** window.
>
> Custom keyboard layouts that you save are stored as an XML file on the host, in the `keyboardLayouts` folder in the global configuration data directory. For example, in `$HOME/.config/VirtualBox/keyboardLayouts` on a Linux host.

1. Display the **Layout List**.

   Click the **Layout List** icon in the task bar of the soft keyboard window.

2. Make a copy of an existing keyboard layout.

   Highlight the required layout and click the **Copy the Selected Layout** icon.

   A new layout entry with a name suffix of `-Copy` is created.

3. Edit the new keyboard layout.

   Highlight the new layout in the **Layout List** and click the **Edit the Selected Layout** icon.

   Enter a new name for the layout.

   Edit keys in the new layout. Click on the key that you want to edit and enter new key captions in the **Captions** fields.

   The keyboard graphic is updated with the new captions.

4. (Optional) Save the layout to file. This means that your custom keyboard layout will be available for future use.

   Highlight the new layout in the **Layout List** and click the **Save the Selected Layout into File** icon.

   Any custom layouts that you create can later be removed from the Layout List, by highlighting and clicking the **Delete the Selected Layout** icon.

Chapter 2 Installation Details

As installation of Oracle VM VirtualBox varies depending on your host operating system, the following sections provide installation instructions for Windows, Mac OS X, Linux, and Oracle Solaris.

2.1. Installing on Windows Hosts

2.1.1. Prerequisites

For the various versions of Windows that are supported as host operating systems, please refer to Section 1.4, "Supported Host Operating Systems".

In addition, Windows Installer must be present on your system. This should be the case for all supported Windows platforms.

2.1.2. Performing the Installation

The Oracle VM VirtualBox installation can be started in either of the following ways:

• By double-clicking on the executable file.

• By entering the following command:

```
VirtualBox-<version>-<revision>-Win.exe -extract
```

This will extract the installer into a temporary directory, along with the .MSI file. Run the following command to perform the installation:

```
msiexec /i VirtualBox-<version>-<revision>-Win.msi
```

Using either way displays the installation **Welcome** dialog and enables you to choose where to install Oracle VM VirtualBox, and which components to install. In addition to the Oracle VM VirtualBox application, the following components are available:

• **USB support.** This package contains special drivers for your Windows host that Oracle VM VirtualBox requires to fully support USB devices inside your virtual machines.

• **Networking.** This package contains extra networking drivers for your Windows host that Oracle VM VirtualBox needs to support Bridged Networking. This enables your VM's virtual network cards to be accessed from other machines on your physical network.

• **Python support.** This package contains Python scripting support for the Oracle VM VirtualBox API, see Oracle VM VirtualBox Programming Interfaces. For this to work, an already working Windows Python installation on the system is required.

See, for example: http://www.python.org/download/windows/.

Note

Python version at least 2.6 is required. Python 3 is also supported.

Depending on your Windows configuration, you may see warnings about unsigned drivers, or similar. Click **Continue** for these warnings, as otherwise Oracle VM VirtualBox might not function correctly after installation.

The installer will create an Oracle VM VirtualBox group in the Windows **Start** menu, which enables you to launch the application and access its documentation.

With standard settings, Oracle VM VirtualBox will be installed for all users on the local system. If this is not wanted, you must invoke the installer by first extracting as follows:

```
VirtualBox.exe -extract
```

Then, run either of the following commands on the extracted .MSI file. This will install Oracle VM VirtualBox only for the current user.

```
VirtualBox.exe -msiparams ALLUSERS=2
```

```
msiexec /i VirtualBox-<version>-Win.msi ALLUSERS=2
```

If you do not want to install all features of Oracle VM VirtualBox, you can set the optional `ADDLOCAL` parameter to explicitly name the features to be installed. The following features are available:

VBoxApplication   Main binaries of Oracle VM VirtualBox.

> **Note**
>
> This feature must not be absent, since it contains the minimum set of files to have working Oracle VM VirtualBox installation.

VBoxUSB   USB support.

VBoxNetwork   All networking support. This includes the VBoxNetworkFlt and VBoxNetworkAdp features.

VBoxNetworkFlt   Bridged networking support.

VBoxNetworkAdp   Host-only networking support

VBoxPython   Python support

For example, to only install USB support along with the main binaries, run either of the following commands:

```
VirtualBox.exe -msiparams ADDLOCAL=VBoxApplication,VBoxUSB
```

```
msiexec /i VirtualBox-<version>-Win.msi ADDLOCAL=VBoxApplication,VBoxUSB
```

The user is able to choose between NDIS5 and NDIS6 host network filter drivers during the installation. This is done using a command line parameter, `NETWORKTYPE`. The NDIS6 driver is the default for most supported Windows hosts. For some legacy Windows versions, the installer will automatically select the NDIS5 driver and this cannot be changed.

You can force an install of the legacy NDIS5 host network filter driver by specifying `NETWORKTYPE=NDIS5`. For example, to install the NDIS5 driver on Windows 7 use either of the following commands:

```
VirtualBox.exe -msiparams NETWORKTYPE=NDIS5
```

```
msiexec /i VirtualBox-<version>-Win;.msi NETWORKTYPE=NDIS5
```

2.1.3. Uninstallation

As Oracle VM VirtualBox uses the standard Microsoft Windows installer, Oracle VM VirtualBox can be safely uninstalled at any time. Click the program entry in the **Add/Remove Programs** list in the Windows Control Panel.

2.1.4. Unattended Installation

Unattended installations can be performed using the standard MSI support.

2.1.5. Public Properties

Public properties can be specified with the MSI API, to control additional behavior and features of the Windows host installer. Use either of the following commands:

```
VirtualBox.exe -msiparams NAME=VALUE [...]
```

```
msiexec /i VirtualBox-<version>-Win.msi NAME=VALUE [...]
```

The following public properties are available.

- VBOX_INSTALLDESKTOPSHORTCUT

  Specifies whether or not an Oracle VM VirtualBox icon on the desktop should be created.

  Set to 1 to enable, 0 to disable. Default is 1.

- VBOX_INSTALLQUICKLAUNCHSHORTCUT

  Specifies whether or not an Oracle VM VirtualBox icon in the Quick Launch Bar should be created.

  Set to 1 to enable, 0 to disable. Default is 1.

- VBOX_REGISTERFILEEXTENSIONS

  Specifies whether or not the file extensions .vbox, .vbox-extpack, .ovf, .ova, .vdi, .vmdk, .vhd and .vdd should be associated with Oracle VM VirtualBox. Files of these types then will be opened with Oracle VM VirtualBox.

  Set to 1 to enable, 0 to disable. Default is 1.

- VBOX_START

  Specifies whether to start Oracle VM VirtualBox right after successful installation.

  Set to 1 to enable, 0 to disable. Default is 1.

2.2. Installing on Mac OS X Hosts

2.2.1. Performing the Installation

For Mac OS X hosts, Oracle VM VirtualBox ships in a `dmg` disk image file. Perform the following steps to install on a Mac OS X host:

1. Double-click on the `dmg` file, to mount the contents.

2. A window opens, prompting you to double-click on the `VirtualBox.pkg` installer file displayed in that window.

3. This starts the installer, which enables you to select where to install Oracle VM VirtualBox.

4. An Oracle VM VirtualBox icon is added to the `Applications` folder in the Finder.

2.2.2. Uninstallation

To uninstall Oracle VM VirtualBox, open the disk image `dmg` file and double-click on the uninstall icon shown.

2.2.3. Unattended Installation

To perform a non-interactive installation of Oracle VM VirtualBox you can use the command line version of the installer application.

Mount the `dmg` disk image file, as described in the installation procedure, or use the following command line:

```
hdiutil attach /path/to/VirtualBox-xyz.dmg
```

Open a terminal session and run the following command:

```
sudo installer -pkg /Volumes/VirtualBox/VirtualBox.pkg -target /Volumes/Macintosh\ HD
```

2.3. Installing on Linux Hosts

2.3.1. Prerequisites

For the various versions of Linux that are supported as host operating systems, see Section 1.4, "Supported Host Operating Systems".

You may need to install the following packages on your Linux system before starting the installation. Some systems will do this for you automatically when you install Oracle VM VirtualBox.

- Qt 5.3.2 or later. Qt 5.6.2 or later is recommended.

- SDL 1.2.7 or later. This graphics library is typically called `libsdl` or similar.

> **Note**
>
> These packages are only required if you want to run the Oracle VM VirtualBox graphical user interfaces. In particular, `VirtualBox`, the graphical VirtualBox Manager, requires both Qt and SDL. If you only want to run `VBoxHeadless`, neither Qt nor SDL are required.

2.3.2. The Oracle VM VirtualBox Kernel Modules

In order to run other operating systems in virtual machines alongside your main operating system, Oracle VM VirtualBox needs to integrate very tightly with your system. To do this it installs a driver module called `vboxdrv` into the system kernel. The kernel is the part of the operating system which controls your processor and physical hardware. Without this kernel module, you can still use the VirtualBox Manager to configure virtual machines, but they will not start.

Network drivers called `vboxnetflt` and `vboxnetadp` are also installed. They enable virtual machines to make more use of your computer's network capabilities and are needed for any virtual machine networking beyond the basic NAT mode.

Since distributing driver modules separately from the kernel is not something which Linux supports well, the Oracle VM VirtualBox install process creates the modules on the system where they will be used. This means that you may need to install some software packages from the distribution which are needed for the build process. Required packages may include the following:

- GNU compiler (GCC)

- GNU Make (make)

- Kernel header files

Also ensure that all system updates have been installed and that your system is running the most up-to-date kernel for the distribution.

> **Note**
>
> The running kernel and the kernel header files must be updated to matching versions.

The following list includes some details of the required files for some common distributions. Start by finding the version name of your kernel, using the command `uname -r` in a terminal. The list assumes that you have not changed too much from the original installation, in particular that you have not installed a different kernel type.

- With Debian and Ubuntu-based distributions, you must install the correct version of the `linux-headers`, usually whichever of `linux-headers-generic`, `linux-headers-amd64`, `linux-headers-i686` or `linux-headers-i686-pae` best matches the kernel version name. Also, the `linux-kbuild` package if it exists. Basic Ubuntu releases should have the correct packages installed by default.

- On Fedora, Red Hat, Oracle Linux and many other RPM-based systems, the kernel version sometimes has a code of letters or a word close to the end of the version name. For example "uek" for the Oracle Unbreakable Enterprise Kernel or "default" or "desktop" for the standard kernels. In this case, the package name is `kernel-uek-devel` or equivalent. If there is no such code, it is usually `kernel-devel`.

- On some SUSE and openSUSE Linux versions, you may need to install the `kernel-source` and `kernel-syms` packages.

If you suspect that something has gone wrong with module installation, check that your system is set up as described above and try running the following command, as root:

```
rcvboxdrv setup
```

2.3.2.1. Kernel Modules and UEFI Secure Boot

If you are running on a system using UEFI (Unified Extensible Firmware Interface) Secure Boot, you may need to sign the following kernel modules before you can load them:

- `vboxdrv`

- `vboxnetadp`

- `vboxnetflt`

- `vboxpci`

See your system documentation for details of the kernel module signing process.

2.3.3. Performing the Installation

Oracle VM VirtualBox is available in a number of package formats native to various common Linux distributions. See Section 1.4, "Supported Host Operating Systems". In addition, there is an alternative generic installer (.run) which you can use on supported Linux distributions.

2.3.3.1. Installing Oracle VM VirtualBox from a Debian or Ubuntu Package

Download the appropriate package for your distribution. The following example assumes that you are installing to a 64-bit Ubuntu Xenial system. Use `dpkg` to install the Debian package,as follows:

```
sudo dpkg -i virtualbox-6.0_version-number_Ubuntu_xenial_amd64.deb
```

The installer will also try to build kernel modules suitable for the current running kernel. If the build process is not successful you will be shown a warning and the package will be left unconfigured. Look at `/var/log/vbox-install.log` to find out why the compilation failed. You may have to install the appropriate Linux kernel headers, see Section 2.3.2, "The Oracle VM VirtualBox Kernel Modules". After correcting any problems, run the following command:

```
sudo rcvboxdrv setup
```

This will start a second attempt to build the module.

If a suitable kernel module was found in the package or the module was successfully built, the installation script will attempt to load that module. If this fails, please see Linux Kernel Module Refuses to Load for further information.

Once Oracle VM VirtualBox has been successfully installed and configured, you can start it by clicking **VirtualBox** in your **Start** menu or from the command line. See Section 2.3.5, "Starting Oracle VM VirtualBox on Linux".

2.3.3.2. Using the Alternative Generic Installer (VirtualBox.run)

The alternative generic installer performs the following steps:

- Unpacks the application files to the target directory `/opt/VirtualBox/`, which cannot be changed.

- Builds and installs the Oracle VM VirtualBox kernel modules: `vboxdrv`, `vboxnetflt`, and `vboxnetadp`.

- Creates `/sbin/rcvboxdrv`, an init script to start the Oracle VM VirtualBox kernel module.

- Creates a new system group called `vboxusers`.

- Creates symbolic links in `/usr/bin` to a shell script `/opt/VirtualBox/VBox` which does some sanity checks and dispatches to the actual executables: `VirtualBox`, `VBoxVRDP`, `VBoxHeadless` and `VBoxManage`.

- Creates `/etc/udev/rules.d/60-vboxdrv.rules`, a description file for udev, if that is present, which makes the USB devices accessible to all users in the `vboxusers` group.

- Writes the installation directory to `/etc/vbox/vbox.cfg`.

The installer must be executed as root with either `install` or `uninstall` as the first parameter. For example:

```
sudo ./VirtualBox.run install
```

Or if you do not have the `sudo` command available, run the following as root instead:

```
./VirtualBox.run install
```

Add every user who needs to access USB devices from a VirtualBox guests to the group `vboxusers`. Either use the OS user management tools or run the following command as root:

```
sudo usermod -a -G vboxusers username
```

> **Note**
>
> The `usermod` command of some older Linux distributions does not support the `-a` option, which adds the user to the given group without affecting membership of other groups. In this case, find out the current group memberships with the `groups` command and add all these groups in a comma-separated list to the command line after the `-G` option. For example: `usermod -G group1,group2,vboxusers username`.

2.3.3.3. Performing a Manual Installation

If you cannot use the shell script installer described in Section 2.3.3.2, "Using the Alternative Generic Installer (VirtualBox.run)", you can perform a manual installation. Run the installer as follows:

```
./VirtualBox.run --keep --noexec
```

This will unpack all the files needed for installation in the directory `install` under the current directory. The Oracle VM VirtualBox application files are contained in `VirtualBox.tar.bz2` which you can unpack to any directory on your system. For example:

```
sudo mkdir /opt/VirtualBox
sudo tar jxf ./install/VirtualBox.tar.bz2 -C /opt/VirtualBox
```

To run the same example as root, use the following commands:

```
mkdir /opt/VirtualBox
tar jxf ./install/VirtualBox.tar.bz2 -C /opt/VirtualBox
```

The sources for Oracle VM VirtualBox's kernel module are provided in the `src` directory. To build the module, change to the directory and use the following command:

```
make
```

If everything builds correctly, run the following command to install the module to the appropriate module directory:

```
sudo make install
```

In case you do not have sudo, switch the user account to root and run the following command:

```
make install
```

The Oracle VM VirtualBox kernel module needs a device node to operate. The above `make` command will tell you how to create the device node, depending on your Linux system. The procedure is slightly different for a classical Linux setup with a `/dev` directory, a system with the now deprecated `devfs` and a modern Linux system with `udev`.

On certain Linux distributions, you might experience difficulties building the module. You will have to analyze the error messages from the build system to diagnose the cause of the problems. In general, make sure that the correct Linux kernel sources are used for the build process.

Note that the `/dev/vboxdrv` kernel module device node must be owned by root:root and must be read/writable only for the user.

Next, you install the system initialization script for the kernel module and activate the initialization script using the right method for your distribution, as follows:

```
cp /opt/VirtualBox/vboxdrv.sh /sbin/rcvboxdrv
```

This example assumes you installed Oracle VM VirtualBox to the `/opt/VirtualBox` directory.

Create a configuration file for Oracle VM VirtualBox, as follows:

```
mkdir /etc/vbox
echo INSTALL_DIR=/opt/VirtualBox > /etc/vbox/vbox.cfg
```

Create the following symbolic links:

```
ln -sf /opt/VirtualBox/VBox.sh /usr/bin/VirtualBox
ln -sf /opt/VirtualBox/VBox.sh /usr/bin/VBoxManage
ln -sf /opt/VirtualBox/VBox.sh /usr/bin/VBoxHeadless
```

2.3.3.4. Updating and Uninstalling Oracle VM VirtualBox

Before updating or uninstalling Oracle VM VirtualBox, you must terminate any virtual machines which are currently running and exit the Oracle VM VirtualBox or VBoxSVC applications. To update Oracle VM VirtualBox, simply run the installer of the updated version. To uninstall Oracle VM VirtualBox, run the installer as follows:

```
sudo ./VirtualBox.run uninstall
```

As root, you can use the following command:

```
./VirtualBox.run uninstall
```

You can uninstall the .run package as follows:

```
/opt/VirtualBox/uninstall.sh
```

To manually uninstall Oracle VM VirtualBox, perform the manual installation steps in reverse order.

2.3.3.5. Automatic Installation of Debian Packages

The Debian packages will request some user feedback when installed for the first time. The debconf system is used to perform this task. To prevent any user interaction during installation, default values can be defined. A file `vboxconf` can contain the following debconf settings:

```
virtualbox virtualbox/module-compilation-allowed boolean true
virtualbox virtualbox/delete-old-modules boolean true
```

The first line enables compilation of the vboxdrv kernel module if no module was found for the current kernel. The second line enables the package to delete any old vboxdrv kernel modules compiled by previous installations.

These default settings can be applied prior to the installation of the Oracle VM VirtualBox Debian package, as follows:

```
debconf-set-selections vboxconf
```

In addition there are some common configuration options that can be set prior to the installation. See Section 2.3.3.7, "Automatic Installation Options".

2.3.3.6. Automatic Installation of RPM Packages

The RPM format does not provide a configuration system comparable to the debconf system. See Section 2.3.3.7, "Automatic Installation Options" for how to set some common installation options provided by Oracle VM VirtualBox.

2.3.3.7. Automatic Installation Options

To configure the installation process for .deb and .rpm packages, you can create a response file named `/etc/default/virtualbox`. The automatic generation of the udev rule can be prevented with the following setting:

```
INSTALL_NO_UDEV=1
```

The creation of the group vboxusers can be prevented as follows:

```
INSTALL_NO_GROUP=1
```

If the following line is specified, the package installer will not try to build the `vboxdrv` kernel module if no module fitting the current kernel was found.

```
INSTALL_NO_VBOXDRV=1
```

2.3.4. The vboxusers Group

The Linux installers create the system user group `vboxusers` during installation. Any system user who is going to use USB devices from Oracle VM VirtualBox guests must be a member of that group. A user can be made a member of the group `vboxusers` either by using the desktop user and group tools, or with the following command:

```
sudo usermod -a -G vboxusers username
```

2.3.5. Starting Oracle VM VirtualBox on Linux

The easiest way to start an Oracle VM VirtualBox program is by running the program of your choice (`VirtualBox`, `VBoxManage`, or `VBoxHeadless`) from a terminal. These are symbolic links to `VBox.sh` that start the required program for you.

The following detailed instructions should only be of interest if you wish to execute Oracle VM VirtualBox without installing it first. You should start by compiling the `vboxdrv` kernel module and inserting it into the Linux kernel. Oracle VM VirtualBox consists of a service daemon, `VBoxSVC`, and several application programs. The daemon is automatically started if necessary. All Oracle VM VirtualBox applications will communicate with the daemon through UNIX local domain sockets. There can be multiple daemon instances under different user accounts and applications can only communicate with the daemon running under the user account as the application. The local domain socket resides in a subdirectory of your system's directory for temporary files called `.vbox-<username>-ipc`. In case of communication problems or server startup problems, you may try to remove this directory.

All Oracle VM VirtualBox applications (`VirtualBox`, `VBoxManage`, and `VBoxHeadless`) require the Oracle VM VirtualBox directory to be in the library path, as follows:

```
LD_LIBRARY_PATH=. ./VBoxManage showvminfo "Windows XP"
```

2.4. Installing on Oracle Solaris Hosts

For the specific versions of Oracle Solaris that are supported as host operating systems, see Section 1.4, "Supported Host Operating Systems".

If you have a previously installed instance of Oracle VM VirtualBox on your Oracle Solaris host, please uninstall it first before installing a new instance. See Section 2.4.4, "Uninstallation" for uninstall instructions.

2.4.1. Performing the Installation

Oracle VM VirtualBox is available as a standard Oracle Solaris package. Download the Oracle VM VirtualBox SunOS package, which includes the 64-bit version of Oracle VM VirtualBox. *The installation must be performed as root and from the global zone*. This is because the Oracle VM VirtualBox installer loads kernel drivers, which cannot be done from non-global zones. To verify which zone you are currently in, execute the `zonename` command.

To start installation, run the following commands:

```
gunzip -cd VirtualBox-version-number-SunOS.tar.gz | tar xvf -
```

The Oracle VM VirtualBox kernel package is integrated into the main package. Install the Oracle VM VirtualBox package as follows:

```
pkgadd -d VirtualBox-version-number-SunOS.pkg
```

The installer will then prompt you to enter the package you wish to install. Choose **1** or **all** and proceed. Next the installer will ask you if you want to allow the postinstall script to be executed. Choose **y** and proceed, as it is essential to execute this script which installs the Oracle VM VirtualBox kernel module. Following this confirmation the installer will install Oracle VM VirtualBox and execute the postinstall setup script.

Once the postinstall script has been executed your installation is now complete. You may now safely delete the uncompressed package and `autoresponse` files from your system. Oracle VM VirtualBox is installed in `/opt/VirtualBox`.

> **Note**
>
> If you need to use Oracle VM VirtualBox from non-global zones, see Section 2.4.6, "Configuring a Zone for Running Oracle VM VirtualBox".

2.4.2. The vboxuser Group

The installer creates the system user group `vboxuser` during installation for Oracle Solaris hosts that support the USB features required by Oracle VM VirtualBox. Any system user who is going to use USB devices from Oracle VM VirtualBox guests must be a member of this group. A user can be made a member of this group either by using the desktop user and group tools or by running the following command as root:

```
usermod -G vboxuser username
```

Note that adding an active user to the `vboxuser` group will require the user to log out and then log in again. This should be done manually after successful installation of the package.

2.4.3. Starting Oracle VM VirtualBox on Oracle Solaris

The easiest way to start an Oracle VM VirtualBox program is by running the program of your choice (`VirtualBox`, `VBoxManage`, or `VBoxHeadless`) from a terminal. These are symbolic links to `VBox.sh` that start the required program for you.

Alternatively, you can directly invoke the required programs from `/opt/VirtualBox`. Using the links provided is easier as you do not have to enter the full path.

You can configure some elements of the `VirtualBox` Qt GUI, such as fonts and colours, by running `VBoxQtconfig` from the terminal.

2.4.4. Uninstallation

Uninstallation of Oracle VM VirtualBox on Oracle Solaris requires root permissions. To perform the uninstallation, start a root terminal session and run the following command:

```
pkgrm SUNWvbox
```

After confirmation, this will remove Oracle VM VirtualBox from your system.

2.4.5. Unattended Installation

To perform a non-interactive installation of Oracle VM VirtualBox there is a response file named `autoresponse`. The installer uses this for responses to inputs, rather than prompting the user.

Extract the tar.gz package as described in Section 2.4.1, "Performing the Installation". Then open a root terminal session and run the following command:

```
pkgadd -d VirtualBox-version-number-SunOS-x86 -n -a autoresponse SUNWvbox
```

To perform a non-interactive uninstallation, open a root terminal session and run the following command:

```
pkgrm -n -a /opt/VirtualBox/autoresponse SUNWvbox
```

2.4.6. Configuring a Zone for Running Oracle VM VirtualBox

Assuming that Oracle VM VirtualBox has already been installed into your zone, you need to give the zone access to Oracle VM VirtualBox's device node. This is done by performing the following steps. Start a root terminal and run the following command:

```
zonecfg -z vboxzone
```

Replace `vboxzone` with the name of the zone where you intend to run Oracle VM VirtualBox.

Use `zonecfg` to add the `device` resource and `match` properties to the zone, as follows:

```
zonecfg:vboxzone>add device
zonecfg:vboxzone:device>set match=/dev/vboxdrv
zonecfg:vboxzone:device>end
zonecfg:vboxzone>add device
zonecfg:vboxzone:device>set match=/dev/vboxdrvu
zonecfg:vboxzone:device>end
zonecfg:vboxzone>exit
```

On Oracle Solaris 11 or later, you may also add a device for `/dev/vboxusbmon`, similar to that shown above.

If you are not using sparse root zones, you will need to loopback mount `/opt/VirtualBox` from the global zone into the non-global zone at the same path. This is specified below using the `dir` attribute and the `special` attribute. For example:

```
zonecfg:vboxzone>add fs
zonecfg:vboxzone:device>set dir=/opt/VirtualBox
zonecfg:vboxzone:device>set special=/opt/VirtualBox
zonecfg:vboxzone:device>set type=lofs
zonecfg:vboxzone:device>end
zonecfg:vboxzone>exit
```

Reboot the zone using `zoneadm` and you should be able to run Oracle VM VirtualBox from within the configured zone.

Chapter 3 Configuring Virtual Machines

This chapter provides detailed steps for configuring an Oracle VM VirtualBox virtual machine (VM). For an introduction to Oracle VM VirtualBox and steps to get your first virtual machine running, see Chapter 1, *First Steps*.

You have considerable latitude when deciding what virtual hardware to provide to the guest. Use virtual hardware to communicate with the host system or with other guests. For example, you can use virtual hardware in the following ways:

- Have Oracle VM VirtualBox present an ISO CD-ROM image to a guest system as if it were a physical CD-ROM.

- Provide a guest system access to the physical network through its virtual network card.

- Provide the host system, other guests, and computers on the Internet access to the guest system.

3.1. Supported Guest Operating Systems

Because Oracle VM VirtualBox is designed to provide a generic virtualization environment for x86 systems, it can run operating systems (OSes) of any kind. However, Oracle VM VirtualBox focuses on the following guest systems:

- **Windows NT 4.0:**

  - Fully supports all versions, editions, and service packs. Note that you might encounter issues with some older service packs, so install at least service pack 6a.

  - Guest Additions are available with a limited feature set.

- **Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10 (non-Insider Preview releases), Windows Server 2016, Windows Server 2019:**

  - Fully supports all versions, editions, and service packs, including 64-bit versions.

  - Note that you must enable hardware virtualization when running at least Windows 8.

  - Guest Additions are available.

- **MS-DOS, Windows 3.x, Windows 95, Windows 98, Windows ME:**

  - Limited testing has been performed.

  - Use beyond legacy installation mechanisms is not recommended.

  - Guest Additions are not available.

- **Linux 2.4:**

  Limited support.

- **Linux 2.6:**

  - Fully supports all versions and editions, both 32-bit and 64-bit.

- For best performance, use at least Linux kernel version 2.6.13.

- Guest Additions are available.

  > **Note**
  >
  > Certain Linux kernel releases have bugs that prevent them from executing in a virtual environment. See Buggy Linux 2.6 Kernel Versions.

- **Linux 3.x and later:**

  - Fully supports all versions and editions, both 32-bit and 64-bit.

  - Guest Additions are available.

- **Oracle Solaris 10 and Oracle Solaris 11:**

  - Fully supports all versions starting with Oracle Solaris 10 8/08 and Oracle Solaris 11.

  - Supports 64-bit prior to Oracle Solaris 11 11/11, and 32-bit.

  - Guest Additions are available.

- **FreeBSD:**

  - Limited support.

  - Note that you must enable hardware virtualization when running FreeBSD.

  - Guest Additions are not available.

- **OpenBSD:**

  - Supports at least version 3.7.

  - Note that you must enable hardware virtualization when running OpenBSD.

  - Guest Additions are not available.

- **OS/2 Warp 4.5:**

  - Only MCP2 is supported. Other OS/2 versions might not work.

  - Note that you must enable hardware virtualization when running OS/2 Warp 4.5.

  - Guest Additions are available with a limited feature set. See Known Limitations.

- **Mac OS X:**

  - Oracle VM VirtualBox 3.2 added experimental support for Mac OS X guests, with restrictions. See Section 3.1.1, "Mac OS X Guests" and Known Limitations.

  - Guest Additions are not available.

3.1.1. Mac OS X Guests

Oracle VM VirtualBox enables you to install and execute unmodified versions of Mac OS X guests on supported host hardware. Note that this feature is experimental and thus unsupported.

Oracle VM VirtualBox is the first product to provide the modern PC architecture expected by OS X without requiring any of the modifications used by competing virtualization solutions. For example, some competing solutions perform modifications to the Mac OS X install DVDs, such as a different boot loader and replaced files.

Be aware of the following important issues before you attempt to install a Mac OS X guest:

- Mac OS X is commercial, licensed software and contains **both license and technical restrictions** that limit its use to certain hardware and usage scenarios. You must understand and comply with these restrictions.

  In particular, Apple prohibits the installation of most versions of Mac OS X on non-Apple hardware.

  These license restrictions are also enforced on a technical level. Mac OS X verifies that it is running on Apple hardware. Most DVDs that accompany Apple hardware check for the exact model. These restrictions are *not* circumvented by Oracle VM VirtualBox and continue to apply.

- Only **CPUs** that are known and tested by Apple are supported. As a result, if your Intel CPU is newer than the Mac OS X build, or if you have a non-Intel CPU, you will likely encounter a panic during bootup with an "Unsupported CPU" exception.

  Ensure that you use the Mac OS X DVD that comes with your Apple hardware.

- The Mac OS X installer expects the hard disk to be *partitioned*. So, the installer will not offer a partition selection to you. Before you can install the software successfully, start the Disk Utility from the Tools menu and partition the hard disk. Close the Disk Utility and proceed with the installation.

- In addition, Mac OS X support in Oracle VM VirtualBox is an experimental feature. See Known Limitations.

3.1.2. 64-bit Guests

Warning

Be sure to enable **I/O APIC** for virtual machines that you intend to use in 64-bit mode. This is especially true for 64-bit Windows VMs. See Section 3.4.2, "Advanced Tab". For 64-bit Windows guests, ensure that the VM uses the **Intel networking device** because there is no 64-bit driver support for the AMD PCNet card. See Section 6.1, "Virtual Networking Hardware".

If you use the **Create VM** wizard of the VirtualBox Manager, Oracle VM VirtualBox automatically uses the correct settings for each selected 64-bit OS type. See Section 1.7, "Creating Your First Virtual Machine".

3.2. Unattended Guest Installation

Oracle VM VirtualBox can install a guest OS automatically. You only need to provide the installation medium and a few other parameters, such as the name of the default user.

Performing an unattended guest installation involves the following steps:

- **Create a new VM.** Use one of the following methods:

  - The VirtualBox Manager, see Section 1.7, "Creating Your First Virtual Machine".

  - The `VBoxManage createvm` command, see Section 7.7, "VBoxManage createvm".

  For the new VM, choose the guest OS type and accept the default settings for that OS. The following sections in this chapter describe how to change the settings for a VM.

- **Prepare the VM for unattended guest installation.** Use the `VBoxManage unattended` command, see Section 7.38, "VBoxManage unattended".

  During this step, Oracle VM VirtualBox scans the installation medium and changes certain parameters to ensure a seamless installation as a guest running on Oracle VM VirtualBox.

- **Start the VM.** Use the VirtualBox Manager or the `VBoxManage startvm` command.

  When you start the VM, the unattended installation is performed automatically.

  The installation operation changes the boot device order to boot the virtual hard disk first and then the virtual DVD drive. If the virtual hard disk is empty prior to the automatic installation, the VM boots from the virtual DVD drive and begins the installation.

  If the virtual hard disk contains a bootable OS, the installation operation exits. In this case, change the boot device order manually by pressing F12 during the BIOS splash screen.

Section 3.2.1, "An Example of Unattended Guest Installation" describes how to perform an unattended guest installation for an Oracle Linux guest.

3.2.1. An Example of Unattended Guest Installation

The following example shows how to perform an unattended guest installation for an Oracle Linux VM. The example uses various `VBoxManage` commands to prepare the guest VM. The `VBoxManage unattended install` command is then used to install and configure the guest OS.

1. Create the virtual machine.

```
# VM="ol7-autoinstall"
# VBoxManage list ostypes
# VBoxManage createvm --name $VM --ostype "Oracle_64" --register
```

Note the following:

- The $VM variable represents the name of the VM.

- The `VBoxManage list ostypes` command lists the guest OSes supported by Oracle VM VirtualBox, including the name used for each OS in the `VBoxManage` commands.

- A 64-bit Oracle Linux 7 VM is created and registered with Oracle VM VirtualBox.

- The VM has a unique UUID.

- An XML settings file is generated.

2. Create a virtual hard disk and storage devices for the VM.

```
# VBoxManage createhd --filename /VirtualBox/$VM/$VM.vdi --size 32768
# VBoxManage storagectl $VM --name "SATA Controller" --add sata --controller IntelAHCI
# VBoxManage storageattach $VM --storagectl "SATA Controller" --port 0 --device 0 \
--type hdd --medium /VirtualBox/$VM/$VM.vdi
# VBoxManage storagectl $VM --name "IDE Controller" --add ide
# VBoxManage storageattach $VM --storagectl "IDE Controller" --port 0 --device 0 \
--type dvddrive --medium /u01/Software/OL/OracleLinux-R7-U6-Server-x86_64-dvd.iso
```

The previous commands do the following:

- Create a 32768 MB virtual hard disk.

- Create a SATA storage controller and attach the virtual hard disk.

- Create an IDE storage controller for a virtual DVD drive and attach an Oracle Linux installation ISO.

3. (Optional) Configure some settings for the VM.

```
# VBoxManage modifyvm $VM --ioapic on
# VBoxManage modifyvm $VM --boot1 dvd --boot2 disk --boot3 none --boot4 none
# VBoxManage modifyvm $VM --memory 8192 --vram 128
```

The previous commands do the following:

- Enable I/O APIC for the motherboard of the VM.

- Configure the boot device order for the VM.

- Allocate 8192 MB of RAM and 128 MB of video RAM to the VM.

4. Perform an unattended install of the OS.

```
# VBoxManage unattended install $VM \
--iso=/u01/Software/OL/OracleLinux-R7-U6-Server-x86_64-dvd.iso \
--user=login --full-user-name=name --password password \
--install-additions --time-zone=CET
```

The previous command does the following:

- Specifies an Oracle Linux ISO as the installation ISO.

- Specifies a login name, full name, and login password for a default user on the guest OS.

  Note that the specified password is also used for the root user account on the guest.

- Installs the Guest Additions on the VM.

- Sets the time zone for the guest OS to Central European Time (CET).

5. Start the virtual machine.

   This step completes the unattended installation process.

```
# VBoxManage startvm $VM --type headless
```

The VM starts in headless mode, which means that the VirtualBox Manager window does not open.

6. (Optional) Update the guest OS to use the latest Oracle Linux packages.

   On the guest VM, run the following command:

```
# yum update
```

3.3. Emulated Hardware

Oracle VM VirtualBox virtualizes nearly all hardware of the host. Depending on a VM's configuration, the guest will see the following virtual hardware:

- **Input devices.** Oracle VM VirtualBox can emulate a standard PS/2 keyboard and mouse. These devices are supported by most guest OSes.

  In addition, Oracle VM VirtualBox can provide virtual USB input devices to avoid having to capture mouse and keyboard, as described in Section 1.8.2, "Capturing and Releasing Keyboard and Mouse".

- **Graphics.** The default Oracle VM VirtualBox graphics device for Windows guests is an SVGA device. For Linux guests, the default graphics device emulates a VMware SVGA graphics device. See Section 3.6.1, "Screen Tab".

  For legacy guest OSes, a VGA-compatible graphics device is available.

- **Storage.** Oracle VM VirtualBox emulates the most common types of hard disk controllers. See Section 5.1, "Hard Disk Controllers". Whereas supporting only one of these controllers would be enough for Oracle VM VirtualBox by itself, this multitude of storage adapters is required for compatibility with other hypervisors. Windows is very selective about its boot devices, and migrating VMs between hypervisors is very difficult or impossible if the storage controllers are different.

- **Networking.** See Section 6.1, "Virtual Networking Hardware".

- **USB.** Oracle VM VirtualBox emulates these types of USB host controllers: xHCI, EHCI, and OHCI. While xHCI handles all USB transfer speeds, some legacy guest OSes may not support xHCI. Note that for some legacy Windows guests, third party drivers must be installed for xHCI support.

  Legacy guest OSes typically support OHCI and EHCI. These two controllers are needed because OHCI only handles USB low-speed and full-speed devices (both USB 1.x and 2.0), while EHCI only handles high-speed devices (USB 2.0 only).

  The emulated USB controllers do not communicate directly with devices on the host. Instead they communicate with a virtual USB layer which abstracts the USB protocol and enables the use of remote USB devices.

- **Audio.** See Section 3.8, "Audio Settings".

3.4. General Settings

In the **Settings** window, under **General**, you can configure the most fundamental aspects of the virtual machine such as memory and essential hardware. The following tabs are available.

3.4.1. Basic Tab

In the **Basic** tab of the **General** settings category, you can find these settings:

- **Name:** The name of the the VM, as shown in the list of VMs in the main VirtualBox Manager window. Using this name, Oracle VM VirtualBox also saves the VM's configuration files. If you change the name, Oracle VM VirtualBox renames these files as well. As a result, you can only use characters which are allowed for file names on your host OS.

  Note that internally, Oracle VM VirtualBox uses unique identifiers (UUIDs) to identify virtual machines. You can display these using the `VBoxManage` commands.

- **Type:** The type of the guest OS for the VM. This is the same setting that is specified in the **New Virtual Machine** wizard. See Section 1.7, "Creating Your First Virtual Machine".

  Whereas the default settings of a newly created VM depend on the selected OS type, changing the type later has no effect on VM settings. This value is purely informational and decorative.

- **Version:** The version of the guest OS for the VM. This is the same setting that is specified in the **New Virtual Machine** wizard. See Section 1.7, "Creating Your First Virtual Machine".

3.4.2. Advanced Tab

The following settings are available in the **Advanced** tab:

- **Snapshot Folder:** By default, Oracle VM VirtualBox saves snapshot data together with your other Oracle VM VirtualBox configuration data. See Where Oracle VM VirtualBox Stores its Files. With this setting, you can specify any other folder for each VM.

- **Shared Clipboard:** You can select here whether the clipboard of the guest OS should be shared with that of your host. If you select **Bidirectional**, then Oracle VM VirtualBox will always make sure that both clipboards contain the same data. If you select **Host to Guest** or **Guest to Host**, then Oracle VM VirtualBox will only ever copy clipboard data in one direction.

  Clipboard sharing requires that the Oracle VM VirtualBox Guest Additions be installed. In such a case, this setting has no effect. See Chapter 4, *Guest Additions*.

  For security reasons, the shared clipboard is disabled by default. This setting can be changed at any time using the **Shared Clipboard** menu item in the **Devices** menu of the virtual machine.

- **Drag and Drop:** This setting enables support for drag and drop. Select an object, such as a file, from the host or guest and directly copy or open it on the guest or host. Multiple drag and drop modes for a VM enable restricting of access in either direction.

  For drag and drop to work the Guest Additions need to be installed on the guest.

  Note

  Drag and drop is disabled by default. This setting can be changed at any time using the **Drag and Drop** menu item in the **Devices** menu of the virtual machine.

  See Section 4.4, "Drag and Drop".

3.4.3. Description Tab

On the **Description** tab you can enter a description for your virtual machine. This has no effect on the functionality of the machine, but you may find this space useful to note down things such as the configuration of a virtual machine and the software that has been installed into it.

To insert a line break into the **Description** text field, press Shift+Enter.

3.4.4. Disk Encryption Tab

The **Disk Encryption** tab enables you to encrypt disks that are attached to the virtual machine.

To enable disk encryption, select the **Enable Disk Encryption** check box.

Settings are available to configure the cipher used for encryption and the encryption password.

3.5. System Settings

The **System** category groups various settings that are related to the basic hardware that is presented to the virtual machine.

Note

As the activation mechanism of Microsoft Windows is sensitive to hardware changes, if you are changing hardware settings for a Windows guest, some of these changes may trigger a request for another activation with Microsoft.

The following tabs are available.

3.5.1. Motherboard Tab

On the **Motherboard** tab, you can configure virtual hardware that would normally be on the motherboard of a real computer.

- **Base Memory:** Sets the amount of RAM that is allocated and given to the VM when it is running. The specified amount of memory will be requested from the host OS, so it must be available or made available as free memory on the host when attempting to start the VM and will not be available to the host while the VM is running. This is the same setting that was specified in the **New Virtual Machine** wizard, as described in Section 1.7, "Creating Your First Virtual Machine".

  Generally, it is possible to change the memory size after installing the guest OS. But you must not reduce the memory to an amount where the OS would no longer boot.

- **Boot Order:** Determines the order in which the guest OS will attempt to boot from the various virtual boot devices. Analogous to a real PC's BIOS setting, Oracle VM VirtualBox can tell a guest OS to start from the virtual floppy, the virtual CD/DVD drive, the virtual hard drive (each of these as defined by the other VM settings), the network, or none of these.

  If you select **Network**, the VM will attempt to boot from a network using the PXE mechanism. This needs to be configured in detail on the command line. See Section 7.8, "VBoxManage modifyvm".

- **Chipset:** You can select which chipset will be presented to the virtual machine. PIIX3 is the default chipset for most guests. For some guest OSes such as Mac OS X, the PIIX3 chipset is not well supported. As a result, Oracle VM VirtualBox supports an emulation of the ICH9 chipset, which supports PCI express, three PCI buses, PCI-to-PCI bridges and Message Signaled Interrupts (MSI). This enables modern OSes to address more PCI devices and no longer requires IRQ sharing. Using the ICH9 chipset it is also possible to configure up to 36 network cards, compared to a maximum of eight network adapters with PIIX3. Note that ICH9 support is experimental and not recommended for guest OSes which do not require it.

- **Pointing Device:** The default virtual pointing device for some guest OSes is the traditional PS/2 mouse. If set to **USB Tablet**, Oracle VM VirtualBox reports to the virtual machine that a USB tablet device is present and communicates mouse events to the virtual machine through this device. Another setting is **USB Multi-Touch Tablet**, which is suitable for guests running Windows 8 or later.

  Using the virtual USB tablet has the advantage that movements are reported in absolute coordinates, instead of as relative position changes. This enables Oracle VM VirtualBox to translate mouse events over the VM window into tablet events without having to "capture" the mouse in the guest as described in Section 1.8.2, "Capturing and Releasing Keyboard and Mouse". This makes using the VM less tedious even if Guest Additions are not installed.

- **Enable I/O APIC:** Advanced Programmable Interrupt Controllers (APICs) are an x86 hardware feature that have replaced Programmable Interrupt Controllers (PICs). With an I/O APIC, OSes can use more than 16 interrupt requests (IRQs) and therefore avoid IRQ sharing for improved reliability.

  Note

  Enabling the I/O APIC is *required*, especially for 64-bit Windows guest OSes. It is also required if you want to use more than one virtual CPU in a virtual machine.

  However, software support for I/O APICs has been unreliable with some OSes other than Windows. Also, the use of an I/O APIC slightly increases the overhead of virtualization and therefore slows down the guest OS a little.

> **Warning**
>
> All Windows OSes install different kernels, depending on whether an I/O APIC is available. As with ACPI, the I/O APIC therefore *must not be turned off after installation* of a Windows guest OS. Turning it on after installation will have no effect however.

- **Enable EFI:** Enables Extensible Firmware Interface (EFI), which replaces the legacy BIOS and may be useful for certain advanced use cases. See Section 3.14, "Alternative Firmware (EFI)".

- **Hardware Clock in UTC Time:** If selected, Oracle VM VirtualBox will report the system time in UTC format to the guest instead of the local (host) time. This affects how the virtual real-time clock (RTC) operates and may be useful for UNIX-like guest OSes, which typically expect the hardware clock to be set to UTC.

In addition, you can turn off the **Advanced Configuration and Power Interface (ACPI)** which Oracle VM VirtualBox presents to the guest OS by default.

ACPI is the current industry standard to allow OSes to recognize hardware, configure motherboards and other devices and manage power. As most computers contain this feature and Windows and Linux support ACPI, it is also enabled by default in Oracle VM VirtualBox. ACPI can only be turned off using the command line. See Section 7.8, "VBoxManage modifyvm".

> **Warning**
>
> All Windows OSes install different kernels, depending on whether ACPI is available. This means that ACPI *must not be turned off* after installation of a Windows guest OS. However, turning it on after installation will have no effect.

3.5.2. Processor Tab

On the **Processor** tab, you can configure settings for the CPU used by the virtual machine.

- **Processor(s):** Sets the number of virtual CPU cores the guest OSes can see. Oracle VM VirtualBox supports symmetrical multiprocessing (SMP) and can present up to 32 virtual CPU cores to each virtual machine.

  You should not configure virtual machines to use more CPU cores than are available physically. This includes real cores, with no hyperthreads.

- **Execution Cap:** Configures the CPU execution cap. This limits the amount of time a host CPU spends to emulate a virtual CPU. The default setting is 100%, meaning that there is no limitation. A setting of 50% implies a single virtual CPU can use up to 50% of a single host CPU. Note that limiting the execution time of the virtual CPUs may cause guest timing problems.

  A warning is displayed at the bottom of the Processor tab if an Execution Cap setting is made that may affect system performance.

- **Enable PAE/NX:** Determines whether the PAE and NX capabilities of the host CPU will be exposed to the virtual machine.

  PAE stands for Physical Address Extension. Normally, if enabled and supported by the OS, then even a 32-bit x86 CPU can access more than 4 GB of RAM. This is made possible by adding another 4 bits to memory addresses, so that with 36 bits, up to 64 GB can be addressed. Some OSes, such as Ubuntu Server, require PAE support from the CPU and cannot be run in a virtual machine without it.

- **Enable Nested VT-x/AMD-V**: Enables nested virtualization, with passthrough of hardware virtualization functions to the guest VM.

  This feature is available on host systems that use an AMD CPU. For Intel CPUs, the option is grayed out.

With virtual machines running modern server OSes, Oracle VM VirtualBox also supports CPU hot-plugging. For details, see CPU Hot-Plugging.

3.5.3. Acceleration Tab

On this tab, you can configure Oracle VM VirtualBox to use hardware virtualization extensions that your host CPU supports.

- **Paravirtualization Interface:** Oracle VM VirtualBox provides paravirtualization interfaces to improve time-keeping accuracy and performance of guest OSes. The options available are documented under the `--paravirtprovider` option in Section 7.8, "VBoxManage modifyvm". For further details on the paravirtualization providers, see Paravirtualization Providers.

- **Hardware Virtualization:** You can select for each virtual machine individually whether Oracle VM VirtualBox should use software or hardware virtualization.

  - **Enable VT-x/AMD-V:** Enables Intel VT-x and AMD-V hardware extensions if the host CPU supports them.

  - **Enable Nested Paging:** If the host CPU supports the nested paging (AMD-V) or EPT (Intel VT-x) features, then you can expect a significant performance increase by enabling nested paging in addition to hardware virtualization. For technical details, see Nested Paging and VPIDs. For Intel EPT security recommendations, see CVE-2018-3646.

  Advanced users may be interested in technical details about software versus hardware virtualization. See Hardware vs. Software Virtualization.

In most cases, the default settings on the **Acceleration** tab will work well. Oracle VM VirtualBox selects sensible defaults, depending on the OS that you selected when you created the virtual machine. In certain situations, however, you may want to change the preconfigured defaults.

3.6. Display Settings

The following tabs are available for configuring the display for a virtual machine.

3.6.1. Screen Tab

- **Video Memory:** Sets the size of the memory provided by the virtual graphics card available to the guest, in MB. As with the main memory, the specified amount will be allocated from the host's resident memory. Based on the amount of video memory, higher resolutions and color depths may be available.

  The GUI will show a warning if the amount of video memory is too small to be able to switch the VM into full screen mode. The minimum value depends on the number of virtual monitors, the screen resolution and the color depth of the host display as well as on the use of *3D acceleration* and *2D video acceleration*. A rough estimate is (*color depth* / 8) x *vertical pixels* x *horizontal pixels* x *number of screens* = *number of bytes*. Extra memory may be required if display acceleration is used.

- **Monitor Count:** With this setting, Oracle VM VirtualBox can provide more than one virtual monitor to a virtual machine. If a guest OS supports multiple attached monitors, Oracle VM VirtualBox can pretend that multiple virtual monitors are present. Up to eight such virtual monitors are supported.

The output of the multiple monitors are displayed on the host in multiple VM windows which are running side by side. However, in full screen and seamless mode, they use the available physical monitors attached to the host. As a result, for full screen and seamless modes to work with multiple monitors, you will need at least as many physical monitors as you have virtual monitors configured, or Oracle VM VirtualBox will report an error.

You can configure the relationship between guest and host monitors using the **View** menu by pressing Host key + Home when you are in full screen or seamless mode.

See also Known Limitations.

- **Scale Factor:** Enables scaling of the display size. For multiple monitor displays, you can set the scale factor for individual monitors, or globally for all of the monitors. Use the slider to select a scaling factor up to 200%.

  You can set a default scale factor for all VMs. Use the **Display** tab in the Global Settings dialogs.

- **Enable 3D Acceleration:** If a virtual machine has Guest Additions installed, you can select here whether the guest should support accelerated 3D graphics. See Section 4.5.1, "Hardware 3D Acceleration (OpenGL and Direct3D 8/9)".

- **Enable 2D Video Acceleration:** If a virtual machine with Microsoft Windows has Guest Additions installed, you can select here whether the guest should support accelerated 2D video graphics. See Section 4.5.2, "Hardware 2D Video Acceleration for Windows Guests".

- **Graphics Controller:** Specifies the graphics adapter type used by the guest VM. Note that you must install the Guest Additions on the guest VM to specify the VBoxSVGA or VMSVGA graphics controller. The following options are available:

  - **VBoxSVGA:** The default graphics controller for new VMs that use Windows 7 or later.

    This graphics controller improves performance and 3D support when compared to the legacy VBoxVGA option.

  - **VBoxVGA:** Use this graphics controller for legacy guest OSes. This is the default graphics controller for Windows versions before Windows 7 and for Oracle Solaris.

  - **VMSVGA:** Use this graphics controller to emulate a VMware SVGA graphics device. This is the default graphics controller for Linux guests.

  - **None:** Does not emulate a graphics adapter type.

3.6.2. Remote Display Tab

On the **Remote Display** tab, if the VirtualBox Remote Display Extension (VRDE) is installed, you can enable the VRDP server that is built into Oracle VM VirtualBox. This enables you to connect to the console of the virtual machine remotely with any standard RDP viewer, such as `mstsc.exe` that comes with Microsoft Windows. On Linux and Oracle Solaris systems you can use the standard open source `rdesktop` program. These features are described in Remote Display (VRDP Support).

- **Enable Server:** Select this check box and configure settings for the remote display connection.

3.6.3. Recording Tab

On the **Recording** tab you can enable video and audio recording for a virtual machine and change related settings. Note that these features can be enabled and disabled while a VM is running.

- **Enable Recording:** Select this check box and select a **Recording Mode** option.

- **Recording Mode:** You can choose to record video, audio, or both video and audio.

  Some settings on the **Recording** tab may be grayed out, depending on the **Recording Mode** setting.

- **File Path:** The file where the recording is saved.

- **Frame Size:** The video resolution of the recorded video, in pixels. The drop-down list enables you to select from common frame sizes.

- **Frame Rate:** Use the slider to set the maximum number of video frames per second (FPS) to record. Frames that have a higher frequency are skipped. Increasing this value reduces the number of skipped frames and increases the file size.

- **Quality:** Use the slider to set the the bit rate of the video in kilobits per second. Increasing this value improves the appearance of the video at the cost of an increased file size.

- **Audio Quality:** Use the slider to set the quality of the audio recording. Increasing this value improves the audio quality at the cost of an increased file size.

- **Screens:** For a multiple monitor display, you can select which screens to record video from.

As you adjust the video and audio recording settings, the approximate output file size for a five minute video is shown.

3.7. Storage Settings

The **Storage** category in the VM settings enables you to connect virtual hard disk, CD/DVD, and floppy images and drives to your virtual machine.

In a real PC, so-called *storage controllers* connect physical disk drives to the rest of the computer. Similarly, Oracle VM VirtualBox presents virtual storage controllers to a virtual machine. Under each controller, the virtual devices, such as hard disks, CD/DVD or floppy drives, attached to the controller are shown.

> **Note**
>
> This section gives a quick introduction to the Oracle VM VirtualBox storage settings. See Chapter 5, *Virtual Storage* for a full description of the available storage settings in Oracle VM VirtualBox.

If you have used the **Create VM** wizard to create a machine, you will normally see something like the following:

Figure 3.1 Storage Settings for a Virtual Machine
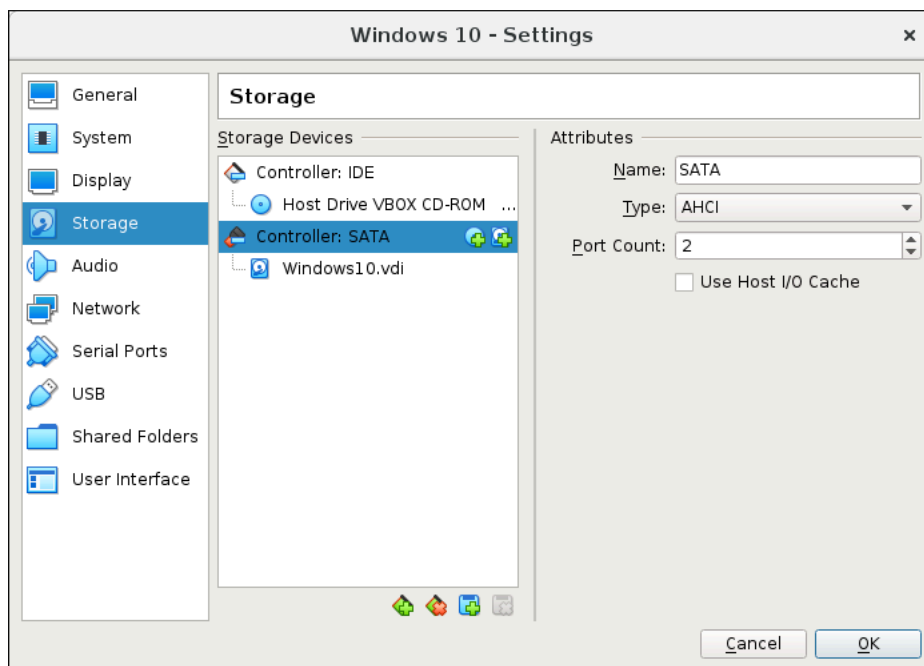
Depending on the guest OS type that you selected when you created the VM, a new VM includes the following storage devices:

- **IDE controller.** A virtual CD/DVD drive is attached to the secondary master port of the IDE controller.

- **SATA controller.** This is a modern type of storage controller for higher hard disk data throughput, to which the virtual hard disks are attached. Initially you will normally have one such virtual disk, but as shown in the previous screenshot, you can have more than one. Each is represented by a disk image file, such as a VDI file in this example.

If you created your VM with an older version of Oracle VM VirtualBox, the default storage layout may differ. You might then only have an IDE controller to which both the CD/DVD drive and the hard disks have been attached. This might also apply if you selected an older OS type when you created the VM. Since older OSes do not support SATA without additional drivers, Oracle VM VirtualBox will make sure that no such devices are present initially. See Section 5.1, "Hard Disk Controllers".

Oracle VM VirtualBox also provides a *floppy controller*. You cannot add devices other than floppy drives to this controller. Virtual floppy drives, like virtual CD/DVD drives, can be connected to either a host floppy drive, if you have one, or a disk image, which in this case must be in RAW format.

You can modify these media attachments freely. For example, if you wish to copy some files from another virtual disk that you created, you can connect that disk as a second hard disk, as in the above screenshot. You could also add a second virtual CD/DVD drive, or change where these items are attached. The following options are available:

- To **add another virtual hard disk, or a CD/DVD or floppy drive**, select the storage controller to which it should be added (IDE, SATA, SCSI, SAS, floppy controller) and then click the **Add Disk** button below the tree. You can then either select **Add CD/DVD Device** or **Add Hard Disk**. If you clicked on a floppy controller, you can add a floppy drive instead. Alternatively, right-click on the storage controller and select a menu item there.

  On the right part of the window, you can then set the following:

1. The **device slot** of the controller that the virtual disk is connected to. IDE controllers have four slots which have traditionally been called primary master, primary slave, secondary master, and secondary slave. By contrast, SATA and SCSI controllers offer you up to 30 slots for attaching virtual devices.

2. The **image file** to use.

   - For virtual hard disks, a button with a drop-down list appears on the right, offering you to either select a **virtual hard disk file** using a standard file dialog or to **create a new hard disk** (image file). The latter option displays the **Create New Disk** wizard, described in Section 1.7, "Creating Your First Virtual Machine".

     For virtual floppy drives, a dialog enables you to create and format a new floppy disk image automatically.

     For details on the image file types that are supported, see Section 5.2, "Disk Image Files (VDI, VMDK, VHD, HDD)".

   - For virtual CD/DVD drives, the image files will typically be in the standard ISO format instead. Most commonly, you will select this option when installing an OS from an ISO file that you have obtained from the Internet. For example, most Linux distributions are available in this way.

     For virtual CD/DVD drives, the following additional options are available:

     - If you select **Host Drive** from the list, then the physical device of the host computer is connected to the VM, so that the guest OS can read from and write to your physical device. This is, for instance, useful if you want to install Windows from a real installation CD. In this case, select your host drive from the drop-down list presented.

       If you want to write, or burn, CDs or DVDs using the host drive, you need to also enable the **Passthrough** option. See Section 5.9, "CD/DVD Support".

     - If you select **Remove Disk from Virtual Drive**, Oracle VM VirtualBox will present an empty CD/DVD drive to the guest into which no media has been inserted.

- To **remove an attachment**, either select it and click on the **Remove** icon at the bottom, or right-click on it and select the menu item.

Removable media, such as CD/DVDs and floppies, can be changed while the guest is running. Since the **Settings** dialog is not available at that time, you can also access these settings from the **Devices** menu of your virtual machine window.

3.8. Audio Settings

The **Audio** section in a virtual machine's **Settings** window determines whether the VM will detect a connected sound card, and if the audio output should be played on the host system.

To enable audio for a guest, select the **Enable Audio** check box. The following settings are available:

- **Host Audio Driver:** The audio driver that Oracle VM VirtualBox uses on the host. On a Linux host, depending on your host configuration, you can select between the OSS, ALSA, or the PulseAudio subsystem. On newer Linux distributions, the PulseAudio subsystem is preferred.

  Only OSS is supported on Oracle Solaris hosts. The Oracle Solaris Audio audio backend is no longer supported on Oracle Solaris hosts.

- **Audio Controller:** You can choose between the emulation of an Intel AC'97 controller, an Intel HD Audio controller, or a SoundBlaster 16 card.

- **Enable Audio Output:** Enables audio output only for the VM.

- **Enable Audio Input:** Enables audio input only for the VM.

3.9. Network Settings

The **Network** section in a virtual machine's **Settings** window enables you to configure how Oracle VM VirtualBox presents virtual network cards to your VM, and how they operate.

When you first create a virtual machine, Oracle VM VirtualBox by default enables one virtual network card and selects the Network Address Translation (NAT) mode for it. This way the guest can connect to the outside world using the host's networking and the outside world can connect to services on the guest which you choose to make visible outside of the virtual machine.

This default setup is good for the majority of Oracle VM VirtualBox users. However, Oracle VM VirtualBox is extremely flexible in how it can virtualize networking. It supports many virtual network cards per virtual machine. The first four virtual network cards can be configured in detail in the VirtualBox Manager window. Additional network cards can be configured using the `VBoxManage` command.

Many networking options are available. See Chapter 6, *Virtual Networking* for more information.

3.10. Serial Ports

Oracle VM VirtualBox supports the use of virtual serial ports in a virtual machine.

Ever since the original IBM PC, personal computers have been equipped with one or two serial ports, also called COM ports by DOS and Windows. Serial ports were commonly used with modems, and some computer mice used to be connected to serial ports before USB became commonplace.

While serial ports are no longer as common as they used to be, there are still some important uses left for them. For example, serial ports can be used to set up a primitive network over a null-modem cable, in case Ethernet is not available. Also, serial ports are indispensable for system programmers needing to do kernel debugging, since kernel debugging software usually interacts with developers over a serial port. With virtual serial ports, system programmers can do kernel debugging on a virtual machine instead of needing a real computer to connect to.

If a virtual serial port is enabled, the guest OS sees a standard 16550A compatible UART device. Other UART types can be configured using the `VBoxManage modifyvm` command. Both receiving and transmitting data is supported. How this virtual serial port is then connected to the host is configurable, and the details depend on your host OS.

You can use either the Settings tabs or the `VBoxManage` command to set up virtual serial ports. For the latter, see Section 7.8, "VBoxManage modifyvm" for information on the `--uart`, `--uartmode` and `--uarttype` options.

You can configure up to four virtual serial ports per virtual machine. For each device, you must set the following:

1. **Port Number:** This determines the serial port that the virtual machine should see. For best results, use the traditional values as follows:

   - COM1: I/O base 0x3F8, IRQ 4

   - COM2: I/O base 0x2F8, IRQ 3

- COM3: I/O base 0x3E8, IRQ 4

- COM4: I/O base 0x2E8, IRQ 3

You can also configure a user-defined serial port. Enter an I/O base address and interrupt (IRQ).

2. **Port Mode:** What the virtual port is connected to. For each virtual serial port, you have the following options:

- **Disconnected:** The guest will see the device, but it will behave as if no cable had been connected to it.

- **Host Device:** Connects the virtual serial port to a physical serial port on your host. On a Windows host, this will be a name like `COM1`. On Linux or Oracle Solaris hosts, it will be a device node like `/dev/ttyS0`. Oracle VM VirtualBox will then simply redirect all data received from and sent to the virtual serial port to the physical device.

- **Host Pipe:** Configure Oracle VM VirtualBox to connect the virtual serial port to a software pipe on the host. This depends on your host OS, as follows:

  - On a Windows host, data will be sent and received through a named pipe. The pipe name must be in the format `\\.\pipe\`*name* where *name* should identify the virtual machine but may be freely chosen.

  - On a Mac, Linux, or Oracle Solaris host, a local domain socket is used instead. The socket filename must be chosen such that the user running Oracle VM VirtualBox has sufficient privileges to create and write to it. The `/tmp` directory is often a good candidate.

    On Linux there are various tools which can connect to a local domain socket or create one in server mode. The most flexible tool is `socat` and is available as part of many distributions.

In this case, you can configure whether Oracle VM VirtualBox should create the named pipe, or the local domain socket non-Windows hosts, itself or whether Oracle VM VirtualBox should assume that

the pipe or socket exists already. With the `VBoxManage` command-line options, this is referred to as server mode or client mode, respectively.

For a direct connection between two virtual machines, corresponding to a null-modem cable, simply configure one VM to create a pipe or socket and another to attach to it.

- **Raw File:** Send the virtual serial port output to a file. This option is very useful for capturing diagnostic output from a guest. Any file may be used for this purpose, as long as the user running Oracle VM VirtualBox has sufficient privileges to create and write to the file.

- **TCP Socket:** Useful for forwarding serial traffic over TCP/IP, acting as a server, or it can act as a TCP client connecting to other servers. This option enables a remote machine to directly connect to the guest's serial port using TCP.

  - **TCP Server:** Deselect the **Connect to Existing Pipe/Socket** check box and specify the port number in the **Path/Address** field. This is typically 23 or 2023. Note that on UNIX-like systems you will have to use a port a number greater than 1024 for regular users.

    The client can use software such as `PuTTY` or the `telnet` command line tool to access the TCP Server.

  - **TCP Client:** To create a virtual null-modem cable over the Internet or LAN, the other side can connect using TCP by specifying $hostname:port$ in the **Path/Address** field. The TCP socket will act in client mode if you select the **Connect to Existing Pipe/Socket** check box.

Up to four serial ports can be configured per virtual machine, but you can pick any port numbers out of the above. However, serial ports cannot reliably share interrupts. If both ports are to be used at the same time, they must use different interrupt levels, for example COM1 and COM2, but not COM1 and COM3.

3.11. USB Support

3.11.1. USB Settings

The **USB** section in a virtual machine's **Settings** window enables you to configure Oracle VM VirtualBox's sophisticated USB support.

Oracle VM VirtualBox can enable virtual machines to access the USB devices on your host directly. To achieve this, Oracle VM VirtualBox presents the guest OS with a virtual USB controller. As soon as the guest system starts using a USB device, it will appear as unavailable on the host.

Note

- Be careful with USB devices that are currently in use on the host. For example, if you allow your guest to connect to your USB hard disk that is currently mounted on the host, when the guest is activated, it will be disconnected from the host without a proper shutdown. This may cause data loss.

- Oracle Solaris hosts have a few known limitations regarding USB support. See Known Limitations.

In addition to allowing a guest access to your local USB devices, Oracle VM VirtualBox even enables your guests to connect to remote USB devices by use of the VirtualBox Remote Desktop Extension (VRDE). See Remote USB.

To enable USB for a VM, select the **Enable USB Controller** check box. The following settings are available:

- **USB Controller:** Selects a controller with the specified level of USB support, as follows:

  - OHCI for USB 1.1

  - EHCI for USB 2.0. This also enables OHCI.

  - xHCI for USB 3.0. This supports all USB speeds.

    > **Note**
    >
    > The xHCI and EHCI controllers are shipped as an Oracle VM VirtualBox extension package, which must be installed separately. See Section 1.5, "Installing Oracle VM VirtualBox and Extension Packs".

- **USB Device Filters:** When USB support is enabled for a VM, you can determine in detail which devices will be automatically attached to the guest. For this, you can create filters by specifying certain properties of the USB device. USB devices with a matching filter will be automatically passed to the guest once they are attached to the host. USB devices without a matching filter can be passed manually to the guest, for example by using the **Devices**, **USB** menu.

  Clicking on the **+** button to the right of the **USB Device Filters** window creates a new filter. You can give the filter a name, for later reference, and specify the filter criteria. The more criteria you specify, the more precisely devices will be selected. For instance, if you specify only a vendor ID of 046d, all devices produced by Logitech will be available to the guest. If you fill in all fields, on the other hand, the filter will only apply to a particular device model from a particular vendor, and not even to other devices of the same type with a different revision and serial number.

  In detail, the following criteria are available:

  - **Vendor and Product ID.** With USB, each vendor of USB products carries an identification number that is unique world-wide, called the *vendor ID*. Similarly, each line of products is assigned a *product ID* number. Both numbers are commonly written in hexadecimal, and a colon separates the vendor from the product ID. For example, `046d:c016` stands for Logitech as a vendor, and the M-UV69a Optical Wheel Mouse product.

    Alternatively, you can also specify **Manufacturer** and **Product** by name.

    To list all the USB devices that are connected to your host machine with their respective vendor IDs and product IDs, use the following command:

    ```
    VBoxManage list usbhost
    ```

    On Windows, you can also see all USB devices that are attached to your system in the Device Manager. On Linux, you can use the `lsusb` command.

  - **Serial Number.** While vendor ID and product ID are quite specific to identify USB devices, if you have two identical devices of the same brand and product line, you will also need their serial numbers to filter them out correctly.

- **Remote.** This setting specifies whether the device will be local only, remote only, such as over VRDP, or either.

On a Windows host, you will need to unplug and reconnect a USB device to use it after creating a filter for it.

As an example, you could create a new USB filter and specify a vendor ID of 046d for Logitech, Inc, a manufacturer index of 1, and "not remote". Then any USB devices on the host system produced by Logitech, Inc with a manufacturer index of 1 will be visible to the guest system.

Several filters can select a single device. For example, a filter which selects all Logitech devices, and one which selects a particular webcam.

You can deactivate filters without deleting them by deselecting the check box next to the filter name.

3.11.2. Implementation Notes for Windows and Linux Hosts

On Windows hosts, a kernel mode device driver provides USB proxy support. It implements both a USB monitor, which enables Oracle VM VirtualBox to capture devices when they are plugged in, and a USB device driver to claim USB devices for a particular virtual machine. System reboots are not necessary after installing the driver. Also, you do not need to replug devices for Oracle VM VirtualBox to claim them.

On supported Linux hosts, Oracle VM VirtualBox accesses USB devices through special files in the file system. When Oracle VM VirtualBox is installed, these are made available to all users in the `vboxusers` system group. In order to be able to access USB from guest systems, make sure that you are a member of this group.

On some legacy Linux hosts, USB devices are accessed using the `usbfs` file system. Therefore, the user executing Oracle VM VirtualBox needs read and write permission to the USB file system. Most distributions provide a group, such as `usbusers`, which the Oracle VM VirtualBox user needs to be added to. Also, Oracle VM VirtualBox can only proxy to virtual machines USB devices which are not claimed by a Linux host USB driver. The `Driver=` entry in `/proc/bus/usb/devices` will show you which devices are currently claimed. See also USB Not Working for details about `usbfs`.

3.12. Shared Folders

Shared folders enable you to easily exchange data between a virtual machine and your host. This feature requires that the Oracle VM VirtualBox Guest Additions be installed in a virtual machine and is described in detail in Section 4.3, "Shared Folders".

3.13. User Interface

The **User Interface** section enables you to change certain aspects of the user interface of this VM.

- **Menu Bar:** This widget enables you to disable menus by clicking on the menu to release it, menu entries by deselecting the check box of the entry to disable it and the complete menu bar by deselecting the rightmost check box.

- **Mini ToolBar:** In full screen or seamless mode, Oracle VM VirtualBox can display a small toolbar that contains some of the items that are normally available from the virtual machine's menu bar. This toolbar reduces itself to a small gray line unless you move the mouse over it. With the toolbar, you can return from full screen or seamless mode, control machine execution or enable certain devices. If you do not want to see the toolbar, disable this setting.

The second setting enables you to show the toolbar at the top of the screen, instead of showing it at the bottom.

- **Status Bar:** This widget enables you to disable icons on the status bar by deselecting the check box of an icon to disable it, to rearrange icons by dragging and dropping the icon, and to disable the complete status bar by deselecting the leftmost check box.

3.14. Alternative Firmware (EFI)

Oracle VM VirtualBox includes experimental support for the Extensible Firmware Interface (EFI), which is an industry standard intended to replace the legacy BIOS as the primary interface for bootstrapping computers and certain system services later.

By default, Oracle VM VirtualBox uses the BIOS firmware for virtual machines. To use EFI for a given virtual machine, you can enable EFI in the machine's **Settings** dialog. See Section 3.5.1, "Motherboard Tab". Alternatively, use the `VBoxManage` command line interface as follows:

```
VBoxManage modifyvm "VM name" --firmware efi
```

To switch back to using the BIOS:

```
VBoxManage modifyvm "VM name" --firmware bios
```

One notable user of EFI is Apple Mac OS X. More recent Linux versions and Windows releases, starting with Vista, also offer special versions that can be booted using EFI.

Another possible use of EFI in Oracle VM VirtualBox is development and testing of EFI applications, without booting any OS.

Note that the Oracle VM VirtualBox EFI support is experimental and will be enhanced as EFI matures and becomes more widespread. Mac OS X, Linux, and newer Windows guests are known to work fine. Windows 7 guests are unable to boot with the Oracle VM VirtualBox EFI implementation.

3.14.1. Video Modes in EFI

EFI provides two distinct video interfaces: GOP (Graphics Output Protocol) and UGA (Universal Graphics Adapter). Modern OSes, such as Mac OS X, generally use GOP, while some older ones still use UGA. Oracle VM VirtualBox provides a configuration option to control the graphics resolution for both interfaces, making the difference mostly irrelevant for users.

The default resolution is 1024x768. To select a graphics resolution for EFI, use the following `VBoxManage` command:

```
VBoxManage setextradata "VM name" VBoxInternal2/EfiGraphicsResolution HxV
```

Determine the horizontal resolution H and the vertical resolution V from the following list of default resolutions:

VGA	640x480, 32bpp, 4:3
SVGA	800x600, 32bpp, 4:3
XGA	1024x768, 32bpp, 4:3
XGA+	1152x864, 32bpp, 4:3

HD	1280x720, 32bpp, 16:9
WXGA	1280x800, 32bpp, 16:10
SXGA	1280x1024, 32bpp, 5:4
SXGA+	1400x1050, 32bpp, 4:3
WXGA+	1440x900, 32bpp, 16:10
HD+	1600x900, 32bpp, 16:9
UXGA	1600x1200, 32bpp, 4:3
WSXGA+	1680x1050, 32bpp, 16:10
Full HD	1920x1080, 32bpp, 16:9
WUXGA	1920x1200, 32bpp, 16:10
DCI 2K	2048x1080, 32bpp, 19:10
Full HD+	2160x1440, 32bpp, 3:2
Unnamed	2304x1440, 32bpp, 16:10
QHD	2560x1440, 32bpp, 16:9
WQXGA	2560x1600, 32bpp, 16:10
QWXGA+	2880x1800, 32bpp, 16:10
QHD+	3200x1800, 32bpp, 16:9
WQSXGA	3200x2048, 32bpp, 16:10
4K UHD	3840x2160, 32bpp, 16:9
WQUXGA	3840x2400, 32bpp, 16:10
DCI 4K	4096x2160, 32bpp, 19:10
HXGA	4096x3072, 32bpp, 4:3
UHD+	5120x2880, 32bpp, 16:9
WHXGA	5120x3200, 32bpp, 16:10
WHSXGA	6400x4096, 32bpp, 16:10
HUXGA	6400x4800, 32bpp, 4:3
8K UHD2	7680x4320, 32bpp, 16:9

If this list of default resolution does not cover your needs, see Custom VESA Resolutions. Note that the color depth value specified in a custom video mode must be specified. Color depths of 8, 16, 24, and 32 are accepted. EFI assumes a color depth of 32 by default.

The EFI default video resolution settings can only be changed when the VM is powered off.

3.14.2. Specifying Boot Arguments

It is currently not possible to manipulate EFI variables from within a running guest. For example, setting the `boot-args` variable by running the `nvram` tool in a Mac OS X guest will not work. As an alternative method, `VBoxInternal2/EfiBootArgs` extradata can be passed to a VM in order to set the `boot-args` variable. To change the `boot-args` EFI variable, use the following command:

```
VBoxManage setextradata "VM name" VBoxInternal2/EfiBootArgs <value>
```

Chapter 4 Guest Additions

The previous chapter covered getting started with Oracle VM VirtualBox and installing operating systems in a virtual machine. For any serious and interactive use, the Oracle VM VirtualBox Guest Additions will make your life much easier by providing closer integration between host and guest and improving the interactive performance of guest systems. This chapter describes the Guest Additions in detail.

4.1. Introduction to Guest Additions

As mentioned in Section 1.2, "Some Terminology", the Guest Additions are designed to be installed *inside* a virtual machine after the guest operating system has been installed. They consist of device drivers and system applications that optimize the guest operating system for better performance and usability. See Section 3.1, "Supported Guest Operating Systems" for details on what guest operating systems are fully supported with Guest Additions by Oracle VM VirtualBox.

The Oracle VM VirtualBox Guest Additions for all supported guest operating systems are provided as a single CD-ROM image file which is called `VBoxGuestAdditions.iso`. This image file is located in the installation directory of Oracle VM VirtualBox. To install the Guest Additions for a particular VM, you mount this ISO file in your VM as a virtual CD-ROM and install from there.

The Guest Additions offer the following features:

- **Mouse pointer integration**. To overcome the limitations for mouse support described in Section 1.8.2, "Capturing and Releasing Keyboard and Mouse", this feature provides you with seamless mouse support. You will only have one mouse pointer and pressing the Host key is no longer required to *free* the mouse from being captured by the guest OS. To make this work, a special mouse driver is installed in the guest that communicates with the physical mouse driver on your host and moves the guest mouse pointer accordingly.

- **Shared folders.** These provide an easy way to exchange files between the host and the guest. Much like ordinary Windows network shares, you can tell Oracle VM VirtualBox to treat a certain host directory as a shared folder, and Oracle VM VirtualBox will make it available to the guest operating system as a network share, irrespective of whether the guest actually has a network. See Section 4.3, "Shared Folders".

- **Better video support.** While the virtual graphics card which Oracle VM VirtualBox emulates for any guest operating system provides all the basic features, the custom video drivers that are installed with the Guest Additions provide you with extra high and non-standard video modes, as well as accelerated video performance.

  In addition, with Windows, Linux, and Oracle Solaris guests, you can resize the virtual machine's window if the Guest Additions are installed. The video resolution in the guest will be automatically adjusted, as if you had manually entered an arbitrary resolution in the guest's **Display** settings. See Section 1.8.5, "Resizing the Machine's Window".

  If the Guest Additions are installed, 3D graphics and 2D video for guest applications can be accelerated. See Section 4.5, "Hardware-Accelerated Graphics".

- **Seamless windows.** With this feature, the individual windows that are displayed on the desktop of the virtual machine can be mapped on the host's desktop, as if the underlying application was actually running on the host. See Section 4.6, "Seamless Windows".

- **Generic host/guest communication channels.** The Guest Additions enable you to control and monitor guest execution. The *guest properties* provide a generic string-based mechanism to exchange data bits between a guest and a host, some of which have special meanings for controlling and monitoring the guest. See Section 4.7, "Guest Properties".

Additionally, applications can be started in a guest from the host. See Section 4.9, "Guest Control of Applications".

- **Time synchronization.** With the Guest Additions installed, Oracle VM VirtualBox can ensure that the guest's system time is better synchronized with that of the host.

  For various reasons, the time in the guest might run at a slightly different rate than the time on the host. The host could be receiving updates through NTP and its own time might not run linearly. A VM could also be paused, which stops the flow of time in the guest for a shorter or longer period of time. When the wall clock time between the guest and host only differs slightly, the time synchronization service attempts to gradually and smoothly adjust the guest time in small increments to either catch up or lose time. When the difference is too great, for example if a VM paused for hours or restored from saved state, the guest time is changed immediately, without a gradual adjustment.

  The Guest Additions will resynchronize the time regularly. See Tuning the Guest Additions Time Synchronization Parameters for how to configure the parameters of the time synchronization mechanism.

- **Shared clipboard.** With the Guest Additions installed, the clipboard of the guest operating system can optionally be shared with your host operating system. See Section 3.4, "General Settings".

- **Automated logins.** Also called credentials passing. See Automated Guest Logins.

Each version of Oracle VM VirtualBox, even minor releases, ship with their own version of the Guest Additions. While the interfaces through which the Oracle VM VirtualBox core communicates with the Guest Additions are kept stable so that Guest Additions already installed in a VM should continue to work when Oracle VM VirtualBox is upgraded on the host, for best results, it is recommended to keep the Guest Additions at the same version.

The Windows and Linux Guest Additions therefore check automatically whether they have to be updated. If the host is running a newer Oracle VM VirtualBox version than the Guest Additions, a notification with further instructions is displayed in the guest.

To disable this update check for the Guest Additions of a given virtual machine, set the value of its `/VirtualBox/GuestAdd/CheckHostVersion` guest property to `0`. See Section 4.7, "Guest Properties".

4.2. Installing and Maintaining Guest Additions

Guest Additions are available for virtual machines running Windows, Linux, Oracle Solaris, or OS/2. The following sections describe the specifics of each variant in detail.

4.2.1. Guest Additions for Windows

The Oracle VM VirtualBox Windows Guest Additions are designed to be installed in a virtual machine running a Windows operating system. The following versions of Windows guests are supported:

- Microsoft Windows NT 4.0 (any service pack)

- Microsoft Windows 2000 (any service pack)

- Microsoft Windows XP (any service pack)

- Microsoft Windows Server 2003 (any service pack)

- Microsoft Windows Server 2008

- Microsoft Windows Vista (all editions)

- Microsoft Windows 7 (all editions)

- Microsoft Windows 8 (all editions)

- Microsoft Windows 10 RTM build 10240

- Microsoft Windows Server 2012

4.2.1.1. Installing the Windows Guest Additions

In the **Devices** menu in the virtual machine's menu bar, Oracle VM VirtualBox has a menu item **Insert Guest Additions CD Image**, which mounts the Guest Additions ISO file inside your virtual machine. A Windows guest should then automatically start the Guest Additions installer, which installs the Guest Additions on your Windows guest.

For other guest operating systems, or if automatic start of software on a CD is disabled, you need to do a manual start of the installer.

> **Note**
>
> For the basic Direct3D acceleration to work in a Windows guest, you have to install the WDDM video driver available for Windows Vista or later.
>
> For Windows 8 and later, only the WDDM Direct3D video driver is available. For basic Direct3D acceleration to work in Windows XP guests, you have to install the Guest Additions in Safe Mode. See Known Limitations for details.

If you prefer to mount the Guest Additions manually, you can perform the following steps:

1. Start the virtual machine in which you have installed Windows.

2. Select **Mount CD/DVD-ROM** from the **Devices** menu in the virtual machine's menu bar and then **CD/DVD-ROM Image**. This displays the Virtual Media Manager, described in Section 5.3, "The Virtual Media Manager".

3. In the Virtual Media Manager, click **Add** and browse your host file system for the `VBoxGuestAdditions.iso` file.

   - On a Windows host, this file is in the Oracle VM VirtualBox installation directory, usually in `C:\Program files\Oracle\VirtualBox`.

   - On Mac OS X hosts, this file is in the application bundle of Oracle VM VirtualBox. Right-click on the Oracle VM VirtualBox icon in Finder and choose **Show Package Contents**. The file is located in the `Contents/MacOS` folder.

   - On a Linux host, this file is in the `additions` folder where you installed Oracle VM VirtualBox, usually `/opt/VirtualBox/`.

   - On Oracle Solaris hosts, this file is in the `additions` folder where you installed Oracle VM VirtualBox, usually `/opt/VirtualBox`.

4. In the Virtual Media Manager, select the ISO file and click **Select** button. This mounts the ISO file and presents it to your Windows guest as a CD-ROM.

Unless you have the Autostart feature disabled in your Windows guest, Windows will now autostart the Oracle VM VirtualBox Guest Additions installation program from the Additions ISO. If the Autostart feature has been turned off, choose `VBoxWindowsAdditions.exe` from the CD/DVD drive inside the guest to start the installer.

The installer will add several device drivers to the Windows driver database and then invoke the hardware detection wizard.

Depending on your configuration, it might display warnings that the drivers are not digitally signed. You must confirm these in order to continue the installation and properly install the Additions.

After installation, reboot your guest operating system to activate the Additions.

4.2.1.2. Updating the Windows Guest Additions

Windows Guest Additions can be updated by running the installation program again. This replaces the previous Additions drivers with updated versions.

Alternatively, you can also open the Windows Device Manager and select **Update Driver...** for the following devices:

- Oracle VM VirtualBox Graphics Adapter

- Oracle VM VirtualBox System Device

For each, choose the option to provide your own driver, click **Have Disk** and navigate to the CD-ROM drive with the Guest Additions.

4.2.1.3. Unattended Installation

To avoid popups when performing an unattended installation of the Oracle VM VirtualBox Guest Additions, the code signing certificates used to sign the drivers needs to be installed in the correct certificate stores on the guest operating system. Failure to do this will cause a typical Windows installation to display multiple dialogs asking whether you want to install a particular driver.

> **Note**
>
> On some Windows versions, such as Windows 2000 and Windows XP, the user intervention popups mentioned above are always displayed, even after importing the Oracle certificates.

Installing the code signing certificates on a Windows guest can be done automatically. Use the `VBoxCertUtil.exe` utility from the `cert` folder on the Guest Additions installation CD.

Use the following steps:

1. Log in as Administrator on the guest.

2. Mount the Oracle VM VirtualBox Guest Additions .ISO.

3. Open a command line window on the guest and change to the `cert` folder on the Oracle VM VirtualBox Guest Additions CD.

4. Run the following command:

   ```
   VBoxCertUtil.exe add-trusted-publisher vbox*.cer --root vbox*.cer
   ```

   This command installs the certificates to the certificate store. When installing the same certificate more than once, an appropriate error will be displayed.

To allow for completely unattended guest installations, you can specify a command line parameter to the install launcher:

```
VBoxWindowsAdditions.exe /S
```

This automatically installs the right files and drivers for the corresponding platform, either 32-bit or 64-bit.

> **Note**
>
> By default on an unattended installation on a Vista or Windows 7 guest, there will be the XPDM graphics driver installed. This graphics driver does not support Windows Aero / Direct3D on the guest. Instead, the WDDM graphics driver needs to be installed. To select this driver by default, add the command line parameter `/with_wddm` when invoking the Windows Guest Additions installer. This is only required for Vista and Windows 7.

> **Note**
>
> For Windows Aero to run correctly on a guest, the guest's VRAM size needs to be configured to at least 128 MB.

For more options regarding unattended guest installations, consult the command line help by using the command:

```
VBoxWindowsAdditions.exe /?
```

4.2.1.4. Manual File Extraction

If you would like to install the files and drivers manually, you can extract the files from the Windows Guest Additions setup as follows:

```
VBoxWindowsAdditions.exe /extract
```

To explicitly extract the Windows Guest Additions for another platform than the current running one, such as 64-bit files on a 32-bit system, you must use the appropriate platform installer. Use `VBoxWindowsAdditions-x86.exe` or `VBoxWindowsAdditions-amd64.exe` with the `/extract` parameter.

4.2.2. Guest Additions for Linux

Like the Windows Guest Additions, the Oracle VM VirtualBox Guest Additions for Linux are a set of device drivers and system applications which may be installed in the guest operating system.

The following Linux distributions are officially supported:

- Oracle Linux as of version 5, including UEK kernels

- Fedora as of Fedora Core 4

- Red Hat Enterprise Linux as of version 3

- SUSE and openSUSE Linux as of version 9

- Ubuntu as of version 5.10

Many other distributions are known to work with the Guest Additions.

The version of the Linux kernel supplied by default in SUSE and openSUSE 10.2, Ubuntu 6.10 (all versions) and Ubuntu 6.06 (server edition) contains a bug which can cause it to crash during startup when it is run in a virtual machine. The Guest Additions work in those distributions.

Note that some Linux distributions already come with all or part of the Oracle VM VirtualBox Guest Additions. You may choose to keep the distribution's version of the Guest Additions but these are often not up to date and limited in functionality, so we recommend replacing them with the Guest Additions that come with Oracle VM VirtualBox. The Oracle VM VirtualBox Linux Guest Additions installer tries to detect an existing installation and replace them but depending on how the distribution integrates the Guest

Additions, this may require some manual interaction. It is highly recommended to take a snapshot of the virtual machine before replacing preinstalled Guest Additions.

4.2.2.1. Installing the Linux Guest Additions

The Oracle VM VirtualBox Guest Additions for Linux are provided on the same virtual CD-ROM file as the Guest Additions for Windows. See Section 4.2.1.1, "Installing the Windows Guest Additions". They also come with an installation program that guides you through the setup process. However, due to the significant differences between Linux distributions, installation may be slightly more complex when compared to Windows.

Installation generally involves the following steps:

1. Before installing the Guest Additions, you prepare your guest system for building external kernel modules. This works as described in Section 2.3.2, "The Oracle VM VirtualBox Kernel Modules", except that this step must be performed in your Linux *guest* instead of on a Linux host system.

    If you suspect that something has gone wrong, check that your guest is set up correctly and run the following command as root:

    ```
    rcvboxadd setup
    ```

2. Insert the `VBoxGuestAdditions.iso` CD file into your Linux guest's virtual CD-ROM drive, as described for a Windows guest in Section 4.2.1.1, "Installing the Windows Guest Additions".

3. Change to the directory where your CD-ROM drive is mounted and run the following command as root:

    ```
    sh ./VBoxLinuxAdditions.run
    ```

4.2.2.2. Graphics and Mouse Integration

In Linux and Oracle Solaris guests, Oracle VM VirtualBox graphics and mouse integration goes through the X Window System. Oracle VM VirtualBox can use the X.Org variant of the system, or XFree86 version 4.3 which is identical to the first X.Org release. During the installation process, the X.Org display server will be set up to use the graphics and mouse drivers which come with the Guest Additions.

After installing the Guest Additions into a fresh installation of a supported Linux distribution or Oracle Solaris system, many unsupported systems will work correctly too, the guest's graphics mode will change to fit the size of the Oracle VM VirtualBox window on the host when it is resized. You can also ask the guest system to switch to a particular resolution by sending a video mode hint using the `VBoxManage` tool.

Multiple guest monitors are supported in guests using the X.Org server version 1.3, which is part of release 7.3 of the X Window System version 11, or a later version. The layout of the guest screens can be adjusted as needed using the tools which come with the guest operating system.

If you want to understand more about the details of how the X.Org drivers are set up, in particular if you wish to use them in a setting which our installer does not handle correctly, see Guest Graphics and Mouse Driver Setup in Depth.

4.2.2.3. Updating the Linux Guest Additions

The Guest Additions can simply be updated by going through the installation procedure again with an updated CD-ROM image. This will replace the drivers with updated versions. You should reboot after updating the Guest Additions.

4.2.2.4. Uninstalling the Linux Guest Additions

If you have a version of the Guest Additions installed on your virtual machine and wish to remove it without installing new ones, you can do so by inserting the Guest Additions CD image into the virtual CD-ROM

drive as described above. Then run the installer for the current Guest Additions with the `uninstall` parameter from the path that the CD image is mounted on in the guest, as follows:

```
sh ./VBoxLinuxAdditions.run uninstall
```

While this will normally work without issues, you may need to do some manual cleanup of the guest in some cases, especially of the XFree86Config or xorg.conf file. In particular, if the Additions version installed or the guest operating system were very old, or if you made your own changes to the Guest Additions setup after you installed them.

You can uninstall the Additions as follows:

```
/opt/VBoxGuestAdditions-version/uninstall.sh
```

Replace `/opt/VBoxGuestAdditions-version` with the correct Guest Additions installation directory.

4.2.3. Guest Additions for Oracle Solaris

Like the Windows Guest Additions, the Oracle VM VirtualBox Guest Additions for Oracle Solaris take the form of a set of device drivers and system applications which may be installed in the guest operating system.

The following Oracle Solaris distributions are officially supported:

- Oracle Solaris 11, including Oracle Solaris 11 Express

- Oracle Solaris 10 4/08 and later

Other distributions may work if they are based on comparable software releases.

4.2.3.1. Installing the Oracle Solaris Guest Additions

The Oracle VM VirtualBox Guest Additions for Oracle Solaris are provided on the same ISO CD-ROM as the Additions for Windows and Linux. They come with an installation program that guides you through the setup process.

Installation involves the following steps:

1. Mount the `VBoxGuestAdditions.iso` file as your Oracle Solaris guest's virtual CD-ROM drive, exactly the same way as described for a Windows guest in Section 4.2.1.1, "Installing the Windows Guest Additions".

    If the CD-ROM drive on the guest does not get mounted, as seen with some versions of Oracle Solaris 10, run the following command as root:

    ```
    svcadm restart volfs
    ```

2. Change to the directory where your CD-ROM drive is mounted and run the following command as root:

    ```
    pkgadd -G -d ./VBoxSolarisAdditions.pkg
    ```

3. Choose **1** and confirm installation of the Guest Additions package. After the installation is complete, log out and log in to X server on your guest, to activate the X11 Guest Additions.

4.2.3.2. Uninstalling the Oracle Solaris Guest Additions

The Oracle Solaris Guest Additions can be safely removed by removing the package from the guest. Open a root terminal session and run the following command:

```
pkgrm SUNWvboxguest
```

4.2.3.3. Updating the Oracle Solaris Guest Additions

The Guest Additions should be updated by first uninstalling the existing Guest Additions and then installing the new ones. Attempting to install new Guest Additions without removing the existing ones is not possible.

4.2.4. Guest Additions for OS/2

Oracle VM VirtualBox also ships with a set of drivers that improve running OS/2 in a virtual machine. Due to restrictions of OS/2 itself, this variant of the Guest Additions has a limited feature set. See Known Limitations for details.

The OS/2 Guest Additions are provided on the same ISO CD-ROM as those for the other platforms. Mount the ISO in OS/2 as described previously. The OS/2 Guest Additions are located in the directory `\OS2`.

We do not provide an automatic installer at this time. See the `readme.txt` file in the CD-ROM directory, which describes how to install the OS/2 Guest Additions manually.

4.3. Shared Folders

With the *shared folders* feature of Oracle VM VirtualBox, you can access files of your host system from within the guest system. This is similar to how you would use network shares in Windows networks, except that shared folders do not require networking, only the Guest Additions. Shared folders are supported with Windows 2000 or later, Linux, and Oracle Solaris guests. Oracle VM VirtualBox includes experimental support for Mac OS X and OS/2 guests.

Shared folders physically reside on the *host* and are then shared with the guest, which uses a special file system driver in the Guest Additions to talk to the host. For Windows guests, shared folders are implemented as a pseudo-network redirector. For Linux and Oracle Solaris guests, the Guest Additions provide a virtual file system.

To share a host folder with a virtual machine in Oracle VM VirtualBox, you must specify the path of the folder and choose a *share name* that the guest can use to access the shared folder. This happens on the host. In the guest you can then use the share name to connect to it and access files.

There are several ways in which shared folders can be set up for a virtual machine:

- In the window of a running VM, you select **Shared Folders** from the **Devices** menu, or click on the folder icon on the status bar in the bottom right corner.

- If a VM is not currently running, you can configure shared folders in the virtual machine's **Settings** dialog.

- From the command line, you can create shared folders using `VBoxManage`, as follows:

```
VBoxManage sharedfolder add "VM name" --name "sharename" --hostpath "C:\test"
```

See Section 7.31, "VBoxManage sharedfolder add/remove".

There are two types of shares:

- Permanent shares, that are saved with the VM settings.

- Transient shares, that are added at runtime and disappear when the VM is powered off. These can be created using a checkbox in the VirtualBox Manager, or by using the `--transient` option of the `VBoxManage sharedfolder add` command.

Shared folders can either be read-write or read-only. This means that the guest is either allowed to both read and write, or just read files on the host. By default, shared folders are read-write. Read-only folders

can be created using a checkbox in the VirtualBox Manager, or with the `--readonly` option of the `VBoxManage sharedfolder add` command.

Oracle VM VirtualBox shared folders also support symbolic links, also called *symlinks*, under the following conditions:

- The host operating system must support symlinks. For example, a Mac OS X, Linux, or Oracle Solaris host is required.

- Currently only Linux and Oracle Solaris Guest Additions support symlinks.

- For security reasons the guest OS is not allowed to create symlinks by default. If you trust the guest OS to not abuse the functionality, you can enable creation of symlinks for a shared folder as follows:

```
VBoxManage setextradata "VM name" VBoxInternal2/SharedFoldersEnableSymlinksCreate/sharename 1
```

4.3.1. Manual Mounting

You can mount the shared folder from inside a VM, in the same way as you would mount an ordinary network share:

- In a Windows guest, shared folders are browseable and therefore visible in Windows Explorer. To attach the host's shared folder to your Windows guest, open Windows Explorer and look for the folder in **My Networking Place**s, **Entire Network**, **Oracle VM VirtualBox Shared Folders**. By right-clicking on a shared folder and selecting **Map Network Drive** from the menu that pops up, you can assign a drive letter to that shared folder.

  Alternatively, on the Windows command line, use the following command:

  ```
  net use x: \\vboxsvr\sharename
  ```

  While `vboxsvr` is a fixed name, note that `vboxsrv` would also work, replace `x:` with the drive letter that you want to use for the share, and `sharename` with the share name specified with `VBoxManage`.

- In a Linux guest, use the following command:

  ```
  mount -t vboxsf [-o OPTIONS] sharename mountpoint
  ```

  To mount a shared folder during boot, add the following entry to `/etc/fstab`:

  ```
  sharename    mountpoint    vboxsf    defaults   0    0
  ```

- In a Oracle Solaris guest, use the following command:

  ```
  mount -F vboxfs [-o OPTIONS] sharename mountpoint
  ```

  Replace `sharename`, use a lowercase string, with the share name specified with `VBoxManage` or the VirtualBox Manager. Replace `mountpoint` with the path where you want the share to be mounted on the guest, such as `/mnt/share`. The usual mount rules apply. For example, create this directory first if it does not exist yet.

  Here is an example of mounting the shared folder for the user jack on Oracle Solaris:

  ```
  $ id
  uid=5000(jack) gid=1(other)
  $ mkdir /export/home/jack/mount
  $ pfexec mount -F vboxfs -o uid=5000,gid=1 jackshare /export/home/jack/mount
  $ cd ~/mount
  $ ls
  sharedfile1.mp3 sharedfile2.txt
  $
  ```

Beyond the standard options supplied by the `mount` command, the following are available:

```
iocharset CHARSET
```

This option sets the character set used for I/O operations. Note that on Linux guests, if the `iocharset` option is not specified, then the Guest Additions driver will attempt to use the character set specified by the CONFIG_NLS_DEFAULT kernel option. If this option is not set either, then UTF-8 is used.

```
convertcp CHARSET
```

This option specifies the character set used for the shared folder name. This is UTF-8 by default.

The generic mount options, documented in the `mount` manual page, apply also. Especially useful are the options `uid`, `gid` and `mode`, as they can allow access by normal users in read/write mode, depending on the settings, even if root has mounted the filesystem.

- In an OS/2 guest, use the `VBoxControl` command to manage shared folders. For example:

```
VBoxControl sharedfolder use D: MyShareName
VBoxControl sharedfolder unuse D:
VBoxControl sharedfolder list
```

As with Windows guests, shared folders can also be accessed via UNC using `\\VBoxSF\`, `\\VBoxSvr\` or `\\VBoxSrv\` as the server name and the shared folder name as *sharename*.

4.3.2. Automatic Mounting

Oracle VM VirtualBox provides the option to mount shared folders automatically. When automatic mounting is enabled for a shared folder, the Guest Additions service will mount it for you automatically. For Windows or OS/2, a preferred drive letter can also be specified. For Linux or Oracle Solaris, a mount point directory can also be specified.

If a drive letter or mount point is not specified, or is in use already, an alternative location is found by the Guest Additions service. The service searches for an alternative location depending on the guest OS, as follows:

- **Windows and OS/2 guests.** Search for a free drive letter, starting at `Z:`. If all drive letters are assigned, the folder is not mounted.

- **Linux and Oracle Solaris guests.** Folders are mounted under the `/media` directory. The folder name is normalized (no spaces, slashes or colons) and is prefixed with `sf_`.

  For example, if you have a shared folder called `myfiles`, it will appear as `/media/sf_myfiles` in the guest.

  The guest properties `/VirtualBox/GuestAdd/SharedFolders/MountDir` and the more generic `/VirtualBox/GuestAdd/SharedFolders/MountPrefix` can be used to override the automatic mount directory and prefix. See Section 4.7, "Guest Properties".

Access to an automatically mounted shared folder is granted to everyone in a Windows guest, including the guest user. For Linux and Oracle Solaris guests, access is restricted to members of the group `vboxsf` and the `root` user.

4.4. Drag and Drop

Oracle VM VirtualBox enables you to drag and drop content from the host to the guest, and vice versa. For this to work the latest version of the Guest Additions must be installed on the guest.

Drag and drop transparently allows copying or opening files, directories, and even certain clipboard formats from one end to the other. For example, from the host to the guest or from the guest to the host. You then can perform drag and drop operations between the host and a VM, as it would be a native drag and drop operation on the host OS.

At the moment drag and drop is implemented for Windows-based and X-Windows-based systems, both on the host and guest side. As X-Windows supports many different drag and drop protocols only the most common one, XDND, is supported for now. Applications using other protocols, such as Motif or OffiX, will not be recognized by Oracle VM VirtualBox.

In the context of using drag and drop, the origin of the data is called the *source*. That is, where the actual data comes from and is specified. The *target* specifies where the data from the source should go to. Transferring data from the source to the target can be done in various ways, such as copying, moving, or linking.

Note

At the moment only copying of data is supported. Moving or linking is not yet implemented.

When transferring data from the host to the guest OS, the host in this case is the source, whereas the guest OS is the target. However, when transferring data from the guest OS to the host, the guest OS this time became the source and the host is the target.

For security reasons drag and drop can be configured at runtime on a per-VM basis either using the **Drag and Drop** menu item in the **Devices** menu of the virtual machine, as shown below, or the `VBoxManage` command.

Figure 4.1 Drag and Drop Menu Options
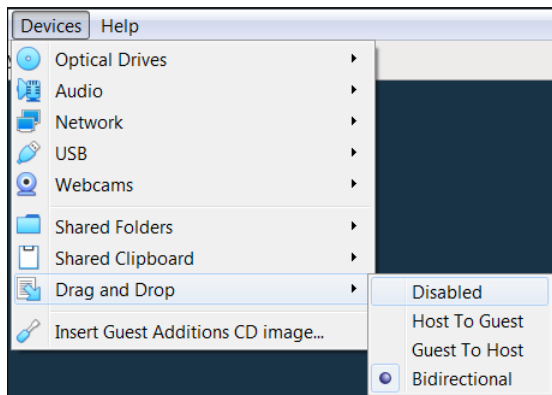
The following drag and drop modes are available:

• **Disabled.** Disables the drag and drop feature entirely. This is the default when creating a new VM.

• **Host To Guest.** Enables drag and drop operations from the host to the guest only.

• **Guest To Host.** Enables drag and drop operations from the guest to the host only.

• **Bidirectional.** Enables drag and drop operations in both directions: from the host to the guest, and from the guest to the host.

Note

Drag and drop support depends on the frontend being used. At the moment, only the VirtualBox Manager frontend provides this functionality.

To use the `VBoxManage` command to control the current drag and drop mode, see Chapter 7, *VBoxManage*. The `modifyvm` and `controlvm` commands enable setting of a VM's current drag and drop mode from the command line.

4.4.1. Supported Formats

As Oracle VM VirtualBox can run on a variety of host operating systems and also supports a wide range of guests, certain data formats must be translated after transfer. This is so that the target operating system, which receiving the data, is able to handle them in an appropriate manner.

> **Note**
>
> When dragging files no data conversion is done in any way. For example, when transferring a file from a Linux guest to a Windows host the Linux-specific line endings are not converted to Windows line endings.

The following formats are handled by the Oracle VM VirtualBox drag and drop service:

- **Plain text:** From applications such as text editors, internet browsers and terminal windows.

- **Files:** From file managers such as Windows Explorer, Nautilus, and Finder.

- **Directories:** For directories, the same formats apply as for files.

4.4.2. Known Limitations

The following limitations are known for drag and drop:

On Windows hosts, dragging and dropping content between UAC-elevated (User Account Control) programs and non-UAC-elevated programs is not allowed. If you start Oracle VM VirtualBox with Administrator privileges then drag and drop will not work with Windows Explorer, which runs with regular user privileges by default.

4.5. Hardware-Accelerated Graphics

4.5.1. Hardware 3D Acceleration (OpenGL and Direct3D 8/9)

The Oracle VM VirtualBox Guest Additions contain experimental hardware 3D support for Windows, Linux, and Oracle Solaris guests.

With this feature, if an application inside your virtual machine uses 3D features through the OpenGL or Direct3D 8/9 programming interfaces, instead of emulating them in software, which would be slow, Oracle VM VirtualBox will attempt to use your host's 3D hardware. This works for all supported host platforms, provided that your host operating system can make use of your accelerated 3D hardware in the first place.

The 3D acceleration feature currently has the following preconditions:

- It is only available for certain Windows, Linux, and Oracle Solaris guests. In particular:

  - 3D acceleration with Windows guests requires Windows 2000 or later. Apart from on Windows 2000 guests, both OpenGL and Direct3D 8/9 are supported on an experimental basis.

  - OpenGL on Linux requires kernel 2.6.27 or later, as well as X.org server version 1.5 or later. Ubuntu 10.10 and Fedora 14 have been tested and confirmed as working.

- OpenGL on Oracle Solaris guests requires X.org server version 1.5 or later.

- The Guest Additions must be installed.

> **Note**
>
> For the basic Direct3D acceleration to work in a Windows Guest, Oracle VM VirtualBox needs to replace Windows system files in the virtual machine. As a result, the Guest Additions installation program offers Direct3D acceleration as an option that must be explicitly enabled. Also, you must install the Guest Additions in Safe Mode. This does *not* apply to the WDDM Direct3D video driver available for Windows Vista and later. See Known Limitations for details.

- Because 3D support is still experimental at this time, it is disabled by default and must be *manually enabled* in the VM settings. See Section 3.6, "Display Settings".

> **Note**
>
> Untrusted guest systems should not be allowed to use the 3D acceleration features of Oracle VM VirtualBox, just as untrusted host software should not be allowed to use 3D acceleration. Drivers for 3D hardware are generally too complex to be made properly secure and any software which is allowed to access them may be able to compromise the operating system running them. In addition, enabling 3D acceleration gives the guest direct access to a large body of additional program code in the Oracle VM VirtualBox host process which it might conceivably be able to use to crash the virtual machine.

To enable Aero theme support, the Oracle VM VirtualBox WDDM video driver must be installed, which is available with the Guest Additions installation. The WDDM driver is not installed by default for Vista and Windows 7 guests and must be *manually selected* in the Guest Additions installer by clicking **No** in the **Would You Like to Install Basic Direct3D Support** dialog displayed when the Direct3D feature is selected.

The Aero theme is not enabled by default on Windows. See your Windows platform documentation for details of how to enable the Aero theme.

Technically, Oracle VM VirtualBox implements 3D acceleration by installing an additional hardware 3D driver inside the guest when the Guest Additions are installed. This driver acts as a hardware 3D driver and reports to the guest operating system that the virtual hardware is capable of 3D hardware acceleration. When an application in the guest then requests hardware acceleration through the OpenGL or Direct3D programming interfaces, these are sent to the host through a special communication tunnel implemented by Oracle VM VirtualBox. The *host* then performs the requested 3D operation using the host's programming interfaces.

4.5.2. Hardware 2D Video Acceleration for Windows Guests

The Oracle VM VirtualBox Guest Additions contain experimental hardware 2D video acceleration support for Windows guests.

With this feature, if an application such as a video player inside your Windows VM uses 2D video overlays to play a movie clip, then Oracle VM VirtualBox will attempt to use your host's video acceleration hardware instead of performing overlay stretching and color conversion in software, which would be slow. This currently works for Windows, Linux and Mac OS X host platforms, provided that your host operating system can make use of 2D video acceleration in the first place.

Hardware 2D video acceleration currently has the following preconditions:

- Only available for Windows guests, running Windows XP or later.

- Guest Additions must be installed.

- Because 2D support is still experimental at this time, it is disabled by default and must be *manually enabled* in the VM settings. See Section 3.6, "Display Settings".

Technically, Oracle VM VirtualBox implements this by exposing video overlay DirectDraw capabilities in the Guest Additions video driver. The driver sends all overlay commands to the host through a special communication tunnel implemented by Oracle VM VirtualBox. On the host side, OpenGL is then used to implement color space transformation and scaling.

4.6. Seamless Windows

With the *seamless windows* feature of Oracle VM VirtualBox, you can have the windows that are displayed within a virtual machine appear side by side next to the windows of your host. This feature is supported for the following guest operating systems, provided that the Guest Additions are installed:

- Windows guests.

- Supported Linux or Oracle Solaris guests running the X Window System.

After seamless windows are enabled, Oracle VM VirtualBox suppresses the display of the desktop background of your guest, allowing you to run the windows of your guest operating system seamlessly next to the windows of your host.

Figure 4.2 Seamless Windows on a Host Desktop
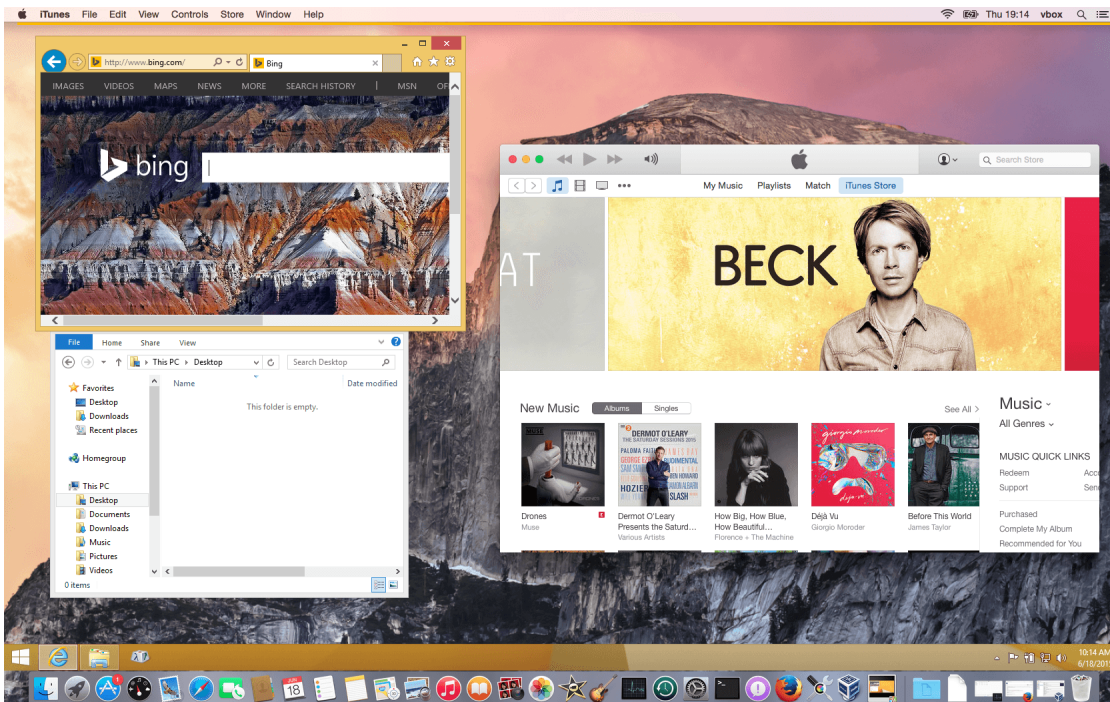
To enable seamless mode, after starting the virtual machine, press the **Host key + L**. The Host key is normally the right control key. This will enlarge the size of the VM's display to the size of your host screen and mask out the guest operating system's background. To disable seamless windows and go back to the normal VM display, press the Host key + L again.

4.7. Guest Properties

Oracle VM VirtualBox enables requests of some properties from a running guest, provided that the Oracle VM VirtualBox Guest Additions are installed and the VM is running. This provides the following advantages:

- A number of predefined VM characteristics are automatically maintained by Oracle VM VirtualBox and can be retrieved on the host. For example, to monitor VM performance and statistics.

- Arbitrary string data can be exchanged between guest and host. This works in both directions.

To accomplish this, Oracle VM VirtualBox establishes a private communication channel between the Oracle VM VirtualBox Guest Additions and the host, and software on both sides can use this channel to exchange string data for arbitrary purposes. Guest properties are simply string keys to which a value is attached. They can be set, or written to, by either the host and the guest. They can also be read from both sides.

In addition to establishing the general mechanism of reading and writing values, a set of predefined guest properties is automatically maintained by the Oracle VM VirtualBox Guest Additions to allow for retrieving interesting guest data such as the guest's exact operating system and service pack level, the installed version of the Guest Additions, users that are currently logged into the guest OS, network statistics and more. These predefined properties are all prefixed with `/VirtualBox/` and organized into a hierarchical tree of keys.

Some of this runtime information is shown when you select **Session Information Dialog** from a virtual machine's **Machine** menu.

A more flexible way to use this channel is with the `VBoxManage guestproperty` command. See Section 7.32, "VBoxManage guestproperty". For example, to have *all* the available guest properties for a given running VM listed with their respective values, use this command:

```
$ VBoxManage guestproperty enumerate "Windows Vista III"
VirtualBox Command Line Management Interface Version version-number
(C) 2005-2019 Oracle Corporation
All rights reserved.

Name: /VirtualBox/GuestInfo/OS/Product, value: Windows Vista Business Edition,
    timestamp: 1229098278843087000, flags:
Name: /VirtualBox/GuestInfo/OS/Release, value: 6.0.6001,
    timestamp: 1229098278950553000, flags:
Name: /VirtualBox/GuestInfo/OS/ServicePack, value: 1,
    timestamp: 1229098279122627000, flags:
Name: /VirtualBox/GuestAdd/InstallDir,
    value: C:/Program Files/Oracle/VirtualBox
    Guest Additions, timestamp: 1229098279269739000, flags:
Name: /VirtualBox/GuestAdd/Revision, value: 40720,
    timestamp: 1229098279345664000, flags:
Name: /VirtualBox/GuestAdd/Version, value: version-number,
    timestamp: 1229098279479515000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxControl.exe, value: version-numberr40720,
    timestamp: 1229098279651731000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxHook.dll, value: version-numberr40720,
    timestamp: 1229098279804835000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxDisp.dll, value: version-numberr40720,
    timestamp: 1229098279880611000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxMRXNP.dll, value: version-numberr40720,
    timestamp: 1229098279882618000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxService.exe, value: version-numberr40720,
    timestamp: 1229098279883195000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxTray.exe, value: version-numberr40720,
    timestamp: 1229098279885027000, flags:
```

```
Name: /VirtualBox/GuestAdd/Components/VBoxGuest.sys, value: version-numberr40720,
    timestamp: 1229098279886838000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxMouse.sys, value: version-numberr40720,
    timestamp: 1229098279890600000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxSF.sys, value: version-numberr40720,
    timestamp: 1229098279893056000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxVideo.sys, value: version-numberr40720,
    timestamp: 1229098279895767000, flags:
Name: /VirtualBox/GuestInfo/OS/LoggedInUsers, value: 1,
    timestamp: 1229099826317660000, flags:
Name: /VirtualBox/GuestInfo/OS/NoLoggedInUsers, value: false,
    timestamp: 1229098455580553000, flags:
Name: /VirtualBox/GuestInfo/Net/Count, value: 1,
    timestamp: 1229099826299785000, flags:
Name: /VirtualBox/HostInfo/GUI/LanguageID, value: C,
    timestamp: 1229098151272771000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/IP, value: 192.168.2.102,
    timestamp: 1229099826300088000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/Broadcast, value: 255.255.255.255,
    timestamp: 1229099826300220000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/Netmask, value: 255.255.255.0,
    timestamp: 1229099826300350000, flags:
Name: /VirtualBox/GuestInfo/Net/0/Status, value: Up,
    timestamp: 1229099826300524000, flags:
Name: /VirtualBox/GuestInfo/OS/LoggedInUsersList, value: username,
    timestamp: 1229099826317386000, flags:
```

To query the value of a single property, use the `get` subcommand as follows:

```
$ VBoxManage guestproperty get "Windows Vista III" "/VirtualBox/GuestInfo/OS/Product"
VirtualBox Command Line Management Interface Version version-number
(C) 2005-2019 Oracle Corporation
All rights reserved.

Value: Windows Vista Business Edition
```

To add or change guest properties from the guest, use the tool `VBoxControl`. This tool is included in the Guest Additions. When started from a Linux guest, this tool requires root privileges for security reasons.

```
$ sudo VBoxControl guestproperty enumerate
VirtualBox Guest Additions Command Line Management Interface Version version-number
(C) 2005-2019 Oracle Corporation
All rights reserved.

Name: /VirtualBox/GuestInfo/OS/Release, value: 2.6.28-18-generic,
    timestamp: 1265813265835667000, flags: <NULL>
Name: /VirtualBox/GuestInfo/OS/Version, value: #59-Ubuntu SMP Thu Jan 28 01:23:03 UTC 2010,
    timestamp: 1265813265836305000, flags: <NULL>
      ...
```

For more complex needs, you can use the Oracle VM VirtualBox programming interfaces. See Oracle VM VirtualBox Programming Interfaces.

4.7.1. Using Guest Properties to Wait on VM Events

The properties `/VirtualBox/HostInfo/VBoxVer`, `/VirtualBox/HostInfo/VBoxVerExt` or `/VirtualBox/HostInfo/VBoxRev` can be waited on to detect that the VM state was restored from saved state or snapshot:

```
$ VBoxControl guestproperty wait /VirtualBox/HostInfo/VBoxVer
```

Similarly the `/VirtualBox/HostInfo/ResumeCounter` can be used to detect that a VM was resumed from the paused state or saved state.

4.8. Guest Control File Manager

The Guest Control File Manager is a feature of the Guest Additions that enables easy copying and moving of files between a guest and the host system. Other file management operations provide support to create new folders and to rename or delete files.

Figure 4.3 Guest Control File Manager
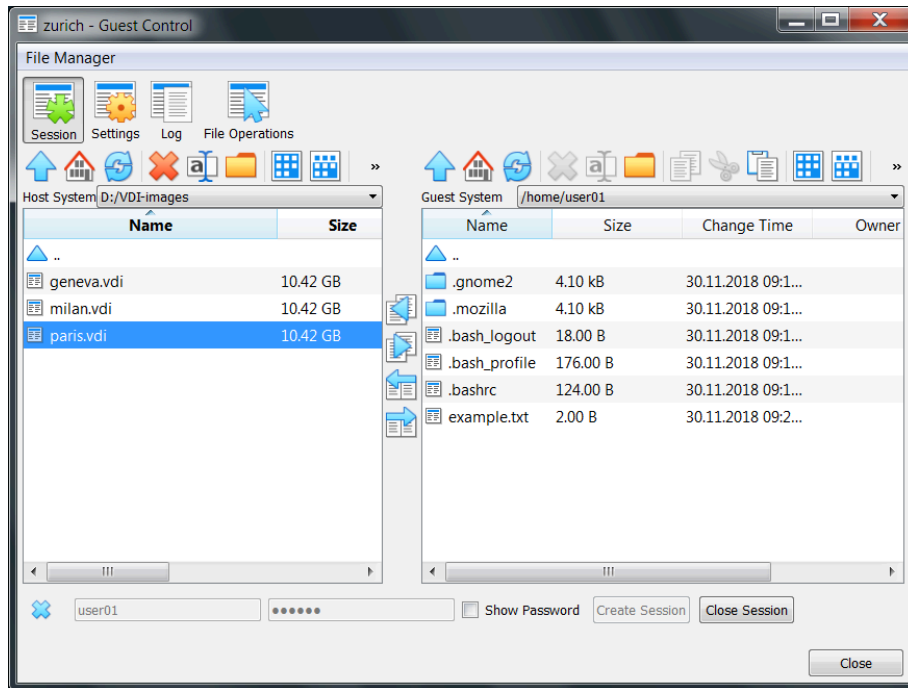
The Guest Control File Manager works by mounting the host file system. Guest users must authenticate and create a guest session before they can transfer files.

4.8.1. Using the Guest Control File Manager

The following steps describe how to use the Guest Control File Manager.

1. Open the Guest Control File Manager.

   In the guest VM, select **Machine**, **File Manager**.

   The left pane shows the files on the host system.

2. Create a guest session.

   At the bottom of the Guest Control File Manager, enter authentication credentials for a user on the guest system.

   Click **Create Session**.

   The contents of the guest VM file system appears in the right pane of the Guest Control File Manager.

3. Transfer files between the guest and the host system by using the move and copy file transfer icons.

   You can copy and move files from a guest to the host system or from the host system to the guest.

4. Close the Guest Control File Manager.

   Click **Close** to end the guest session.

4.9. Guest Control of Applications

The Guest Additions enable starting of applications inside a guest VM from the host system. This feature can be used to automate deployment of software within the guest.

For this to work, the application needs to be installed on the guest. No additional software needs to be installed on the host. Additionally, text mode output to stdout and stderr can be shown on the host for further processing. There are options to specify user credentials and a timeout value, in milliseconds, to limit the time the application is able to run.

The Guest Additions for Windows allow for automatic updating. This applies for already installed Guest Additions versions. Also, copying files from host to the guest as well as remotely creating guest directories is available.

To use these features, use the Oracle VM VirtualBox command line. See Section 7.33, "VBoxManage guestcontrol".

4.10. Memory Overcommitment

In server environments with many VMs, the Guest Additions can be used to share physical host memory between several VMs. This reduces the total amount of memory in use by the VMs. If memory usage is the limiting factor and CPU resources are still available, this can help with running more VMs on each host.

4.10.1. Memory Ballooning

The Guest Additions can change the amount of host memory that a VM uses, while the machine is running. Because of how this is implemented, this feature is called *memory ballooning*.

Note

- Oracle VM VirtualBox supports memory ballooning only on 64-bit hosts. It is not supported on Mac OS X hosts.

- Memory ballooning does not work with large pages enabled. To turn off large pages support for a VM, run `VBoxManage modifyvm` *vmname* `--largepages off`

Normally, to change the amount of memory allocated to a virtual machine, you have to shut down the virtual machine entirely and modify its settings. With memory ballooning, memory that was allocated for a virtual machine can be given to another virtual machine without having to shut the machine down.

When memory ballooning is requested, the Oracle VM VirtualBox Guest Additions, which run inside the guest, allocate physical memory from the guest operating system on the kernel level and lock this memory down in the guest. This ensures that the guest will not use that memory any longer. No guest applications can allocate it, and the guest kernel will not use it either. Oracle VM VirtualBox can then reuse this memory and give it to another virtual machine.

The memory made available through the ballooning mechanism is only available for reuse by Oracle VM VirtualBox. It is *not* returned as free memory to the host. Requesting balloon memory from a running guest will therefore not increase the amount of free, unallocated memory on the host. Effectively, memory

ballooning is therefore a memory overcommitment mechanism for multiple virtual machines while they are running. This can be useful to temporarily start another machine, or in more complicated environments, for sophisticated memory management of many virtual machines that may be running in parallel depending on how memory is used by the guests.

At this time, memory ballooning is only supported through `VBoxManage`. Use the following command to increase or decrease the size of the memory balloon within a running virtual machine that has Guest Additions installed:

```
VBoxManage controlvm "VM name" guestmemoryballoon n
```

where *VM name* is the name or UUID of the virtual machine in question and *n* is the amount of memory to allocate from the guest in megabytes. See Section 7.13, "VBoxManage controlvm".

You can also set a default balloon that will automatically be requested from the VM every time after it has started up with the following command:

```
VBoxManage modifyvm "VM name" --guestmemoryballoon n
```

By default, no balloon memory is allocated. This is a VM setting, like other `modifyvm` settings, and therefore can only be set while the machine is shut down. See Section 7.8, "VBoxManage modifyvm".

4.10.2. Page Fusion

Whereas memory ballooning simply reduces the amount of RAM that is available to a VM, Page Fusion works differently. It avoids memory duplication between several similar running VMs.

In a server environment running several similar VMs on the same host, lots of memory pages are identical. For example, if the VMs are using identical operating systems. Oracle VM VirtualBox's Page Fusion technology can efficiently identify these identical memory pages and share them between multiple VMs.

Note

Oracle VM VirtualBox supports Page Fusion only on 64-bit hosts, and it is not supported on Mac OS X hosts. Page Fusion currently works only with Windows 2000 and later guests.

The more similar the VMs on a given host are, the more efficiently Page Fusion can reduce the amount of host memory that is in use. It therefore works best if all VMs on a host run identical operating systems. Instead of having a complete copy of each operating system in each VM, Page Fusion identifies the identical memory pages in use by these operating systems and eliminates the duplicates, sharing host memory between several machines. This is called *deduplication*. If a VM tries to modify a page that has been shared with other VMs, a new page is allocated again for that VM with a copy of the shared page. This is called *copy on write*. All this is fully transparent to the virtual machine.

You may be familiar with this kind of memory overcommitment from other hypervisor products, which call this feature *page sharing* or *same page merging*. However, Page Fusion differs significantly from those other solutions, whose approaches have several drawbacks:

- Traditional hypervisors scan *all* guest memory and compute checksums, also called hashes, for every single memory page. Then, they look for pages with identical hashes and compare the entire content of those pages. If two pages produce the same hash, it is very likely that the pages are identical in content. This process can take rather long, especially if the system is not idling. As a result, the additional memory only becomes available after a significant amount of time, such as hours or sometimes days. Even worse, this kind of page sharing algorithm generally consumes significant CPU resources and increases the virtualization overhead by 10 to 20%.

Page Fusion in Oracle VM VirtualBox uses logic in the Oracle VM VirtualBox Guest Additions to quickly identify memory cells that are most likely identical across VMs. It can therefore achieve most of the possible savings of page sharing almost immediately and with almost no overhead.

- Page Fusion is also much less likely to be confused by identical memory that it will eliminate, just to learn seconds later that the memory will now change and having to perform a highly expensive and often service-disrupting reallocation.

At this time, Page Fusion can only be controlled with `VBoxManage`, and only while a VM is shut down. To enable Page Fusion for a VM, use the following command:

```
VBoxManage modifyvm "VM name" --pagefusion on
```

You can observe Page Fusion operation using some metrics. `RAM/VMM/Shared` shows the total amount of fused pages, whereas the per-VM metric `Guest/RAM/Usage/Shared` will return the amount of fused memory for a given VM. See Section 7.34, "VBoxManage metrics" for information on how to query metrics.

Note

Enabling Page Fusion might indirectly increase the chances for malicious guests to successfully attack other VMs running on the same host. See Potentially Insecure Operations.

Chapter 5 Virtual Storage

As the virtual machine will most probably expect to see a hard disk built into its virtual computer, Oracle VM VirtualBox must be able to present real storage to the guest as a virtual hard disk. There are presently three methods by which to achieve this:

- Oracle VM VirtualBox can use large image files on a real hard disk and present them to a guest as a virtual hard disk. This is the most common method, described in Section 5.2, "Disk Image Files (VDI, VMDK, VHD, HDD)".

- iSCSI storage servers can be attached to Oracle VM VirtualBox. This is described in Section 5.10, "iSCSI Servers".

- You can allow a virtual machine to access one of your host disks directly. This is an advanced feature, described in Using a Raw Host Hard Disk From a Guest.

Each such virtual storage device, such as an image file, iSCSI target, or physical hard disk, needs to be connected to the virtual hard disk controller that Oracle VM VirtualBox presents to a virtual machine. This is explained in the next section.

5.1. Hard Disk Controllers

In a computing device, hard disks and CD/DVD drives are connected to a device called hard disk controller which drives hard disk operation and data transfers. Oracle VM VirtualBox can emulate the most common types of hard disk controllers typically found in computing devices: IDE, SATA (AHCI), SCSI, SAS, USB-based, and NVMe mass storage devices.

- **IDE (ATA)** controllers are a backwards-compatible yet very advanced extension of the disk controller in the IBM PC/AT (1984). Initially, this interface worked only with hard disks, but was later extended to also support CD-ROM drives and other types of removable media. In physical PCs, this standard uses flat ribbon parallel cables with 40 or 80 wires. Each such cable can connect two devices to a controller, which have traditionally been called *master* and *slave*. Typical PCs had two connectors for such cables. As a result, support for up to four IDE devices was most common.

  In Oracle VM VirtualBox, each virtual machine may have one IDE controller enabled, which gives you up to four virtual storage devices that you can attach to the machine. By default, one of these virtual storage devices, the secondary master, is preconfigured to be the virtual machine's virtual CD/DVD drive. However, you can change the default setting.

  Even if your guest OS has no support for SCSI or SATA devices, it should always be able to see an IDE controller.

  You can also select which exact type of IDE controller hardware Oracle VM VirtualBox should present to the virtual machine: PIIX3, PIIX4, or ICH6. This makes no difference in terms of performance, but if you import a virtual machine from another virtualization product, the OS in that machine may expect a particular controller type and crash if it is not found.

  After you have created a new virtual machine with the **New Virtual Machine** wizard of the graphical user interface, you will typically see one IDE controller in the machine's **Storage** settings. The virtual CD/DVD drive will be attached to one of the four ports of this controller.

- **Serial ATA (SATA)** is a more recent standard than IDE. Compared to IDE, it supports both much higher speeds and more devices per controller. Also, with physical hardware, devices can be added and removed while the system is running. The standard interface for SATA controllers is called Advanced Host Controller Interface (AHCI).

Like a real SATA controller, Oracle VM VirtualBox's virtual SATA controller operates faster and also consumes fewer CPU resources than the virtual IDE controller. Also, this enables you to connect up to 30 virtual hard disks to one machine instead of just three, when compared to the Oracle VM VirtualBox IDE controller with a DVD drive attached.

For this reason, depending on the selected guest OS, Oracle VM VirtualBox uses SATA as the default for newly created virtual machines. One virtual SATA controller is created by default, and the default disk that is created with a new VM is attached to this controller.

> **Warning**
>
> The entire SATA controller and the virtual disks attached to it, including those in IDE compatibility mode, will not be seen by OSes that do not have device support for AHCI. In particular, *there is no support for AHCI in Windows versions before Windows Vista*. Legacy Windows versions such as Windows XP, even with SP3 installed, will not see such disks unless you install additional drivers. It is possible to switch from IDE to SATA after installation by installing the SATA drivers and changing the controller type in the VM **Settings** dialog.
>
> Oracle VM VirtualBox recommends the Intel Matrix Storage drivers, which can be downloaded from http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=2101.

To add a SATA controller to a machine for which it has not been enabled by default, either because it was created by an earlier version of Oracle VM VirtualBox, or because SATA is not supported by default by the selected guest OS, do the following. Go to the **Storage** page of the machine's **Settings** dialog, click **Add Controller** under the Storage Tree box and then select **Add SATA Controller**. The new controller appears as a separate PCI device in the virtual machine, and you can add virtual disks to it.

To change the IDE compatibility mode settings for the SATA controller, see Section 7.18, "VBoxManage storagectl".

- **SCSI** is another established industry standard, standing for Small Computer System Interface. SCSI is as a generic interface for data transfer between all kinds of devices, including storage devices. SCSI is still used for connecting some hard disks and tape devices, but it has mostly been displaced in commodity hardware. It is still in common use in high-performance workstations and servers.

  Primarily for compatibility with other virtualization software, Oracle VM VirtualBox optionally supports LSI Logic and BusLogic SCSI controllers, to each of which up to fifteen virtual hard disks can be attached.

  To enable a SCSI controller, on the **Storage** page of a virtual machine's **Settings** dialog, click **Add Controller** under the Storage Tree box and then select **Add SCSI Controller**. The new controller appears as a separate PCI device in the virtual machine.

  > **Warning**
  >
  > As with the other controller types, a SCSI controller will only be seen by OSes with device support for it. Windows 2003 and later ships with drivers for the LSI Logic controller, while Windows NT 4.0 and Windows 2000 ships with drivers for the BusLogic controller. Windows XP ships with drivers for neither.

- **Serial Attached SCSI (SAS)** is another bus standard which uses the SCSI command set. As opposed to SCSI physical devices, serial cables are used instead of parallel cables. This simplifies physical device

connections. In some ways, therefore, SAS is to SCSI what SATA is to IDE: it enables more reliable and faster connections.

To support high-end guests which require SAS controllers, Oracle VM VirtualBox emulates a LSI Logic SAS controller, which can be enabled much the same way as a SCSI controller. At this time, up to eight devices can be connected to the SAS controller.

Warning

As with SATA, the SAS controller will only be seen by OSes with device support for it. In particular, *there is no support for SAS in Windows before Windows Vista.* So Windows XP, even SP3, will not see such disks unless you install additional drivers.

- The **USB mass storage device class** is a standard to connect external storage devices like hard disks or flash drives to a host through USB. All major OSes support these devices and ship generic drivers making third-party drivers superfluous. In particular, legacy OSes without support for SATA controllers may benefit from USB mass storage devices.

  The virtual USB storage controller offered by Oracle VM VirtualBox works differently to the other storage controller types. While most storage controllers appear as a single PCI device to the guest with multiple disks attached to it, the USB-based storage controller does not appear as virtual storage controller. Each disk attached to the controller appears as a dedicated USB device to the guest.

Warning

Booting from drives attached using USB is only supported when EFI is used as the BIOS lacks USB support.

- **Non volatile memory express (NVMe)** is a standard for connecting non volatile memory (NVM) directly over PCI Express to lift the bandwidth limitation of the previously used SATA protocol for solid-state devices. Unlike other standards the command set is very simple in order to achieve maximum throughput and is not compatible with ATA or SCSI. OSes need to support NVMe devices to make use of them. For example, Windows 8.1 added native NVMe support. For Windows 7, native support was added with an update.

  The NVMe controller is part of the extension pack.

Warning

Booting from drives attached using NVMe is only supported when EFI is used as the BIOS lacks the appropriate driver.

In summary, Oracle VM VirtualBox gives you the following categories of virtual storage slots:

- Four slots attached to the traditional IDE controller, which are always present. One of these is typically a virtual CD/DVD drive.

- 30 slots attached to the SATA controller, if enabled and supported by the guest OS.

- 15 slots attached to the SCSI controller, if enabled and supported by the guest OS.

- Eight slots attached to the SAS controller, if enabled and supported by the guest OS.

- Eight slots attached to the virtual USB controller, if enabled and supported by the guest OS.

- Up to 255 slots attached to the NVMe controller, if enabled and supported by the guest OS.

Given this large choice of storage controllers, you may not know which one to choose. In general, you should avoid IDE unless it is the only controller supported by your guest. Whether you use SATA, SCSI, or SAS does not make any real difference. The variety of controllers is only supplied by Oracle VM VirtualBox for compatibility with existing hardware and other hypervisors.

5.2. Disk Image Files (VDI, VMDK, VHD, HDD)

Disk image files reside on the host system and are seen by the guest systems as hard disks of a certain geometry. When a guest OS reads from or writes to a hard disk, Oracle VM VirtualBox redirects the request to the image file.

Like a physical disk, a virtual disk has a size, or capacity, which must be specified when the image file is created. As opposed to a physical disk however, Oracle VM VirtualBox enables you to expand an image file after creation, even if it has data already. See Section 7.22, "VBoxManage modifymedium".

Oracle VM VirtualBox supports the following types of disk image files:

- **VDI.** Normally, Oracle VM VirtualBox uses its own container format for guest hard disks. This is called a Virtual Disk Image (VDI) file. This format is used when you create a new virtual machine with a new disk.

- **VMDK.** Oracle VM VirtualBox also fully supports the popular and open VMDK container format that is used by many other virtualization products, such as VMware.

- **VHD.** Oracle VM VirtualBox also fully supports the VHD format used by Microsoft.

- **HDD.** Image files of Parallels version 2 (HDD format) are also supported.

  Due to lack of documentation of the format, newer versions such as 3 and 4 are not supported. You can however convert such image files to version 2 format using tools provided by Parallels.

Irrespective of the disk capacity and format, as mentioned in Section 1.7, "Creating Your First Virtual Machine", there are two options for creating a disk image: fixed-size or dynamically allocated.

- **Fixed-size.** If you create a fixed-size image, an image file will be created on your host system which has roughly the same size as the virtual disk's capacity. So, for a 10 GB disk, you will have a 10 GB file. Note that the creation of a fixed-size image can take a long time depending on the size of the image and the write performance of your hard disk.

- **Dynamically allocated.** For more flexible storage management, use a dynamically allocated image. This will initially be very small and not occupy any space for unused virtual disk sectors, but will grow every time a disk sector is written to for the first time, until the drive reaches the maximum capacity chosen when the drive was created. While this format takes less space initially, the fact that Oracle VM VirtualBox needs to expand the image file consumes additional computing resources, so until the disk file size has stabilized, write operations may be slower than with fixed size disks. However, after a time the rate of growth will slow and the average penalty for write operations will be negligible.

5.3. The Virtual Media Manager

Oracle VM VirtualBox keeps track of all the hard disk, CD/DVD-ROM, and floppy disk images which are in use by virtual machines. These are often referred to as *known media* and come from two sources:

- All media currently attached to virtual machines.

- Registered media, for compatibility with legacy Oracle VM VirtualBox versions.

The known media can be viewed and changed using the **Virtual Media Manager**, which you can access from the **File** menu in the VirtualBox Manager window.

Figure 5.1 The Virtual Media Manager
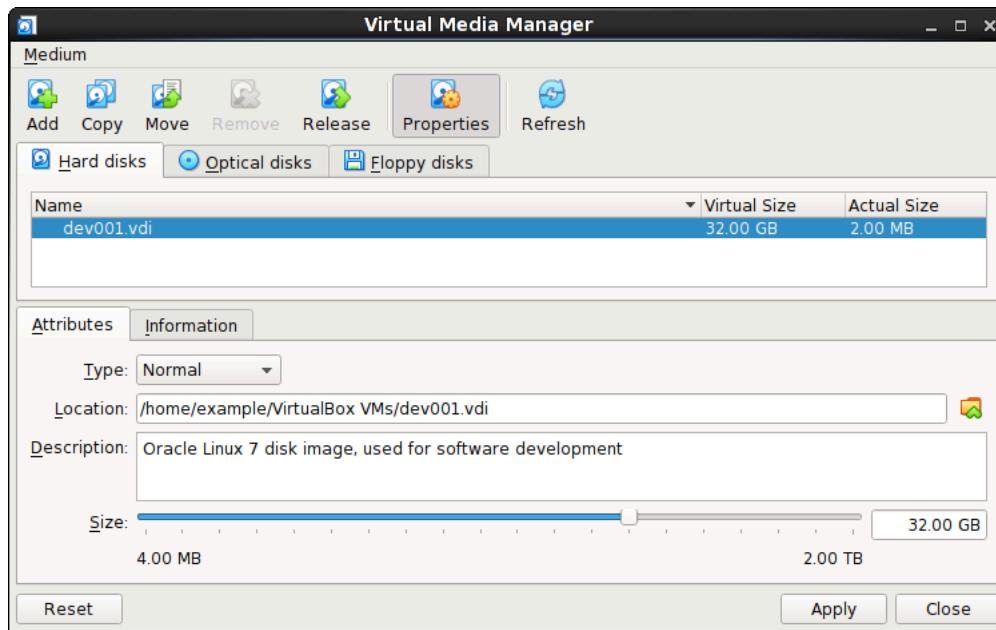
The known media are conveniently grouped in separate tabs for the supported formats. These formats are:

- Hard disk images, either in Oracle VM VirtualBox's own Virtual Disk Image (VDI) format, or in the third-party formats listed in Section 5.2, "Disk Image Files (VDI, VMDK, VHD, HDD)".

- CD/DVD images in standard ISO format.

- Floppy images in standard RAW format.

For each image, the Virtual Media Manager shows you the full path of the image file and other information, such as the virtual machine the image is currently attached to.

The Virtual Media Manager enables you to do the following:

- **Add** an image to the registry.

- **Copy** a virtual hard disk to create another one.

  You can specify one of the following target types: VDI, VHD, or VMDK.

- **Move** an image that is currently in the registry to another location.

  A file dialog prompts you for the new image file location.

  When you use the Virtual Media Manager to move a disk image, Oracle VM VirtualBox updates all related configuration files automatically.

  > **Note**
  >
  > Always use the Virtual Media Manager or the `VBoxManage modifymedium` command to move a disk image.
  >
  > If you use a file management feature of the host OS to move a disk image to a new location, run the `VBoxManage modifymedium --setlocation`

command to configure the new path of the disk image on the host file system. This command updates the Oracle VM VirtualBox configuration automatically.

- **Remove** an image from the registry. You can optionally delete the image file when removing the image.

- **Release** an image to detach it from a VM. This action only applies if the image is currently attached to a VM as a virtual hard disk.

- View and edit the **Properties** of a disk image.

    Available properties include the following:

    - **Type:** Specifies the snapshot behavior of the disk. See Section 5.4, "Special Image Write Modes".

    - **Location:** Specifies the location of the disk image file on the host system. You can use a file dialog to browse for the disk image location.

    - **Description:** Specifies a short description of the disk image.

    - **Size:** Specifies the size of the disk image. You can use the slider to increase or decrease the disk image size.

    - **Information:** Specifies detailed information about the disk image.

- **Refresh** the property values of the selected disk image.

To perform these actions, highlight the medium in the Virtual Media Manager and then do one of the following:

- Click an icon in the Virtual Media Manager task bar.

- Right-click the medium and select an option.

Use the **Storage** page in a VM's **Settings** dialog to create a new disk image. By default, disk images are stored in the VM's folder.

You can copy hard disk image files to other host systems and then import them in to VMs from the host system. However, some Windows guest OSes may require that you configure the new VM in a similar way to the old one.

> **Note**
>
> Do not simply make copies of virtual disk images. If you import such a second copy into a VM, Oracle VM VirtualBox issues an error because Oracle VM VirtualBox assigns a universally unique identifier (UUID) to each disk image to ensure that it is only used one time. See Section 5.6, "Cloning Disk Images". Also, if you want to copy a VM to another system, use the Oracle VM VirtualBox import and export features. See Section 1.14, "Importing and Exporting Virtual Machines".

5.4. Special Image Write Modes

For each virtual disk image supported by Oracle VM VirtualBox, you can determine separately how it should be affected by write operations from a virtual machine and snapshot operations. This applies to all of the aforementioned image formats (VDI, VMDK, VHD, or HDD) and irrespective of whether an image is fixed-size or dynamically allocated.

By default, images are in *normal* mode. To mark an existing image with one of the non-standard modes listed below, use `VBoxManage modifymedium`. See Section 7.22, "VBoxManage modifymedium".

Alternatively, use `VBoxManage storageattach` to attach the image to a VM and specify the `--mtype` argument. See Section 7.17, "VBoxManage storageattach".

The available virtual disk image modes are as follows:

- **Normal images** have no restrictions on how guests can read from and write to the disk. This is the default image mode.

  When you take a snapshot of your virtual machine as described in Section 1.10, "Snapshots", the state of a normal hard disk is recorded together with the snapshot, and when reverting to the snapshot, its state will be fully reset.

  The image file itself is not reset. Instead, when a snapshot is taken, Oracle VM VirtualBox *freezes* the image file and no longer writes to it. For the write operations from the VM, a second, *differencing* image file is created which receives only the changes to the original image. See Section 5.5, "Differencing Images".

  While you can attach the same normal image to more than one virtual machine, only one of these virtual machines attached to the same image file can be executed simultaneously, as otherwise there would be conflicts if several machines write to the same image file.

- **Write-through hard disks** are completely unaffected by snapshots. Their state is *not* saved when a snapshot is taken, and not restored when a snapshot is restored.

- **Shareable hard disks** are a variant of write-through hard disks. In principle they behave exactly the same. Their state is *not* saved when a snapshot is taken, and not restored when a snapshot is restored. The difference only shows if you attach such disks to several VMs. Shareable disks may be attached to several VMs which may run concurrently. This makes them suitable for use by cluster filesystems between VMs and similar applications which are explicitly prepared to access a disk concurrently. Only fixed size images can be used in this way, and dynamically allocated images are rejected.

  Warning

  This is an expert feature, and misuse can lead to data loss, as regular filesystems are not prepared to handle simultaneous changes by several parties.

- **Immutable images** only remember write accesses temporarily while the virtual machine is running. All changes are lost when the virtual machine is powered on the next time. As a result, as opposed to Normal images, the same immutable image can be used with several virtual machines without restrictions.

  Creating an immutable image makes little sense since it would be initially empty and lose its contents with every machine restart. You would have a disk that is always unformatted when the machine starts up. Instead, you can first create a normal image and then later mark it as immutable when you decide that the contents are useful.

  If you take a snapshot of a machine with immutable images, then on every machine power-up, those images are reset to the state of the last (current) snapshot, instead of the state of the original immutable image.

  Note

  As a special exception, immutable images are *not* reset if they are attached to a machine in a saved state or whose last snapshot was taken while the machine was running. This is called an *online snapshot*. As a result, if the machine's current snapshot is an online snapshot, its immutable images behave exactly like

| the a normal image. To reenable the automatic resetting of such images, delete the current snapshot of the machine.

Oracle VM VirtualBox never writes to an immutable image directly at all. All write operations from the machine are directed to a differencing image. The next time the VM is powered on, the differencing image is reset so that every time the VM starts, its immutable images have exactly the same content.

The differencing image is only reset when the machine is powered on from within Oracle VM VirtualBox, not when you reboot by requesting a reboot from within the machine. This is also why immutable images behave as described above when snapshots are also present, which use differencing images as well.

If the automatic discarding of the differencing image on VM startup does not fit your needs, you can turn it off using the `autoreset` parameter of `VBoxManage modifymedium`. See Section 7.22, "VBoxManage modifymedium".

- **Multiattach mode images** can be attached to more than one virtual machine at the same time, even if these machines are running simultaneously. For each virtual machine to which such an image is attached, a differencing image is created. As a result, data that is written to such a virtual disk by one machine is not seen by the other machines to which the image is attached. Each machine creates its own write history of the multiattach image.

  Technically, a multiattach image behaves identically to an immutable image except the differencing image is not reset every time the machine starts.

  This mode is useful for sharing files which are almost never written, for instance picture galleries, where every guest changes only a small amount of data and the majority of the disk content remains unchanged. The modified blocks are stored in differencing images which remain relatively small and the shared content is stored only once at the host.

- **Read-only images** are used automatically for CD/DVD images, since CDs/DVDs can never be written to.

The following scenario illustrates the differences between the various image modes, with respect to snapshots.

Assume you have installed your guest OS in your VM, and you have taken a snapshot. Later, your VM is infected with a virus and you would like to go back to the snapshot. With a normal hard disk image, you simply restore the snapshot, and the earlier state of your hard disk image will be restored as well and your virus infection will be undone. With an immutable hard disk, all it takes is to shut down and power on your VM, and the virus infection will be discarded. With a write-through image however, you cannot easily undo the virus infection by means of virtualization, but will have to disinfect your virtual machine like a real computer.

You might find write-through images useful if you want to preserve critical data irrespective of snapshots. As you can attach more than one image to a VM, you may want to have one immutable image for the OS and one write-through image for your data files.

5.5. Differencing Images

The previous section mentioned differencing images and how they are used with snapshots, immutable images, and multiple disk attachments. This section describes in more detail how differencing images work.

A differencing image is a special disk image that only holds the differences to another image. A differencing image by itself is useless, it must always refer to another image. The differencing image is then typically referred to as a *child*, which holds the differences to its *parent*.

When a differencing image is active, it receives all write operations from the virtual machine instead of its parent. The differencing image only contains the sectors of the virtual hard disk that have changed since the differencing image was created. When the machine reads a sector from such a virtual hard disk, it looks into the differencing image first. If the sector is present, it is returned from there. If not, Oracle VM VirtualBox looks into the parent. In other words, the parent becomes *read-only*. It is never written to again, but it is read from if a sector has not changed.

Differencing images can be chained. If another differencing image is created for a virtual disk that already has a differencing image, then it becomes a *grandchild* of the original parent. The first differencing image then becomes read-only as well, and write operations only go to the second-level differencing image. When reading from the virtual disk, Oracle VM VirtualBox needs to look into the second differencing image first, then into the first if the sector was not found, and then into the original image.

There can be an unlimited number of differencing images, and each image can have more than one child. As a result, the differencing images can form a complex tree with parents, siblings, and children, depending on how complex your machine configuration is. Write operations always go to the one *active* differencing image that is attached to the machine, and for read operations, Oracle VM VirtualBox may need to look up all the parents in the chain until the sector in question is found. You can view such a tree in the Virtual Media Manager.

Figure 5.2 Differencing Images, Shown in Virtual Media Manager

In all of these situations, from the point of view of the virtual machine, the virtual hard disk behaves like any other disk. While the virtual machine is running, there is a slight run-time I/O overhead because Oracle VM VirtualBox might need to look up sectors several times. This is not noticeable however since the tables with sector information are always kept in memory and can be looked up quickly.

Differencing images are used in the following situations:

- **Snapshots.** When you create a snapshot, as explained in the previous section, Oracle VM VirtualBox *freezes* the images attached to the virtual machine and creates differencing images for each image that is not in *write-through* mode. From the point of view of the virtual machine, the virtual disks continue to operate before, but all write operations go into the differencing images. Each time you create another snapshot, for each hard disk attachment, another differencing image is created and attached, forming a chain or tree.

In the above screenshot, you see that the original disk image is now attached to a snapshot, representing the state of the disk when the snapshot was taken.

If you *restore* a snapshot, and want to go back to the exact machine state that was stored in the snapshot, the following happens:

- Oracle VM VirtualBox copies the virtual machine settings that were copied into the snapshot back to the virtual machine. As a result, if you have made changes to the machine configuration since taking the snapshot, they are undone.

- If the snapshot was taken while the machine was running, it contains a saved machine state, and that state is restored as well. After restoring the snapshot, the machine will then be in Saved state and resume execution from there when it is next started. Otherwise the machine will be in Powered Off state and do a full boot.

- For each disk image attached to the machine, the differencing image holding all the write operations since the current snapshot was taken is thrown away, and the original parent image is made active again. If you restored the root snapshot, then this will be the root disk image for each attachment. Otherwise, some other differencing image descended from it. This effectively restores the old machine state.

If you later *delete* a snapshot in order to free disk space, for each disk attachment, one of the differencing images becomes obsolete. In this case, the differencing image of the disk attachment cannot simply be deleted. Instead, Oracle VM VirtualBox needs to look at each sector of the differencing image and needs to copy it back into its parent. This is called "merging" images and can be a potentially lengthy process, depending on how large the differencing image is. It can also temporarily need a considerable amount of extra disk space, before the differencing image obsoleted by the merge operation is deleted.

- **Immutable images.** When an image is switched to immutable mode, a differencing image is created as well. As with snapshots, the parent image then becomes read-only, and the differencing image receives all the write operations. Every time the virtual machine is started, all the immutable images which are attached to it have their respective differencing image thrown away, effectively resetting the virtual machine's virtual disk with every restart.

5.6. Cloning Disk Images

You can duplicate hard disk image files on the same host to quickly produce a second virtual machine with the same OS setup. However, you should *only* make copies of virtual disk images using the utility supplied with Oracle VM VirtualBox. See Section 7.23, "VBoxManage clonemedium". This is because Oracle VM VirtualBox assigns a UUID to each disk image, which is also stored inside the image, and Oracle VM VirtualBox will refuse to work with two images that use the same number. If you do accidentally try to reimport a disk image which you copied normally, you can make a second copy using the `VBoxManage clonevm` command and import that instead.

Note that Linux distributions identify the boot hard disk from the ID of the drive. The ID Oracle VM VirtualBox reports for a drive is determined from the UUID of the virtual disk image. So if you clone a disk image and try to boot the copied image the guest might not be able to determine its own boot disk as the UUID changed. In this case you have to adapt the disk ID in your boot loader script, for example `/boot/grub/menu.lst`. The disk ID looks like the following:

```
scsi-SATA_VBOX_HARDDISK_VB5cfdb1e2-c251e503
```

The ID for the copied image can be determined as follows:

```
hdparm -i /dev/sda
```

5.7. Host Input/Output Caching

Oracle VM VirtualBox can optionally disable the I/O caching that the host OS would otherwise perform on disk image files.

Traditionally, Oracle VM VirtualBox has opened disk image files as normal files, which results in them being cached by the host OS like any other file. The main advantage of this is speed: when the guest OS writes to disk and the host OS cache uses delayed writing, the write operation can be reported as completed to the guest OS quickly while the host OS can perform the operation asynchronously. Also, when you start a VM a second time and have enough memory available for the OS to use for caching, large parts of the virtual disk may be in system memory, and the VM can access the data much faster.

Note that this applies only to image files. Buffering does not occur for virtual disks residing on remote iSCSI storage, which is the more common scenario in enterprise-class setups. See Section 5.10, "iSCSI Servers".

While buffering is a useful default setting for virtualizing a few machines on a desktop computer, there are some disadvantages to this approach:

- Delayed writing through the host OS cache is less secure. When the guest OS writes data, it considers the data written even though it has not yet arrived on a physical disk. If for some reason the write does not happen, such as power failure or host crash, the likelihood of data loss increases.

- Disk image files tend to be very large. Caching them can therefore quickly use up the entire host OS cache. Depending on the efficiency of the host OS caching, this may slow down the host immensely, especially if several VMs run at the same time. For example, on Linux hosts, host caching may result in Linux delaying all writes until the host cache is nearly full and then writing out all these changes at once, possibly stalling VM execution for minutes. This can result in I/O errors in the guest as I/O requests time out there.

- Physical memory is often wasted as guest OSes typically have their own I/O caches, which may result in the data being cached twice, in both the guest and the host caches, for little effect.

If you decide to disable host I/O caching for the above reasons, Oracle VM VirtualBox uses its own small cache to buffer writes, but no read caching since this is typically already performed by the guest OS. In addition, Oracle VM VirtualBox fully supports asynchronous I/O for its virtual SATA, SCSI, and SAS controllers through multiple I/O threads.

Since asynchronous I/O is not supported by IDE controllers, for performance reasons, you may want to leave host caching enabled for your VM's virtual IDE controllers.

For this reason, Oracle VM VirtualBox enables you to configure whether the host I/O cache is used for each I/O controller separately. Either select the **Use Host I/O Cache** check box in the **Storage** settings for a given virtual storage controller, or use the following `VBoxManage` command to disable the host I/O cache for a virtual storage controller:

```
VBoxManage storagectl "VM name" --name <controllername> --hostiocache off
```

See Section 7.18, "VBoxManage storagectl".

For the above reasons, Oracle VM VirtualBox uses SATA controllers by default for new virtual machines.

5.8. Limiting Bandwidth for Disk Images

Oracle VM VirtualBox supports limiting of the maximum bandwidth used for asynchronous I/O. Additionally it supports sharing limits through bandwidth groups for several images. It is possible to have more than one such limit.

Limits are configured using `VBoxManage`. The example below creates a bandwidth group named Limit, sets the limit to 20 MB per second, and assigns the group to the attached disks of the VM:

```
VBoxManage bandwidthctl "VM name" add Limit --type disk --limit 20M
VBoxManage storageattach "VM name" --storagectl "SATA" --port 0 --device 0 --type hdd
                               --medium disk1.vdi --bandwidthgroup Limit
VBoxManage storageattach "VM name" --storagectl "SATA" --port 1 --device 0 --type hdd
                               --medium disk2.vdi --bandwidthgroup Limit
```

All disks in a group share the bandwidth limit, meaning that in the example above the bandwidth of both images combined can never exceed 20 MBps. However, if one disk does not require bandwidth the other can use the remaining bandwidth of its group.

The limits for each group can be changed while the VM is running, with changes being picked up immediately. The example below changes the limit for the group created in the example above to 10 MBps:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 10M
```

5.9. CD/DVD Support

Virtual CD/DVD drives by default support only reading. The medium configuration is changeable at runtime. You can select between the following options to provide the medium data:

- **Host Drive** defines that the guest can read from the medium in the host drive.

- **Image file** gives the guest read-only access to the data in the image. This is typically an ISO file.

- **Empty** means a drive without an inserted medium.

Changing between the above, or changing a medium in the host drive that is accessed by a machine, or changing an image file will signal a medium change to the guest OS. The guest OS can then react to the change, for example by starting an installation program.

Medium changes can be prevented by the guest, and Oracle VM VirtualBox reflects that by locking the host drive if appropriate. You can force a medium removal in such situations by using the VirtualBox Manager or the `VBoxManage` command line tool. Effectively this is the equivalent of the emergency eject which many CD/DVD drives provide, with all associated side effects. The guest OS can issue error messages, just like on real hardware, and guest applications may misbehave. Use this with caution.

> **Note**
>
> The identification string of the drive provided to the guest, displayed by configuration tools such as the Windows Device Manager, is always VBOX CD-ROM, irrespective of the current configuration of the virtual drive. This is to prevent hardware detection from being triggered in the guest OS every time the configuration is changed.

The standard CD/DVD emulation enables reading of standard data CD and DVD formats only. As an experimental feature, for additional capabilities, it is possible to give the guest direct access to the CD/DVD host drive by enabling *passthrough* mode. Depending on the host hardware, this may potentially enable the following things to work:

- CD/DVD writing from within the guest, if the host DVD drive is a CD/DVD writer

- Playing audio CDs

- Playing encrypted DVDs

There is a **Passthrough** check box in the VirtualBox Manager dialog for configuring the media attached to a storage controller, or you can use the `--passthrough` option with `VBoxManage storageattach`. See Section 7.17, "VBoxManage storageattach".

Even if passthrough is enabled, unsafe commands, such as updating the drive firmware, will be blocked. Video CD formats are never supported, not even in passthrough mode, and cannot be played from a virtual machine.

On Oracle Solaris hosts, passthrough requires running Oracle VM VirtualBox with real root permissions due to security measures enforced by the host.

5.10. iSCSI Servers

iSCSI stands for *Internet SCSI* and is a standard that supports use of the SCSI protocol over Internet (TCP/IP) connections. Especially with the advent of Gigabit Ethernet, it has become affordable to attach iSCSI storage servers simply as remote hard disks to a computer network. In iSCSI terminology, the server providing storage resources is called an *iSCSI target*, while the client connecting to the server and accessing its resources is called an *iSCSI initiator*.

Oracle VM VirtualBox can transparently present iSCSI remote storage to a virtual machine as a virtual hard disk. The guest OS will not see any difference between a virtual disk image (VDI file) and an iSCSI target. To achieve this, Oracle VM VirtualBox has an integrated iSCSI initiator.

Oracle VM VirtualBox's iSCSI support has been developed according to the iSCSI standard and should work with all standard-conforming iSCSI targets. To use an iSCSI target with Oracle VM VirtualBox, you must use the command line. See Section 7.17, "VBoxManage storageattach".

5.11. vboximg-mount: A Utility for FUSE Mounting a Virtual Disk Image

`vboximg-mount` is a command line utility for Mac OS X hosts that provides raw access to an Oracle VM VirtualBox virtual disk image on the host system. Use this utility to mount, view, and optionally modify the disk image contents.

The utility is based on Filesystem in Userspace (FUSE) technology and uses the VirtualBox runtime engine. Ensure that Oracle VM VirtualBox is running on the host system.

> **Note**
>
> When using `vboximg-mount`, ensure that the following conditions apply:
>
> - The disk image is not being used by any other systems, such as by guest VMs.
>
> - No VMs are running on the host system.

Raw access using FUSE is preferred over direct loopback mounting of virtual disk images, because it is snapshot aware. It can selectively merge disk differencing images in an exposed virtual hard disk, providing historical or up-to-date representations of the virtual disk contents.

`vboximg-mount` enables you to view information about registered VMs, their attached disk media, and any snapshots. Also, you can view partition information for a disk image.

Use the `--help` option to view information about the `vboximg-mount` command usage.

When `vboximg-mount` mounts an Oracle VM VirtualBox disk image, it creates a one level deep file system at a mount point that you specify. The file system includes a device node that represents the

synthesized disk image as a readable or readable-writeable bytestream. This bytestream can be mounted either by using the host OS or by using other FUSE-based file systems.

5.11.1. Viewing Detailed Information About a Virtual Disk Image

The following examples show how to use the `vboximg-mount` command to view information about virtual disk images.

The following command outputs detailed information about all registered VMs and associated snapshots:

```
$ vboximg-mount --list --verbose

    -----------------------------------------------------
    VM Name:    "macOS High Sierra 10.13"
    UUID:       3887d96d-831c-4187-a55a-567c504ff0e1
    Location:  /Volumes/work/vm_guests/macOS High Sierra 10.13/macOS High Sierra 10.13.vbox
        ----------------------
        HDD base:   "macOS High Sierra 10.13.vdi"
        UUID:       f9ea7173-6869-4aa9-b487-68023a655980
        Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/macOS High Sierra 10.13.vdi

          Diff 1:
                UUID:       98c2bac9-cf37-443d-a935-4e879b70166d
                Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/
                Snapshots/{98c2bac9-cf37-443d-a935-4e879b70166d}.vdi
          Diff 2:
                UUID:       f401f381-7377-40b3-948e-3c61241b1a42
                Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/
                Snapshots/{f401f381-7377-40b3-948e-3c61241b1a42}.vdi
        ----------------------
        HDD base:   "simple_fixed_disk.vdi"
        UUID:       ffba4d7e-1277-489d-8173-22ca7660773d
        Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/simple_fixed_disk.vdi

          Diff 1:
                UUID:       aecab681-0d2d-468b-8682-93f79dc97a48
                Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/
                Snapshots/{aecab681-0d2d-468b-8682-93f79dc97a48}.vdi
          Diff 2:
                UUID:       70d6b34d-8422-47fa-8521-3b6929a1971c
                Location:   /Volumes/work/vm_guests/macOS High Sierra 10.13/
                Snapshots/{70d6b34d-8422-47fa-8521-3b6929a1971c}.vdi
    -----------------------------------------------------
    VM Name:    "debian"
    UUID:       5365ab5f-470d-44c0-9863-dad532ee5905
    Location:  /Volumes/work/vm_guests/debian/debian.vbox
        ----------------------
        HDD base:   "debian.vdi"
        UUID:       96d2e92e-0d4e-46ab-a0f1-008fdbf997e7
        Location:   /Volumes/work/vm_guests/debian/ol7.vdi

          Diff 1:
                UUID:       f9cc866a-9166-42e9-a503-bbfe9b7312e8
                Location:   /Volumes/work/vm_guests/debian/Snapshots/
                {f9cc866a-9166-42e9-a503-bbfe9b7312e8}.vdi
```

The following command outputs partition information about the specified disk image:

```
$ vboximg-mount --image=f9ea7173-6869-4aa9-b487-68023a655980 --list

    Virtual disk image:

        Path: /Volumes/work/vm_guests/macOS High Sierra 10.13/macOS High Sierra 10.13.vdi
        UUID: f9ea7173-6869-4aa9-b487-68023a655980
```

```
#    Start    Sectors    Size      Offset   Type
1       40    409599    199.9M      20480   EFI System
2   409640  67453071    32.1G   209735680   Hierarchical File System Plus (HFS+)
3 67862712   1269535    107.8M  34745708544  Apple Boot (Recovery HD)
```

5.11.2. Mounting a Virtual Disk Image

The following steps show how to use the `vboximg-mount` command to mount a partition of a virtual disk image on the host OS.

1. Create a mount point on the host OS. For example:

```
$ mkdir macos_sysdisk
```

2. Show partition information about the virtual disk image.

```
$ vboximg-mount --image=uuid --list
```

where `uuid` is the UUID of the disk image.

3. Use `vboximg-mount` to perform a FUSE mount of a partition on the virtual disk image. For example:

```
$ vboximg-mount --image=uuid -p 2 macos_sysdisk
```

where `uuid` is the UUID for the disk image.

In this example, partition 2 is mounted on the `macos_sysdisk` mount point. The mount includes all snapshots for the disk image.

4. Use the host OS to mount the `vhdd` device node. The FUSE-mounted device node represents the virtual disk image.

```
$ ls macos_sysdisk
  macOS High Sierra 10.13.vdi  vhdd
$ sudo mount macos_sysdisk/vhdd /mnt
```

Chapter 6 Virtual Networking

As mentioned in Section 3.9, "Network Settings", Oracle VM VirtualBox provides up to eight virtual PCI Ethernet cards for each virtual machine. For each such card, you can individually select the following:

- The hardware that will be virtualized.

- The virtualization mode that the virtual card operates in, with respect to your physical networking hardware on the host.

Four of the network cards can be configured in the **Network** section of the **Settings** dialog in the graphical user interface of Oracle VM VirtualBox. You can configure all eight network cards on the command line using `VBoxManage modifyvm`. See Section 7.8, "VBoxManage modifyvm".

This chapter explains the various networking settings in more detail.

6.1. Virtual Networking Hardware

For each card, you can individually select what kind of *hardware* will be presented to the virtual machine. Oracle VM VirtualBox can virtualize the following types of networking hardware:

- AMD PCNet PCI II (Am79C970A)

- AMD PCNet FAST III (Am79C973), the default setting

- Intel PRO/1000 MT Desktop (82540EM)

- Intel PRO/1000 T Server (82543GC)

- Intel PRO/1000 MT Server (82545EM)

- Paravirtualized network adapter (virtio-net)

The PCNet FAST III is the default because it is supported by nearly all operating systems, as well as by the GNU GRUB boot manager. As an exception, the Intel PRO/1000 family adapters are chosen for some guest operating system types that no longer ship with drivers for the PCNet card, such as Windows Vista.

The Intel PRO/1000 MT Desktop type works with Windows Vista and later versions. The T Server variant of the Intel PRO/1000 card is recognized by Windows XP guests without additional driver installation. The MT Server variant facilitates OVF imports from other platforms.

The Paravirtualized network adapter (virtio-net) is special. If you select this adapter, then Oracle VM VirtualBox does *not* virtualize common networking hardware that is supported by common guest operating systems. Instead, Oracle VM VirtualBox expects a special software interface for virtualized environments to be provided by the guest, thus avoiding the complexity of emulating networking hardware and improving network performance. Oracle VM VirtualBox provides support for the industry-standard *virtio* networking drivers, which are part of the open source KVM project.

The virtio networking drivers are available for the following guest operating systems:

- Linux kernels version 2.6.25 or later can be configured to provide virtio support. Some distributions have also back-ported virtio to older kernels.

- For Windows 2000, XP, and Vista, virtio drivers can be downloaded and installed from the KVM project web page:

  http://www.linux-kvm.org/page/WindowsGuestDrivers.

Oracle VM VirtualBox also has limited support for *jumbo frames*. These are networking packets with more than 1500 bytes of data, provided that you use the Intel card virtualization and bridged networking. Jumbo frames are not supported with the AMD networking devices. In those cases, jumbo packets will silently be dropped for both the transmit and the receive direction. Guest operating systems trying to use this feature will observe this as a packet loss, which may lead to unexpected application behavior in the guest. This does not cause problems with guest operating systems in their default configuration, as jumbo frames need to be explicitly enabled.

6.2. Introduction to Networking Modes

Each of the networking adapters can be separately configured to operate in one of the following modes:

- **Not attached.** In this mode, Oracle VM VirtualBox reports to the guest that a network card is present, but that there is no connection. This is as if no Ethernet cable was plugged into the card. Using this mode, it is possible to *pull* the virtual Ethernet cable and disrupt the connection, which can be useful to inform a guest operating system that no network connection is available and enforce a reconfiguration.

- **Network Address Translation (NAT)**. If all you want is to browse the Web, download files, and view email inside the guest, then this default mode should be sufficient for you, and you can skip the rest of this section. Please note that there are certain limitations when using Windows file sharing. See Section 6.3.3, "NAT Limitations".

- **NAT Network.** A NAT network is a type of internal network that allows outbound connections. See Section 6.4, "Network Address Translation Service".

- **Bridged networking.** This is for more advanced networking needs, such as network simulations and running servers in a guest. When enabled, Oracle VM VirtualBox connects to one of your installed network cards and exchanges network packets directly, circumventing your host operating system's network stack.

- **Internal networking.** This can be used to create a different kind of software-based network which is visible to selected virtual machines, but not to applications running on the host or to the outside world.

- **Host-only networking.** This can be used to create a network containing the host and a set of virtual machines, without the need for the host's physical network interface. Instead, a virtual network interface, similar to a loopback interface, is created on the host, providing connectivity among virtual machines and the host.

- **Generic networking.** Rarely used modes which share the same generic network interface, by allowing the user to select a driver which can be included with Oracle VM VirtualBox or be distributed in an extension pack.

  The following sub-modes are available:

  - **UDP Tunnel:** Used to interconnect virtual machines running on different hosts directly, easily, and transparently, over an existing network infrastructure.

  - **VDE (Virtual Distributed Ethernet) networking:** Used to connect to a Virtual Distributed Ethernet switch on a Linux or a FreeBSD host. At the moment this option requires compilation of Oracle VM VirtualBox from sources, as the Oracle packages do not include it.

The following table provides an overview of the most important networking modes.

Table 6.1 Overview of Networking Modes

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Internal	–	–	**+**	–	–
Bridged	**+**	**+**	**+**	**+**	**+**
NAT	**+**	Port forward	–	**+**	Port forward
NATservice	**+**	Port forward	**+**	**+**	Port forward

The following sections describe the available network modes in more detail.

6.3. Network Address Translation (NAT)

Network Address Translation (NAT) is the simplest way of accessing an external network from a virtual machine. Usually, it does not require any configuration on the host network and guest system. For this reason, it is the default networking mode in Oracle VM VirtualBox.

A virtual machine with NAT enabled acts much like a real computer that connects to the Internet through a router. The router, in this case, is the Oracle VM VirtualBox networking engine, which maps traffic from and to the virtual machine transparently. In Oracle VM VirtualBox this router is placed between each virtual machine and the host. This separation maximizes security since by default virtual machines cannot talk to each other.

The disadvantage of NAT mode is that, much like a private network behind a router, the virtual machine is invisible and unreachable from the outside internet. You cannot run a server this way unless you set up port forwarding. See Section 6.3.1, "Configuring Port Forwarding with NAT".

The network frames sent out by the guest operating system are received by Oracle VM VirtualBox's NAT engine, which extracts the TCP/IP data and resends it using the host operating system. To an application on the host, or to another computer on the same network as the host, it looks like the data was sent by the Oracle VM VirtualBox application on the host, using an IP address belonging to the host. Oracle VM VirtualBox listens for replies to the packages sent, and repacks and resends them to the guest machine on its private network.

The virtual machine receives its network address and configuration on the private network from a DHCP server integrated into Oracle VM VirtualBox. The IP address thus assigned to the virtual machine is usually on a completely different network than the host. As more than one card of a virtual machine can be set up to use NAT, the first card is connected to the private network 10.0.2.0, the second card to the network 10.0.3.0 and so on. If you need to change the guest-assigned IP range, see Fine Tuning the Oracle VM VirtualBox NAT Engine.

6.3.1. Configuring Port Forwarding with NAT

As the virtual machine is connected to a private network internal to Oracle VM VirtualBox and invisible to the host, network services on the guest are not accessible to the host machine or to other computers on the same network. However, like a physical router, Oracle VM VirtualBox can make selected services available to the world outside the guest through *port forwarding*. This means that Oracle VM VirtualBox listens to certain ports on the host and resends all packets which arrive there to the guest, on the same or a different port.

To an application on the host or other physical or virtual machines on the network, it looks as though the service being proxied is actually running on the host. This also means that you cannot run the same service on the same ports on the host. However, you still gain the advantages of running the service in a virtual machine. For example, services on the host machine or on other virtual machines cannot be compromised or crashed by a vulnerability or a bug in the service, and the service can run in a different operating system than the host system.

To configure port forwarding you can use the graphical **Port Forwarding** editor which can be found in the **Network Settings** dialog for network adaptors configured to use NAT. Here, you can map host ports to guest ports to allow network traffic to be routed to a specific port in the guest.

Alternatively, the command line tool `VBoxManage` can be used. See Section 7.8, "VBoxManage modifyvm".

You will need to know which ports on the guest the service uses and to decide which ports to use on the host. You may want to use the same ports on the guest and on the host. You can use any ports on the host which are not already in use by a service. For example, to set up incoming NAT connections to an `ssh` server in the guest, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,,22"
```

In the above example, all TCP traffic arriving on port 2222 on any host interface will be forwarded to port 22 in the guest. The protocol name `tcp` is a mandatory attribute defining which protocol should be used for forwarding, `udp` could also be used. The name `guestssh` is purely descriptive and will be auto-generated if omitted. The number after `--natpf` denotes the network card, as with other `VBoxManage` commands.

To remove this forwarding rule, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 delete "guestssh"
```

If for some reason the guest uses a static assigned IP address not leased from the built-in DHCP server, it is required to specify the guest IP when registering the forwarding rule, as follows:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,10.0.2.19,22"
```

This example is identical to the previous one, except that the NAT engine is being told that the guest can be found at the 10.0.2.19 address.

To forward *all* incoming traffic from a specific host interface to the guest, specify the IP of that host interface as follows:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,127.0.0.1,2222,,22"
```

This example forwards all TCP traffic arriving on the localhost interface at 127.0.0.1 through port 2222 to port 22 in the guest.

It is possible to configure incoming NAT connections while the VM is running, see Section 7.13, "VBoxManage controlvm".

6.3.2. PXE Booting with NAT

PXE booting is now supported in NAT mode. The NAT DHCP server provides a boot file name of the form `vmname.pxe` if the directory `TFTP` exists in the directory where the user's `VirtualBox.xml` file is kept. It is the responsibility of the user to provide `vmname.pxe`.

6.3.3. NAT Limitations

There are some limitations of NAT mode which users should be aware of, as follows:

- **ICMP protocol limitations.** Some frequently used network debugging tools, such as `ping` or `traceroute`, rely on the ICMP protocol for sending and receiving messages. Oracle VM VirtualBox ICMP support has some limitations, meaning `ping` should work but some other tools may not work reliably.

- **Receiving of UDP broadcasts.** The guest does not reliably receive UDP broadcasts. In order to save resources, it only listens for a certain amount of time after the guest has sent UDP data on a particular port. As a consequence, NetBios name resolution based on broadcasts does not always work, but WINS always works. As a workaround, you can use the numeric IP of the desired server in the `\\server\share` notation.

- **Some protocols are not supported.** Protocols other than TCP and UDP are not supported. GRE is not supported. This means some VPN products, such as PPTP from Microsoft, cannot be used. There are other VPN products which use only TCP and UDP.

- **Forwarding host ports below 1024.** On UNIX-based hosts, such as Linux, Oracle Solaris, and Mac OS X, it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a port forwarding, the VM will refuse to start.

These limitations normally do not affect standard network use. But the presence of NAT has also subtle effects that may interfere with protocols that are normally working. One example is NFS, where the server is often configured to refuse connections from non-privileged ports, which are those ports not below 1024.

6.4. Network Address Translation Service

The Network Address Translation (NAT) service works in a similar way to a home router, grouping the systems using it into a network and preventing systems outside of this network from directly accessing systems inside it, but letting systems inside communicate with each other and with systems outside using TCP and UDP over IPv4 and IPv6.

A NAT service is attached to an internal network. Virtual machines which are to make use of it should be attached to that internal network. The name of internal network is chosen when the NAT service is created and the internal network will be created if it does not already exist. The following is an example command to create a NAT network:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable
```

Here, natnet1 is the name of the internal network to be used and 192.168.15.0/24 is the network address and mask of the NAT service interface. By default in this static configuration the gateway will be assigned the address 192.168.15.1, the address following the interface address, though this is subject to change. To attach a DHCP server to the internal network, modify the example command as follows:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable --dhcp on
```

To add a DHCP server to an existing network, use the following command:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp on
```

To disable the DHCP server, use the following command:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp off
```

A DHCP server provides a list of registered nameservers, but does not map servers from the 127/8 network.

To start the NAT service, use the following command:

```
VBoxManage natnetwork start --netname natnet1
```

If the network has a DHCP server attached then it will start together with the NAT network service.

To stop the NAT network service, together with any DHCP server:

```
VBoxManage natnetwork stop --netname natnet1
```

To delete the NAT network service:

```
VBoxManage natnetwork remove --netname natnet1
```

This command does not remove the DHCP server if one is enabled on the internal network.

Port-forwarding is supported, using the `--port-forward-4` switch for IPv4 and `--port-forward-6` for IPv6. For example:

```
VBoxManage natnetwork modify \
  --netname natnet1 --port-forward-4 "ssh:tcp:[]:1022:[192.168.15.5]:22"
```

This adds a port-forwarding rule from the host's TCP 1022 port to the port 22 on the guest with IP address 192.168.15.5. Host port, guest port and guest IP are mandatory. To delete the rule, use the following command:

```
VBoxManage natnetwork modify --netname natnet1 --port-forward-4 delete ssh
```

It is possible to bind a NAT service to specified interface. For example:

```
VBoxManage setextradata global "NAT/win-nat-test-0/SourceIp4" 192.168.1.185
```

To see the list of registered NAT networks, use the following command:

```
VBoxManage list natnetworks
```

6.5. Bridged Networking

With bridged networking, Oracle VM VirtualBox uses a device driver on your *host* system that filters data from your physical network adapter. This driver is therefore called a *net filter* driver. This enables Oracle VM VirtualBox to intercept data from the physical network and inject data into it, effectively creating a new network interface in software. When a guest is using such a new software interface, it looks to the host system as though the guest were physically connected to the interface using a network cable. The host can send data to the guest through that interface and receive data from it. This means that you can set up routing or bridging between the guest and the rest of your network.

> **Note**
>
> Even though TAP interfaces are no longer necessary on Linux for bridged networking, you *can* still use TAP interfaces for certain advanced setups, since you can connect a VM to any host interface.

To enable bridged networking, open the **Settings** dialog of a virtual machine, go to the **Network** page and select **Bridged Network** in the drop-down list for the **Attached To** field. Select a host interface from the list at the bottom of the page, which contains the physical network interfaces of your systems. On a typical MacBook, for example, this will allow you to select between en1: AirPort, which is the wireless interface, and en0: Ethernet, which represents the interface with a network cable.

> **Note**
>
> Bridging to a wireless interface is done differently from bridging to a wired interface, because most wireless adapters do not support promiscuous mode. All traffic has to use the MAC address of the host's wireless adapter, and therefore Oracle VM VirtualBox needs to replace the source MAC address in the Ethernet header of an outgoing packet to make sure the reply will be sent to the host interface. When

> Oracle VM VirtualBox sees an incoming packet with a destination IP address that belongs to one of the virtual machine adapters it replaces the destination MAC address in the Ethernet header with the VM adapter's MAC address and passes it on. Oracle VM VirtualBox examines ARP and DHCP packets in order to learn the IP addresses of virtual machines.

Depending on your host operating system, the following limitations apply:

- **Mac OS X hosts.** Functionality is limited when using AirPort, the Mac's wireless networking system, for bridged networking. Currently, Oracle VM VirtualBox supports only IPv4 and IPv6 over AirPort. For other protocols, such as IPX, you must choose a wired interface.

- **Linux hosts.** Functionality is limited when using wireless interfaces for bridged networking. Currently, Oracle VM VirtualBox supports only IPv4 and IPv6 over wireless. For other protocols, such as IPX, you must choose a wired interface.

  Also, setting the MTU to less than 1500 bytes on wired interfaces provided by the sky2 driver on the Marvell Yukon II EC Ultra Ethernet NIC is known to cause packet losses under certain conditions.

  Some adapters strip VLAN tags in hardware. This does not allow you to use VLAN trunking between VM and the external network with pre-2.6.27 Linux kernels, or with host operating systems other than Linux.

- **Oracle Solaris hosts.** There is no support for using wireless interfaces. Filtering guest traffic using IPFilter is also not completely supported due to technical restrictions of the Oracle Solaris networking subsystem. These issues may be addressed in later releases of Oracle Solaris 11.

  On Oracle Solaris 11 hosts build 159 and above, it is possible to use Oracle Solaris Crossbow Virtual Network Interfaces (VNICs) directly with Oracle VM VirtualBox without any additional configuration other than each VNIC must be exclusive for every guest network interface.

  When using VLAN interfaces with Oracle VM VirtualBox, they must be named according to the PPA-hack naming scheme, such as e1000g513001. Otherwise, the guest may receive packets in an unexpected format.

6.6. Internal Networking

Internal Networking is similar to bridged networking in that the VM can directly communicate with the outside world. However, the outside world is limited to other VMs on the same host which connect to the same internal network.

Even though technically, everything that can be done using internal networking can also be done using bridged networking, there are security advantages with internal networking. In bridged networking mode, all traffic goes through a physical interface of the host system. It is therefore possible to attach a packet sniffer such as Wireshark to the host interface and log all traffic that goes over it. If, for any reason, you prefer two or more VMs on the same machine to communicate privately, hiding their data from both the host system and the user, bridged networking therefore is not an option.

Internal networks are created automatically as needed. There is no central configuration. Every internal network is identified simply by its name. Once there is more than one active virtual network card with the same internal network ID, the Oracle VM VirtualBox support driver will automatically *wire* the cards and act as a network switch. The Oracle VM VirtualBox support driver implements a complete Ethernet switch and supports both broadcast/multicast frames and promiscuous mode.

In order to attach a VM's network card to an internal network, set its networking mode to Internal Networking. There are two ways to accomplish this:

- Use the VM's **Settings** dialog in the Oracle VM VirtualBox graphical user interface. In the **Networking** category of the settings dialog, select **Internal Networking** from the drop-down list of networking modes. Select the name of an existing internal network from the drop-down list below, or enter a new name into the **Name** field.

- Use the command line, for example:

```
VBoxManage modifyvm "VM name" --nic<x> intnet
```

Optionally, you can specify a network name with the command:

```
VBoxManage modifyvm "VM name" --intnet<x> "network name"
```

If you do not specify a network name, the network card will be attached to the network `intnet` by default.

Unless you configure the virtual network cards in the guest operating systems that are participating in the internal network to use static IP addresses, you may want to use the DHCP server that is built into Oracle VM VirtualBox to manage IP addresses for the internal network. See Section 7.42, "VBoxManage dhcpserver".

As a security measure, by default, the Linux implementation of internal networking only allows VMs running under the same user ID to establish an internal network. However, it is possible to create a shared internal networking interface, accessible by users with different user IDs.

6.7. Host-Only Networking

Host-only networking can be thought of as a hybrid between the bridged and internal networking modes. As with bridged networking, the virtual machines can talk to each other and the host as if they were connected through a physical Ethernet switch. As with internal networking, a physical networking interface need not be present, and the virtual machines cannot talk to the world outside the host since they are not connected to a physical networking interface.

When host-only networking is used, Oracle VM VirtualBox creates a new software interface on the host which then appears next to your existing network interfaces. In other words, whereas with bridged networking an existing physical interface is used to attach virtual machines to, with host-only networking a new *loopback* interface is created on the host. And whereas with internal networking, the traffic between the virtual machines cannot be seen, the traffic on the loopback interface on the host can be intercepted.

Host-only networking is particularly useful for preconfigured virtual appliances, where multiple virtual machines are shipped together and designed to cooperate. For example, one virtual machine may contain a web server and a second one a database, and since they are intended to talk to each other, the appliance can instruct Oracle VM VirtualBox to set up a host-only network for the two. A second, bridged, network would then connect the web server to the outside world to serve data to, but the outside world cannot connect to the database.

To change a virtual machine's virtual network interface to Host Only mode, do either of the following:

- Go to the **Network** page in the virtual machine's **Settings** dialog and select **Host-Only Networking**.

- On the command line, enter `VBoxManage modifyvm "vmname --nicx hostonly`. See Section 7.8, "VBoxManage modifyvm".

Before you can attach a VM to a host-only network you have to create at least one host-only interface. You can use the VirtualBox Manager for this. Choose **File**, **Preferences**, **Network**, **Host-Only Network**, **(+)Add Host-Only Network**.

Alternatively, you can use the command line:

```
VBoxManage hostonlyif create
```

See Section 7.36, "VBoxManage hostonlyif".

For host-only networking, as with internal networking, you may find the DHCP server useful that is built into Oracle VM VirtualBox. This can be enabled to then manage the IP addresses in the host-only network since otherwise you would need to configure all IP addresses statically.

- In the Oracle VM VirtualBox graphical user interface, you can configure all these items in the global settings by choosing **File**, **Preferences**, **Network**. This lists all host-only networks which are presently in use. Click on the network name and then on **Edit**. You can then modify the adapter and DHCP settings.

- Alternatively, you can use `VBoxManage dhcpserver` on the command line. See Section 7.42, "VBoxManage dhcpserver".

> **Note**
>
> On Linux and Mac OS X hosts the number of host-only interfaces is limited to 128. There is no such limit for Oracle Solaris and Windows hosts.

6.8. UDP Tunnel Networking

This networking mode enables you to interconnect virtual machines running on different hosts.

Technically this is done by encapsulating Ethernet frames sent or received by the guest network card into UDP/IP datagrams, and sending them over any network available to the host.

UDP Tunnel mode has the following parameters:

- **Source UDP port:** The port on which the host listens. Datagrams arriving on this port from any source address will be forwarded to the receiving part of the guest network card.

- **Destination address:** IP address of the target host of the transmitted data.

- **Destination UDP port:** Port number to which the transmitted data is sent.

When interconnecting two virtual machines on two different hosts, their IP addresses must be swapped. On a single host, source and destination UDP ports must be swapped.

In the following example, host 1 uses the IP address 10.0.0.1 and host 2 uses IP address 10.0.0.2. To configure using the command-line:

```
VBoxManage modifyvm "VM 01 on host 1" --nic<x> generic
VBoxManage modifyvm "VM 01 on host 1" --nicgenericdrv<x> UDPTunnel
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dest=10.0.0.2
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> sport=10001
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dport=10002
```

```
VBoxManage modifyvm "VM 02 on host 2" --nic<y> generic
VBoxManage modifyvm "VM 02 on host 2" --nicgenericdrv<y> UDPTunnel
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dest=10.0.0.1
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> sport=10002
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dport=10001
```

Of course, you can always interconnect two virtual machines on the same host, by setting the destination address parameter to 127.0.0.1 on both. It will act similarly to an internal network in this case. However, the host can see the network traffic which it could not in the normal internal network case.

> **Note**
>
> On UNIX-based hosts, such as Linux, Oracle Solaris, and Mac OS X, it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a source UDP port, the VM will refuse to start.

6.9. VDE Networking

Virtual Distributed Ethernet (VDE) is a flexible, virtual network infrastructure system, spanning across multiple hosts in a secure way. It enables L2/L3 switching, including spanning-tree protocol, VLANs, and WAN emulation. It is an optional part of Oracle VM VirtualBox which is only included in the source code.

VDE is a project developed by Renzo Davoli, Associate Professor at the University of Bologna, Italy.

The basic building blocks of the infrastructure are VDE switches, VDE plugs, and VDE wires which interconnect the switches.

The Oracle VM VirtualBox VDE driver has a single parameter: VDE network. This is the name of the VDE network switch socket to which the VM will be connected.

The following basic example shows how to connect a virtual machine to a VDE switch.

1. Create a VDE switch:

   ```
   vde_switch -s /tmp/switch1
   ```

2. Configure VMs using the command-line:

   ```
   VBoxManage modifyvm "VM name" --nic<x> generic
   ```

   ```
   VBoxManage modifyvm "VM name" --nicgenericdrv<x> VDE
   ```

   To connect to an automatically allocated switch port:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1
   ```

   To connect to a specific switch port n:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1[<n>]
   ```

   This command can be useful for VLANs.

3. (Optional) Map between a VDE switch port and a VLAN.

   Using the switch command line:

   ```
   vde$ vlan/create <VLAN>
   ```

   ```
   vde$ port/setvlan <port> <VLAN>
   ```

VDE is available on Linux and FreeBSD hosts only. It is only available if the VDE software and the VDE plugin library from the VirtualSquare project are installed on the host system.

> **Note**
>
> For Linux hosts, the shared library libvdeplug.so must be available in the search path for shared libraries.

For more information on setting up VDE networks, please see the documentation accompanying the software. See also http://wiki.virtualsquare.org/wiki/index.php/VDE_Basic_Networking.

6.10. Limiting Bandwidth for Network Input/Output

Oracle VM VirtualBox supports limiting of the maximum bandwidth used for network transmission. Several network adapters of one VM may share limits through bandwidth groups. It is possible to have more than one such limit.

Note

Oracle VM VirtualBox shapes VM traffic only in the transmit direction, delaying the packets being sent by virtual machines. It does not limit the traffic being received by virtual machines.

Limits are configured through `VBoxManage`. The following example creates a bandwidth group named Limit, sets the limit to 20 Mbps and assigns the group to the first and second adapters of the VM:

```
VBoxManage bandwidthctl "VM name" add Limit --type network --limit 20m
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 Limit
VBoxManage modifyvm "VM name" --nicbandwidthgroup2 Limit
```

All adapters in a group share the bandwidth limit, meaning that in the example above the bandwidth of both adapters combined can never exceed 20 Mbps. However, if one adapter does not require bandwidth the other can use the remaining bandwidth of its group.

The limits for each group can be changed while the VM is running, with changes being picked up immediately. The following example changes the limit for the group created in the previous example to 100 Kbps:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 100k
```

To completely disable shaping for the first adapter of VM use the following command:

```
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 none
```

It is also possible to disable shaping for all adapters assigned to a bandwidth group while VM is running, by specifying the zero limit for the group. For example, for the bandwidth group named Limit:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 0
```

6.11. Improving Network Performance

Oracle VM VirtualBox provides a variety of virtual network adapters that can be attached to the host's network in a number of ways. Depending on which types of adapters and attachments are used the network performance will be different. Performance-wise the virtio network adapter is preferable over Intel PRO/1000 emulated adapters, which are preferred over the PCNet family of adapters. Both virtio and Intel PRO/1000 adapters enjoy the benefit of segmentation and checksum offloading. Segmentation offloading is essential for high performance as it allows for less context switches, dramatically increasing the sizes of packets that cross the VM/host boundary.

Note

Neither virtio nor Intel PRO/1000 drivers for Windows XP support segmentation offloading. Therefore Windows XP guests never reach the same transmission rates as other guest types. Refer to MS Knowledge base article 842264 for additional information.

Three attachment types: Internal, Bridged, and Host-Only, have nearly identical performance. The Internal type is a little bit faster and uses less CPU cycles as the packets never reach the host's network stack.

The NAT attachment type is the slowest and most secure of all attachment types, as it provides network address translation. The generic driver attachment is special and cannot be considered as an alternative to other attachment types.

The number of CPUs assigned to VM does not improve network performance and in some cases may hurt it due to increased concurrency in the guest.

Here is a short summary of things to check in order to improve network performance:

- Whenever possible use the virtio network adapter. Otherwise, use one of the Intel PRO/1000 adapters.

- Use a Bridged attachment instead of NAT.

- Make sure segmentation offloading is enabled in the guest OS. Usually it will be enabled by default. You can check and modify offloading settings using the `ethtool` command on Linux guests.

- Perform a full detailed analysis of network traffic on the VM's network adaptor using a third party tool such as Wireshark. To do this, a promiscuous mode policy needs to be used on the VM's network adaptor. Use of this mode is only possible on the following network types: NAT Network, Bridged Adapter, Internal Network, and Host-Only Adapter.

  To setup a promiscuous mode policy, either select from the drop down list located in the **Network Settings** dialog for the network adaptor or use the command line tool `VBoxManage`. See Section 7.8, "VBoxManage modifyvm".

  Promiscuous mode policies are as follows:

  - `deny`, which hides any traffic not intended for the VM's network adaptor. This is the default setting.

  - `allow-vms`, which hides all host traffic from the VM's network adaptor, but allows it to see traffic from and to other VMs.

  - `allow-all`, which removes all restrictions. The VM's network adaptor sees all traffic.

Chapter 7 VBoxManage

7.1. Introduction

As briefly mentioned in Section 1.16, "Alternative Front-Ends", `VBoxManage` is the command-line interface to Oracle VM VirtualBox. With it, you can completely control Oracle VM VirtualBox from the command line of your host operating system. `VBoxManage` supports all the features that the graphical user interface gives you access to, but it supports a lot more than that. It exposes all the features of the virtualization engine, even those that cannot be accessed from the GUI.

You will need to use the command line if you want to do the following:

• Use a different user interface than the main GUI such as the VBoxHeadless server.

• Control some of the more advanced and experimental configuration settings for a VM.

There are two main things to keep in mind when using `VBoxManage`. First, `VBoxManage` must always be used with a specific subcommand, such as `list` or `createvm` or `startvm`. All the subcommands that `VBoxManage` supports are described in detail in Chapter 7, *VBoxManage*.

Second, most of these subcommands require that you specify a particular virtual machine after the subcommand. There are two ways you can do this:

• You can specify the VM name, as it is shown in the Oracle VM VirtualBox GUI. Note that if that name contains spaces, then you must enclose the entire name in double quotes. This is always required with command line arguments that contain spaces. For example:

```
VBoxManage startvm "Windows XP"
```

• You can specify the UUID, which is the internal unique identifier that Oracle VM VirtualBox uses to refer to the virtual machine. Assuming that the VM called "Windows XP" has the UUID shown below, the following command has the same effect as the previous example:

```
VBoxManage startvm 670e746d-abea-4ba6-ad02-2a3b043810a5
```

You can enter `VBoxManage list vms` to have all currently registered VMs listed with all their settings, including their respective names and UUIDs.

Some typical examples of how to control Oracle VM VirtualBox from the command line are listed below:

• To create a new virtual machine from the command line and immediately register it with Oracle VM VirtualBox, use `VBoxManage createvm` with the `--register` option, as follows:

```
$ VBoxManage createvm --name "SUSE 10.2" --register
VirtualBox Command Line Management Interface Version version-number
(C) 2005-2018 Oracle Corporation
All rights reserved.

Virtual machine 'SUSE 10.2' is created.
UUID: c89fc351-8ec6-4f02-a048-57f4d25288e5
Settings file: '/home/username/.config/VirtualBox/Machines/SUSE 10.2/SUSE 10.2.xml'
```

As can be seen from the above output, a new virtual machine has been created with a new UUID and a new XML settings file.

For more details, see Section 7.7, "VBoxManage createvm".

• To show the configuration of a particular VM, use `VBoxManage showvminfo`. See Section 7.5, "VBoxManage showvminfo" for details and an example.

- To change settings while a VM is powered off, use `VBoxManage modifyvm`. For example:

```
VBoxManage modifyvm "Windows XP" --memory 512
```

See also Section 7.8, "VBoxManage modifyvm".

- To change the storage configuration, such as to add a storage controller and then a virtual disk, use `VBoxManage storagectl` and `VBoxManage storageattach`. See Section 7.18, "VBoxManage storagectl" and Section 7.17, "VBoxManage storageattach".

- To control VM operation, use one of the following:

  - To start a VM that is currently powered off, use `VBoxManage startvm`. See Section 7.12, "VBoxManage startvm".

  - To pause or save a VM that is currently running or change some of its settings, use `VBoxManage controlvm`. See Section 7.13, "VBoxManage controlvm".

7.2. Commands Overview

When running `VBoxManage` without parameters or when supplying an invalid command line, the following command syntax list is shown. Note that the output will be slightly different depending on the host platform. If in doubt, check the output of `VBoxManage` for the commands available on your particular host.

```
Usage:

  VBoxManage [<general option>] <command>


General Options:

  [-v|--version]            print version number and exit
  [-q|--nologo]             suppress the logo
  [--settingspw <pw>]       provide the settings password
  [--settingspwfile <file>] provide a file containing the settings password
  [@<response-file>]        load arguments from the given response file (bourne style)


Commands:

  list [--long|-l] [--sorted|-s]        vms|runningvms|ostypes|hostdvds|hostfloppies|
                          intnets|bridgedifs|hostonlyifs|natnets|dhcpservers|
                          hostinfo|hostcpuids|hddbackends|hdds|dvds|floppies|
                          usbhost|usbfilters|systemproperties|extpacks|
                          groups|webcams|screenshotformats|cloudproviders|
                          cloudprofiles

  showvminfo              <uuid|vmname> [--details]
                          [--machinereadable]
  showvminfo              <uuid|vmname> --log <idx>

  registervm              <filename>

  unregistervm            <uuid|vmname> [--delete]

  createvm                --name <name>
                          [--groups <group>, ...]
                          [--ostype <ostype>]
                          [--register]
                          [--basefolder <path>]
                          [--uuid <uuid>]
                          [--default]
```

```
modifyvm                    <uuid|vmname>
                            [--name <name>]
                            [--groups <group>, ...]
                            [--description <desc>]
                            [--ostype <ostype>]
                            [--iconfile <filename>]
                            [--memory <memorysize in MB>]
                            [--pagefusion on|off]
                            [--vram <vramsize in MB>]
                            [--acpi on|off]
                            [--pciattach 03:04.0]
                            [--pciattach 03:04.0@02:01.0]
                            [--pcidetach 03:04.0]
                            [--ioapic on|off]
                            [--hpet on|off]
                            [--triplefaultreset on|off]
                            [--apic on|off]
                            [--x2apic on|off]
                            [--paravirtprovider none|default|legacy|minimal|
                                              hyperv|kvm]
                            [--paravirtdebug <key=value> [,<key=value> ...]]
                            [--hwvirtex on|off]
                            [--nestedpaging on|off]
                            [--largepages on|off]
                            [--vtxvpid on|off]
                            [--vtxux on|off]
                            [--pae on|off]
                            [--longmode on|off]
                            [--ibpb-on-vm-exit on|off]
                            [--ibpb-on-vm-entry on|off]
                            [--spec-ctrl on|off]
                            [--l1d-flush-on-sched on|off]
                            [--l1d-flush-on-vm-entry on|off]
                            [--mds-clear-on-sched on|off]
                            [--mds-clear-on-vm-entry on|off]
                            [--nested-hw-virt on|off]
                            [--cpu-profile "host|Intel 80[86|286|386]"]
                            [--cpuid-portability-level <0..3>]
                            [--cpuid-set <leaf[:subleaf]> <eax> <ebx> <ecx> <edx>]
                            [--cpuid-remove <leaf[:subleaf]>]
                            [--cpuidremoveall]
                            [--hardwareuuid <uuid>]
                            [--cpus <number>]
                            [--cpuhotplug on|off]
                            [--plugcpu <id>]
                            [--unplugcpu <id>]
                            [--cpuexecutioncap <1-100>]
                            [--rtcuseutc on|off]
                            [--graphicscontroller none|vboxvga|vmsvga|vboxsvga]
                            [--monitorcount <number>]
                            [--accelerate3d on|off]
                            [--accelerate2dvideo on|off]
                            [--firmware bios|efi|efi32|efi64]
                            [--chipset ich9|piix3]
                            [--bioslogofadein on|off]
                            [--bioslogofadeout on|off]
                            [--bioslogodisplaytime <msec>]
                            [--bioslogoimagepath <imagepath>]
                            [--biosbootmenu disabled|menuonly|messageandmenu]
                            [--biosapic disabled|apic|x2apic]
                            [--biossystemtimeoffset <msec>]
                            [--biospxedebug on|off]
                            [--boot<1-4> none|floppy|dvd|disk|net>]
                            [--nic<1-N> none|null|nat|bridged|intnet|hostonly|
                                        generic|natnetwork]
                            [--nictype<1-N> Am79C970A|Am79C973|
                                            82540EM|82543GC|82545EM|
```

```
                                     virtio]
                  [--cableconnected<1-N> on|off]
                  [--nictrace<1-N> on|off]
                  [--nictracefile<1-N> <filename>]
                  [--nicproperty<1-N> name=[value]]
                  [--nicspeed<1-N> <kbps>]
                  [--nicbootprio<1-N> <priority>]
                  [--nicpromisc<1-N> deny|allow-vms|allow-all]
                  [--nicbandwidthgroup<1-N> none|<name>]
                  [--bridgeadapter<1-N> none|<devicename>]
                  [--hostonlyadapter<1-N> none|<devicename>]
                  [--intnet<1-N> <network name>]
                  [--nat-network<1-N> <network name>]
                  [--nicgenericdrv<1-N> <driver>]
                  [--natnet<1-N> <network>|default]
                  [--natsettings<1-N> [<mtu>],[<socksnd>],
                                       [<sockrcv>],[<tcpsnd>],
                                       [<tcprcv>]]
                  [--natpf<1-N> [<rulename>],tcp|udp,[<hostip>],
                                <hostport>,[<guestip>],<guestport>]
                  [--natpf<1-N> delete <rulename>]
                  [--nattftpprefix<1-N> <prefix>]
                  [--nattftpfile<1-N> <file>]
                  [--nattftpserver<1-N> <ip>]
                  [--natbindip<1-N> <ip>]
                  [--natdnspassdomain<1-N> on|off]
                  [--natdnsproxy<1-N> on|off]
                  [--natdnshostresolver<1-N> on|off]
                  [--nataliasmode<1-N> default|[log],[proxyonly],
                                               [sameports]]
                  [--macaddress<1-N> auto|<mac>]
                  [--mouse ps2|usb|usbtablet|usbmultitouch]
                  [--keyboard ps2|usb]
                  [--uart<1-N> off|<I/O base> <IRQ>]
                  [--uartmode<1-N> disconnected|
                                   server <pipe>|
                                   client <pipe>|
                                   tcpserver <port>|
                                   tcpclient <hostname:port>|
                                   file <file>|
                                   <devicename>]
                  [--uarttype<1-N> 16450|16550A|16750]
                  [--lpt<1-N> off|<I/O base> <IRQ>]
                  [--lptmode<1-N> <devicename>]
                  [--guestmemoryballoon <balloonsize in MB>]
                  [--vm-process-priority default|flat|low|normal|high]
                  [--audio none|null|dsound|oss|alsa|pulse|
                           oss|pulse|coreaudio]
                  [--audioin on|off]
                  [--audioout on|off]
                  [--audiocontroller ac97|hda|sb16]
                  [--audiocodec stac9700|ad1980|stac9221|sb16]
                  [--clipboard disabled|hosttoguest|guesttohost|
                               bidirectional]
                  [--draganddrop disabled|hosttoguest|guesttohost|
                                 bidirectional]
                  [--vrde on|off]
                  [--vrdeextpack default|<name>]
                  [--vrdeproperty <name=[value]>]
                  [--vrdeport <hostport>]
                  [--vrdeaddress <hostip>]
                  [--vrdeauthtype null|external|guest]
                  [--vrdeauthlibrary default|<name>]
                  [--vrdemulticon on|off]
                  [--vrdereusecon on|off]
                  [--vrdevideochannel on|off]
                  [--vrdevideochannelquality <percent>]
```

```
                          [--usbohci on|off]
                          [--usbehci on|off]
                          [--usbxhci on|off]
                          [--usbrename <oldname> <newname>]
                          [--snapshotfolder default|<path>]
                          [--teleporter on|off]
                          [--teleporterport <port>]
                          [--teleporteraddress <address|empty>]
                          [--teleporterpassword <password>]
                          [--teleporterpasswordfile <file>|stdin]
                          [--tracing-enabled on|off]
                          [--tracing-config <config-string>]
                          [--tracing-allow-vm-access on|off]
                          [--usbcardreader on|off]
                          [--autostart-enabled on|off]
                          [--autostart-delay <seconds>]
                          [--recording on|off]
                          [--recordingscreens all|<screen ID> [<screen ID> ...]]
                          [--recordingfile <filename>]
                          [--recordingvideores <width> <height>]
                          [--recordingvideorate <rate>]
                          [--recordingvideofps <fps>]
                          [--recordingmaxtime <s>]
                          [--recordingmaxsize <MB>]
                          [--recordingopts <key=value> [,<key=value> ...]]
                          [--defaultfrontend default|<name>]

movevm                    <uuid|vmname>
                          --type basic
                          [--folder <path>]

import                    <ovfname/ovaname>
                          [--dry-run|-n]
                          [--options keepallmacs|keepnatmacs|importtovdi]
                          [more options]
                          (run with -n to have options displayed
                           for a particular OVF)

export                    <machines> --output|-o <name>.<ovf/ova/tar.gz>
                          [--legacy09|--ovf09|--ovf10|--ovf20|--opc10]
                          [--manifest]
                          [--iso]
                          [--options manifest|iso|nomacs|nomacsbutnat]
                          [--vsys <number of virtual system>]
                                  [--vmname <name>]
                                  [--product <product name>]
                                  [--producturl <product url>]
                                  [--vendor <vendor name>]
                                  [--vendorurl <vendor url>]
                                  [--version <version info>]
                                  [--description <description info>]
                                  [--eula <license text>]
                                  [--eulafile <filename>]
                          [--cloud <number of virtual system>]
                                  [--vmname <name>]
                                  [--cloudprofile <cloud profile name>]
                                  [--cloudbucket <bucket name>]
                                  [--cloudkeepobject <true/false>]
                                  [--cloudlaunchmode EMULATED|PARAVIRTUALIZED]
                                  [--cloudlaunchinstance <true/false>]
                                  [--clouddomain <domain>]
                                  [--cloudshape <shape>]
                                  [--clouddisksize <disk size in GB>]
                                  [--cloudocivcn <OCI vcn id>]
                                  [--cloudocisubnet <OCI subnet id>]
                                  [--cloudpublicip <true/false>]
                                  [--cloudprivateip <ip>]
```

```
startvm                 <uuid|vmname>...
                        [--type gui|sdl|headless|separate]
                        [-E|--putenv <NAME>[=<VALUE>]]

controlvm               <uuid|vmname>
                        pause|resume|reset|poweroff|savestate|
                        acpipowerbutton|acpisleepbutton|
                        keyboardputscancode <hex> [<hex> ...]|
                        keyboardputstring <string1> [<string2> ...]|
                        keyboardputfile <filename>|
                        setlinkstate<1-N> on|off |
                        nic<1-N> null|nat|bridged|intnet|hostonly|generic|
                                natnetwork [<devicename>] |
                        nictrace<1-N> on|off |
                        nictracefile<1-N> <filename> |
                        nicproperty<1-N> name=[value] |
                        nicpromisc<1-N> deny|allow-vms|allow-all |
                        natpf<1-N> [<rulename>],tcp|udp,[<hostip>],
                                   <hostport>,[<guestip>],<guestport> |
                        natpf<1-N> delete <rulename> |
                        guestmemoryballoon <balloonsize in MB> |
                        usbattach <uuid>|<address>
                                   [--capturefile <filename>] |
                        usbdetach <uuid>|<address> |
                        audioin on|off |
                        audioout on|off |
                        clipboard disabled|hosttoguest|guesttohost|
                                   bidirectional |
                        draganddrop disabled|hosttoguest|guesttohost|
                                   bidirectional |
                        vrde on|off |
                        vrdeport <port> |
                        vrdeproperty <name=[value]> |
                        vrdevideochannelquality <percent> |
                        setvideomodehint <xres> <yres> <bpp>
                                        [[<display>] [<enabled:yes|no> |
                                         [<xorigin> <yorigin>]]] |
                        setscreenlayout <display> on|primary <xorigin> <yorigin> <xres> <yres> <bpp> | off
                        screenshotpng <file> [display] |
                        recording on|off |
                        recording screens all|none|<screen>,[<screen>...] |
                        recording filename <file> |
                        recording videores <width>x<height> |
                        recording videorate <rate> |
                        recording videofps <fps> |
                        recording maxtime <s> |
                        recording maxfilesize <MB> |
                        setcredentials <username>
                                        --passwordfile <file> | <password>
                                        <domain>
                                        [--allowlocallogon <yes|no>] |
                        teleport --host <name> --port <port>
                                [--maxdowntime <msec>]
                                [--passwordfile <file> |
                                 --password <password>] |
                        plugcpu <id> |
                        unplugcpu <id> |
                        cpuexecutioncap <1-100>
                        webcam <attach [path [settings]]> | <detach [path]> | <list>
                        addencpassword <id>
                                        <password file>|-
                                        [--removeonsuspend <yes|no>]
                        removeencpassword <id>
                        removeallencpasswords
                        changeuartmode<1-N> disconnected|
                                            server <pipe>|
```

```
                                      client <pipe>|
                                      tcpserver <port>|
                                      tcpclient <hostname:port>|
                                      file <file>|
                                      <devicename>
                          vm-process-priority default|flat|low|normal|high

discardstate              <uuid|vmname>

adoptstate                <uuid|vmname> <state_file>

closemedium               [disk|dvd|floppy] <uuid|filename>
                          [--delete]

storageattach             <uuid|vmname>
                          --storagectl <name>
                          [--port <number>]
                          [--device <number>]
                          [--type dvddrive|hdd|fdd]
                          [--medium none|emptydrive|additions|
                                    <uuid|filename>|host:<drive>|iscsi]
                          [--mtype normal|writethrough|immutable|shareable|
                                   readonly|multiattach]
                          [--comment <text>]
                          [--setuuid <uuid>]
                          [--setparentuuid <uuid>]
                          [--passthrough on|off]
                          [--tempeject on|off]
                          [--nonrotational on|off]
                          [--discard on|off]
                          [--hotpluggable on|off]
                          [--bandwidthgroup <name>]
                          [--forceunmount]
                          [--server <name>|<ip>]
                          [--target <target>]
                          [--tport <port>]
                          [--lun <lun>]
                          [--encodedlun <lun>]
                          [--username <username>]
                          [--password <password>]
                          [--passwordfile <file>]
                          [--initiator <initiator>]
                          [--intnet]

storagectl                <uuid|vmname>
                          --name <name>
                          [--add ide|sata|scsi|floppy|sas|usb|pcie|virtio]
                          [--controller LSILogic|LSILogicSAS|BusLogic|
                                        IntelAHCI|PIIX3|PIIX4|ICH6|I82078|
                          [           USB|NVMe|VirtIO]
                          [--portcount <1-n>]
                          [--hostiocache on|off]
                          [--bootable on|off]
                          [--rename <name>]
                          [--remove]

bandwidthctl              <uuid|vmname>
                          add <name> --type disk|network
                              --limit <megabytes per second>[k|m|g|K|M|G] |
                          set <name>
                              --limit <megabytes per second>[k|m|g|K|M|G] |
                          remove <name> |
                          list [--machinereadable]
                          (limit units: k=kilobit, m=megabit, g=gigabit,
                                        K=kilobyte, M=megabyte, G=gigabyte)

showmediuminfo            [disk|dvd|floppy] <uuid|filename>
```

```
createmedium              [disk|dvd|floppy] --filename <filename>
                          [--size <megabytes>|--sizebyte <bytes>]
                          [--diffparent <uuid>|<filename>]
                          [--format VDI|VMDK|VHD] (default: VDI)]
                          [--variant Standard,Fixed,Split2G,Stream,ESX,
                                     Formatted]
                          [[--property <name>=<value>] --property <name>=<value>...

modifymedium              [disk|dvd|floppy] <uuid|filename>
                          [--type normal|writethrough|immutable|shareable|
                                  readonly|multiattach]
                          [--autoreset on|off]
                          [--property <name=[value]>]
                          [--compact]
                          [--resize <megabytes>|--resizebyte <bytes>]
                          [--move <path>]
                          [--setlocation <path>]
                          [--description <description string>]
clonemedium               [disk|dvd|floppy] <uuid|inputfile> <uuid|outputfile>
                          [--format VDI|VMDK|VHD|RAW|<other>]
                          [--variant Standard,Fixed,Split2G,Stream,ESX]
                          [--existing]

mediumproperty            [disk|dvd|floppy] set <uuid|filename>
                          <property> <value>

                          [disk|dvd|floppy] get <uuid|filename>
                          <property>

                          [disk|dvd|floppy] delete <uuid|filename>
                          <property>

encryptmedium             <uuid|filename>
                          [--newpassword <file>|-]
                          [--oldpassword <file>|-]
                          [--cipher <cipher identifier>]
                          [--newpasswordid <password identifier>]

checkmediumpwd            <uuid|filename>
                          <pwd file>|-

convertfromraw            <filename> <outputfile>
                          [--format VDI|VMDK|VHD]
                          [--variant Standard,Fixed,Split2G,Stream,ESX]
                          [--uuid <uuid>]
convertfromraw            stdin <outputfile> <bytes>
                          [--format VDI|VMDK|VHD]
                          [--variant Standard,Fixed,Split2G,Stream,ESX]
                          [--uuid <uuid>]

getextradata              global|<uuid|vmname>
                          <key>|[enumerate]

setextradata              global|<uuid|vmname>
                          <key>
                          [<value>] (no value deletes key)

setproperty               machinefolder default|<folder> |
                          hwvirtexclusive on|off |
                          vrdeauthlibrary default|<library> |
                          websrvauthlibrary default|null|<library> |
                          vrdeextpack null|<library> |
                          autostartdbpath null|<folder> |
                          loghistorycount <value>
                          defaultfrontend default|<name>
                          logginglevel <log setting>
```

	proxymode system|noproxy|manual
	proxyurl <url>
usbfilter	add <index,0-N>
	--target <uuid|vmname>|global
	--name <string>
	--action ignore|hold (global filters only)
	[--active yes|no] (yes)
	[--vendorid <XXXX>] (null)
	[--productid <XXXX>] (null)
	[--revision <IIFF>] (null)
	[--manufacturer <string>] (null)
	[--product <string>] (null)
	[--remote yes|no] (null, VM filters only)
	[--serialnumber <string>] (null)
	[--maskedinterfaces <XXXXXXXX>]
usbfilter	modify <index,0-N>
	--target <uuid|vmname>|global
	[--name <string>]
	[--action ignore|hold] (global filters only)
	[--active yes|no]
	[--vendorid <XXXX>|""]
	[--productid <XXXX>|""]
	[--revision <IIFF>|""]
	[--manufacturer <string>|""]
	[--product <string>|""]
	[--remote yes|no] (null, VM filters only)
	[--serialnumber <string>|""]
	[--maskedinterfaces <XXXXXXXX>]
usbfilter	remove <index,0-N>
	--target <uuid|vmname>|global
sharedfolder	add <uuid|vmname>
	--name <name> --hostpath <hostpath>
	[--transient] [--readonly] [--automount]
sharedfolder	remove <uuid|vmname>
	--name <name> [--transient]
guestproperty	get <uuid|vmname>
	<property> [--verbose]
guestproperty	set <uuid|vmname>
	<property> [<value> [--flags <flags>]]
guestproperty	delete|unset <uuid|vmname>
	<property>
guestproperty	enumerate <uuid|vmname>
	[--patterns <patterns>]
guestproperty	wait <uuid|vmname> <patterns>
	[--timeout <msec>] [--fail-on-timeout]
guestcontrol	<uuid|vmname> [--verbose|-v] [--quiet|-q]
	[--username <name>] [--domain <domain>]
	[--passwordfile <file> | --password <password>]
	run [common-options]
	[--exe <path to executable>] [--timeout <msec>]
	[-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
	[--ignore-operhaned-processes] [--profile]
	[--no-wait-stdout|--wait-stdout]
	[--no-wait-stderr|--wait-stderr]
	[--dos2unix] [--unix2dos]

```
                              -- <program/arg0> [argument1] ... [argumentN]]

                              start [common-options]
                              [--exe <path to executable>] [--timeout <msec>]
                              [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
                              [--ignore-operhaned-processes] [--profile]
                              -- <program/arg0> [argument1] ... [argumentN]]

                              copyfrom [common-options]
                              [--follow] [-R|--recursive]
                              <guest-src0> [guest-src1 [...]] <host-dst>

                              copyfrom [common-options]
                              [--follow] [-R|--recursive]
                              [--target-directory <host-dst-dir>]
                              <guest-src0> [guest-src1 [...]]

                              copyto [common-options]
                              [--follow] [-R|--recursive]
                              <host-src0> [host-src1 [...]] <guest-dst>

                              copyto [common-options]
                              [--follow] [-R|--recursive]
                              [--target-directory <guest-dst>]
                              <host-src0> [host-src1 [...]]

                              mkdir|createdir[ectory] [common-options]
                              [--parents] [--mode <mode>]
                              <guest directory> [...]

                              rmdir|removedir[ectory] [common-options]
                              [-R|--recursive]
                              <guest directory> [...]

                              removefile|rm [common-options] [-f|--force]
                              <guest file> [...]

                              mv|move|ren[ame] [common-options]
                              <source> [source1 [...]] <dest>

                              mktemp|createtemp[orary] [common-options]
                              [--secure] [--mode <mode>] [--tmpdir <directory>]
                              <template>

                              stat [common-options]
                              <file> [...]

guestcontrol                  <uuid|vmname> [--verbose|-v] [--quiet|-q]

                              list <all|sessions|processes|files> [common-opts]

                              closeprocess [common-options]
                              <    --session-id <ID>
                                | --session-name <name or pattern>
                              <PID1> [PID1 [...]]

                              closesession [common-options]
                              <  --all | --session-id <ID>
                                | --session-name <name or pattern> >

                              updatega|updateguestadditions|updateadditions
                              [--source <guest additions .ISO>]
                              [--wait-start] [common-options]
                              [-- [<argument1>] ... [<argumentN>]]

                              watch [common-options]
```

```
metrics                 list [*|host|<vmname> [<metric_list>]]
                                             (comma-separated)

metrics                 setup
                        [--period <seconds>] (default: 1)
                        [--samples <count>] (default: 1)
                        [--list]
                        [*|host|<vmname> [<metric_list>]]

metrics                 query [*|host|<vmname> [<metric_list>]]

metrics                 enable
                        [--list]
                        [*|host|<vmname> [<metric_list>]]

metrics                 disable
                        [--list]
                        [*|host|<vmname> [<metric_list>]]

metrics                 collect
                        [--period <seconds>] (default: 1)
                        [--samples <count>] (default: 1)
                        [--list]
                        [--detach]
                        [*|host|<vmname> [<metric_list>]]

natnetwork              add --netname <name>
                        --network <network>
                        [--enable|--disable]
                        [--dhcp on|off]
                        [--port-forward-4 <rule>]
                        [--loopback-4 <rule>]
                        [--ipv6 on|off]
                        [--port-forward-6 <rule>]
                        [--loopback-6 <rule>]

natnetwork              remove --netname <name>

natnetwork              modify --netname <name>
                        [--network <network>]
                        [--enable|--disable]
                        [--dhcp on|off]
                        [--port-forward-4 <rule>]
                        [--loopback-4 <rule>]
                        [--ipv6 on|off]
                        [--port-forward-6 <rule>]
                        [--loopback-6 <rule>]

natnetwork              start --netname <name>

natnetwork              stop --netname <name>

natnetwork              list [<pattern>]

hostonlyif              ipconfig <name>
                        [--dhcp |
                        --ip<ipv4> [--netmask<ipv4> (def: 255.255.255.0)] |
                        --ipv6<ipv6> [--netmasklengthv6<length> (def: 64)]]
                        create |
                        remove <name>

usbdevsource            add <source name>
                        --backend <backend>
                        --address <address>
usbdevsource            remove <source name>
```

VBoxManage snapshot { *uuid/vmname* }VBoxManage snapshot { *uuid/vmname*
} take { *snapshot-name* } [--description=*description*] [--live] [--uniquename
Number,Timestamp,Space,Force]VBoxManage snapshot { *uuid/vmname* } delete { *snapshot-name*
}VBoxManage snapshot { *uuid/vmname* } restore { *snapshot-name* }VBoxManage snapshot {
uuid/vmname } restorecurrent VBoxManage snapshot { *uuid/vmname* } edit { *snapshot-name* | --
current } [--description=*description*] [--name=*new-name*]VBoxManage snapshot { *uuid/vmname* }
list [[--details] | [--machinereadable]]VBoxManage snapshot { *uuid/vmname* } showvminfo { *snapshot-
name* }

VBoxManage clonevm { *vmname/uuid* } [--basefolder=*basefolder*] [--group=*group*,...] [--mode=
machine | machinechildren | all] [--name=*name*] [--options=*option*,...] [--register] [--snapshot=*snapshot-
name*] [--uuid=*uuid*]

VBoxManage mediumio {[--disk=*uuid/filename*] | [--dvd=*uuid/filename*] | [--floppy=*uuid/
filename*]} [--password-file-*/filename*] formatfat [--quick]VBoxManage mediumio {[--disk=*uuid/
filename*] | [--dvd=*uuid/filename*] | [--floppy=*uuid/filename*]} [--password-file-*/filename*] cat
[--hex] [--offset=*byte-offset*] [--size=*bytes*] [--output=*-/filename*]VBoxManage mediumio {[--
disk=*uuid/filename*] | [--dvd=*uuid/filename*] | [--floppy=*uuid/filename*]} [--password-file-*/
filename*] stream [--format=*image-format*] [--variant=*image-variant*] [--output=*-/filename*]

VBoxManage dhcpserver add { --network=*netname* | --interface=*ifname* } {--server-ip=*address*} {--
netmask=*mask*} {--lower-ip=*address*} {--upper-ip=*address*} { --enable | --disable }
[[--global] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-
opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-
time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--group=*name*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...]
[--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--incl-mac=*address*...] [--excl-
mac=*address*...] [--incl-mac-wild=*pattern*...] [--excl-mac-wild=*pattern*...] [--incl-vendor=*string*...]
[--excl-vendor=*string*...] [--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--incl-
user=*string*...] [--excl-user=*string*...] [--incl-user-wild=*pattern*...] [--excl-user-wild=*pattern*...] [--min-
lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--vm=*name/uuid*} [--nic=*1-N*] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-
opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-
lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-
address=*address*] ...]
[{--mac-address=*address*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-
no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-
lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-
address=*address*] ...]VBoxManage dhcpserver modify { --network=*netname* | --interface=*ifname*
} [--server-ip=*address*] [--lower-ip=*address*] [--upper-ip=*address*] [--netmask=*mask*] [--enable | --
disable]
[[--global] [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no
hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-
no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--
max-lease-time=*seconds*] [--remove-config] ...]
[{--group=*name*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...]
[--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--
unsupress-opt=*dhcp-opt-no*...] [--del-mac=*address*...] [--incl-mac=*address*...] [--excl-mac=*address*...]
[--del-mac-wild=*pattern*...] [--incl-mac-wild=*pattern*...] [--excl-mac-wild=*pattern*...] [--del-
vendor=*string*...] [--incl-vendor=*string*...] [--excl-vendor=*string*...] [--del-vendor-wild=*pattern*...]
[--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--del-user=*string*...] [--incl-
user=*string*...] [--excl-user=*string*...] [--del-user-wild=*pattern*...] [--incl-user-wild=*pattern*...] [--excl-
user-wild=*pattern*...] [--zap-conditions] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--
max-lease-time=*seconds*] [--remove-config] ...]

[{--vm=*name*|*uuid*} [--nic=*1-N*] [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...] [{--mac-address=*address*} [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...]`VBoxManage dhcpserver remove` { --network=*netname* | --interface=*ifname* }`VBoxManage dhcpserver restart` { --network=*netname* | --interface=*ifname* }`VBoxManage dhcpserver findlease` { --network=*netname* | --interface=*ifname* } {--mac-address=*mac*}

`VBoxManage debugvm` { *uuid*|*vmname* } dumpvmcore [--filename=*name*]`VBoxManage debugvm` { *uuid*|*vmname* } info { *item* } [*args* ...]`VBoxManage debugvm` { *uuid*|*vmname* } injectnmi `VBoxManage debugvm` { *uuid*|*vmname* } log [[--release] | [--debug]] [*group-settings* ...]`VBoxManage debugvm` { *uuid*|*vmname* } logdest [[--release] | [--debug]] [*destinations* ...]`VBoxManage debugvm` { *uuid*|*vmname* } logflags [[--release] | [--debug]] [*flags* ...]`VBoxManage debugvm` { *uuid*|*vmname* } osdetect `VBoxManage debugvm` { *uuid*|*vmname* } osinfo `VBoxManage debugvm` { *uuid*|*vmname* } osdmesg [--lines=*lines*]`VBoxManage debugvm` { *uuid*|*vmname* } getregisters [--cpu=*id*] [*reg-set.reg-name* ...]`VBoxManage debugvm` { *uuid*|*vmname* } setregisters [--cpu=*id*] [*reg-set.reg-name=value* ...]`VBoxManage debugvm` { *uuid*|*vmname* } show [[--human-readable] | [--sh-export] | [--sh-eval] | [--cmd-set]] [*settings-item* ...]`VBoxManage debugvm` { *uuid*|*vmname* } stack [--cpu=*id*]`VBoxManage debugvm` { *uuid*|*vmname* } statistics [--reset] [--descriptions] [--pattern=*pattern*]

`VBoxManage extpack install` [--replace] { *tarball* }`VBoxManage extpack uninstall` [--force] { *name* }`VBoxManage extpack cleanup`

`VBoxManage unattended detect` {--iso=*install-iso*} [--machine-readable]`VBoxManage unattended install` { *uuid*|*vmname* } {--iso=*install-iso*} [--user=*login*] [--password=*password*] [--password-file=*file*] [--full-user-name=*name*] [--key=*product-key*] [--install-additions] [--no-install-additions] [--additions-iso=*add-iso*] [--install-txs] [--no-install-txs] [--validation-kit-iso=*testing-iso*] [--locale=*ll_CC*] [--country=*CC*] [--time-zone=*tz*] [--hostname=*fqdn*] [--package-selection-adjustment=*keyword*] [--dry-run] [--auxiliary-base-path=*path*] [--image-index=*number*] [--script-template=*file*] [--post-install-template=*file*] [--post-install-command=*command*] [--extra-install-kernel-parameters=*params*] [--language=*lang*] [--start-vm=*session-type*]

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} add [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} update [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} delete `VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} show

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} list instances [--state=*string*] [--compartment-id=*string*]`VBoxManage cloud` {--provider=*name*} {--profile=*name*} list images [--state=*string*] [--compartment-id=*string*]

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance create {--domain-name=*name*} {--image-id=*unique id*} {--display-name=*name*} {--shape=*type*} {--subnet=*unique id*} {--boot-disk-size=*size in GB*} [--publicip=*true/false*] [--privateip=*IP address*] [--launch-mode=*NATIVE/EMULATED/PARAVIRTUALIZED*]`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance info {--id=*unique id*}`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance terminate {--id=*unique id*}`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance start {--id=*unique id*}`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance pause {--id=*unique id*}

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image create {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*] [--instance-id=*unique id*]`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image info {--id=*unique id*}`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image delete {--id=*unique id*}`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image import {--id=*unique id*} [--bucket-name=*name*] [--object-name=*name*]`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image export {--id=*unique id*} {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*]

Each time `VBoxManage` is invoked, only one command can be executed. However, a command might support several subcommands which then can be invoked in one single call. The following sections provide detailed reference information on the different commands.

7.3. General Options

- `-v|--version`: Show the version of this tool and exit.

- `--nologo`: Suppress the output of the logo information. This option is useful for scripts.

- `--settingspw`: Specifiy a settings password.

- `--settingspwfile`: Specify a file containing the settings password.

The settings password is used for certain settings which need to be stored in encrypted form for security reasons. At the moment, the only encrypted setting is the iSCSI initiator secret, see Section 7.17, "VBoxManage storageattach". As long as no settings password is specified, this information is stored in *plain text*. After using the `--settingspw|--settingspwfile` option once, it must be always used. Otherwise, the encrypted setting cannot be unencrypted.

7.4. VBoxManage list

The `list` command gives relevant information about your system and information about Oracle VM VirtualBox's current settings.

The following subcommands are available with `VBoxManage list`:

- `vms`: Lists all virtual machines currently registered with Oracle VM VirtualBox. By default this displays a compact list with each VM's name and UUID. If you also specify `--long` or `-l`, this will be a detailed list as with the `showvminfo` command, see Section 7.5, "VBoxManage showvminfo".

- `runningvms`: Lists all currently running virtual machines by their unique identifiers (UUIDs) in the same format as with `vms`.

- `ostypes`: Lists all guest operating systems presently known to Oracle VM VirtualBox, along with the identifiers used to refer to them with the `modifyvm` command.

- `hostdvds`, `hostfloppies`: Lists the DVD, floppy, bridged networking, and host-only networking interfaces on the host, along with the name used to access them from within Oracle VM VirtualBox.

- `intnets`: Displays information about the internal networks.

- `bridgedifs`, `hostonlyifs`, `natnets`, `dhcpservers`: Lists the bridged network interfaces, host-only network interfaces, NAT network interfaces, and DHCP servers currently available on the host. See Chapter 6, *Virtual Networking*.

- `hostinfo`: Displays information about the host system, such as CPUs, memory size, and operating system version.

- `hostcpuids`: Lists the CPUID parameters for the host CPUs. This can be used for a more fine grained analyis of the host's virtualization capabilities.

- `hddbackends`: Lists all known virtual disk back-ends of Oracle VM VirtualBox. For each such format, such as VDI, VMDK, or RAW, this subcommand lists the back-end's capabilities and configuration.

- `hdds`, `dvds`, `floppies`: Shows information about virtual disk images currently in use by Oracle VM VirtualBox, including all their settings, the unique identifiers (UUIDs) associated with them by Oracle VM VirtualBox and all files associated with them. This is the command-line equivalent of the Virtual Media Manager. See Section 5.3, "The Virtual Media Manager".

- `usbhost`: Shows information about USB devices attached to the host, including information useful for constructing USB filters and whether they are currently in use by the host.

- `usbfilters`: Lists all global USB filters registered with Oracle VM VirtualBox and displays the filter parameters. Global USB filters are for devices which are accessible to all virtual machines.

- `systemproperties`: Displays some global Oracle VM VirtualBox settings, such as minimum and maximum guest RAM and virtual hard disk size, folder settings and the current authentication library in use.

- `extpacks`: Displays all Oracle VM VirtualBox extension packs that are currently installed. See Section 1.5, "Installing Oracle VM VirtualBox and Extension Packs" and Section 7.41, "VBoxManage extpack".

- `groups`: Displays details of the VM Groups. See Section 1.9, "Using VM Groups".

- `webcams`: Displays a list of webcams attached to the running VM. The output format is a list of absolute paths or aliases that were used for attaching the webcams to the VM using the webcam attach command.

- `screenshotformats`: Displays a list of available screenshot formats.

- `cloudproviders`: Displays a list of cloud providers that are supported by Oracle VM VirtualBox. Oracle Cloud Infrastructure is an example of a cloud provider.

- `cloudprofiles`: Displays a list of cloud profiles that have been configured.

  Cloud profiles are used when exporting VMs to a cloud service. See Section 1.14.4, "Exporting an Appliance to Oracle Cloud Infrastructure".

7.5. VBoxManage showvminfo

The `showvminfo` command shows information about a particular virtual machine. This is the same information as `VBoxManage list vms --long` would show for all virtual machines.

You will see information as shown in the following example.

```
$ VBoxManage showvminfo "Windows XP"
VirtualBox Command Line Management Interface Version version-number
(C) 2005-2018 Oracle Corporation
All rights reserved.

Name:            Windows XP
Guest OS:        Other/Unknown
UUID:            1bf3464d-57c6-4d49-92a9-a5cc3816b7e7
Config file:     /home/username/.config/VirtualBox/Machines/Windows XP/Windows XP.xml
Memory size:     512MB
VRAM size:       12MB
```

```
Number of CPUs:   2
Boot menu mode:   message and menu
Boot Device (1): DVD
Boot Device (2): HardDisk
Boot Device (3): Not Assigned
Boot Device (4): Not Assigned
ACPI:             on
IOAPIC:           on
...
```

Use the `--machinereadable` option to produce the same output, but in machine readable format with a property=value string on each line. For example:

```
...
groups="/"
ostype="Oracle (64-bit)"
UUID="457af700-bc0a-4258-aa3c-13b03da171f2"
...
```

7.6. VBoxManage registervm/unregistervm

The `registervm` command enables you to import a virtual machine definition in an XML file into Oracle VM VirtualBox. The machine must not conflict with one already registered in Oracle VM VirtualBox and it may not have any hard or removable disks attached. It is advisable to place the definition file in the machines folder before registering it.

Note

When creating a new virtual machine with `VBoxManage createvm`, as shown in Section 7.7, "VBoxManage createvm", you can directly specify the `--register` option to avoid having to register it separately.

The `unregistervm` command unregisters a virtual machine. If `--delete` is also specified, the following files will also be deleted automatically:

- All hard disk image files, including differencing files, which are used by the machine and not shared with other machines.

- Saved state files that the machine created. One if the machine was in Saved state and one for each online snapshot.

- The machine XML file and its backups.

- The machine log files.

- The machine directory, if it is empty after having deleted all of the above files.

7.7. VBoxManage createvm

The `VBoxManage createvm` command creates a new XML virtual machine definition file.

You must specify the name of the VM by using `--name` *name*. This name is used by default as the file name of the settings file that has the `.xml` extension and the machine folder, which is a subfolder of the `.config/VirtualBox/Machines` folder. Note that the machine folder path name varies based on the OS type and the Oracle VM VirtualBox version.

Ensure that the VM name conforms to the host OS's file name requirements. If you later rename the VM, the file and folder names will be updated to match the new name automatically.

The `--basefolder path` option specifies the machine folder path name. Note that the names of the file and the folder do not change if you rename the VM.

The `--group group-ID, ...` option assigns the VM to the specified groups. Note that group IDs always start with `/` so that they can be nested. By default, each VM is assigned membership to the `/` group.

The `--ostype ostype` option specifies the guest OS to run in the VM. Run the `VBoxManage list ostypes` command to see the available OS types.

The `--uuid uuid` option specifies the universal unique identifier (UUID) of the VM. The UUID must be unique within the namespace of the host or of its VM group memberships. By default, the `VBoxManage` command automatically generates the UUID.

The `--default` option applies a default hardware configuration for the specified guest OS. By default, the VM is created with minimal hardware.

The `--register` option registers the VM with your Oracle VM VirtualBox installation. By default, the `VBoxManage createvm` command creates only the XML configuration for the VM but does not registered the VM. If you do not register the VM at creation, you can run the `VBoxManage registervm` command after you create the VM.

7.8. VBoxManage modifyvm

This command changes the properties of a registered virtual machine which is not running. Most of the properties that this command makes available correspond to the VM settings that Oracle VM VirtualBox graphical user interface displays in each VM's **Settings** dialog. These are described in Chapter 3, *Configuring Virtual Machines*. However, some of the more advanced settings are only available through the `VBoxManage` interface.

These commands require that the machine is powered off, neither running nor in a Saved state. Some machine settings can also be changed while a machine is running. Those settings will then have a corresponding subcommand with the `VBoxManage controlvm` subcommand. See Section 7.13, "VBoxManage controlvm".

7.8.1. General Settings

The following general settings are available through `VBoxManage modifyvm`:

- `--name <name>`: Changes the VM's name and can be used to rename the internal virtual machine files, as described in Section 7.7, "VBoxManage createvm".

- `--groups <group>, ...`: Changes the group membership of a VM. Groups always start with a `/` and can be nested. By default VMs are in group `/`.

- `--description <desc>`: Changes the VM's description, which is a way to record details about the VM in a way which is meaningful for the user. The GUI interprets HTML formatting, the command line allows arbitrary strings potentially containing multiple lines.

- `--ostype <ostype>`: Specifies what guest operating system is supposed to run in the VM. To learn about the various identifiers that can be used here, use `VBoxManage list ostypes`.

- `--iconfile <filename>`: Specifies the absolute path on the host file system for the Oracle VM VirtualBox icon to be displayed in the VM.

- `--memory <memorysize>`: Sets the amount of RAM, in MB, that the virtual machine should allocate for itself from the host. See Section 1.7, "Creating Your First Virtual Machine".

- `--pagefusion on|off`: Enables and disables the Page Fusion feature. Page Fusion is disabled by default. The Page Fusion feature minimises memory duplication between VMs with similar configurations running on the same host. See Section 4.10.2, "Page Fusion".

- `--vram <vramsize>`: Sets the amount of RAM that the virtual graphics card should have. See Section 3.6, "Display Settings".

- `--acpi on|off` and `--ioapic on|off`: Determines whether the VM has ACPI and I/O APIC support. See Section 3.5.1, "Motherboard Tab".

- `--pciattach <host PCI address [@ guest PCI bus address]>`: Attaches a specified PCI network controller on the host to a specified PCI bus on the guest. See PCI Passthrough.

- `--pcidetach <host PCI address>`: Detaches a specified PCI network controller on the host from the attached PCI bus on the guest. See PCI Passthrough.

- `--hardwareuuid <uuid>`: The UUID presented to the guest through memory tables (DMI/SMBIOS), hardware, and guest properties. By default this is the same as the VM UUID. This setting is useful when cloning a VM. Teleporting takes care of this automatically.

- `--cpus <cpucount>`: Sets the number of virtual CPUs for the virtual machine, see Section 3.5.2, "Processor Tab". If CPU hot-plugging is enabled, this then sets the *maximum* number of virtual CPUs that can be plugged into the virtual machines.

- `--cpuhotplug on|off`: Enables CPU hot-plugging. When enabled, virtual CPUs can be added to and removed from a virtual machine while it is running. See CPU Hot-Plugging.

- `--plugcpu|unplugcpu <id>`: If CPU hot-plugging is enabled, this setting adds or removes a virtual CPU on the virtual machine. `<id>` specifies the index of the virtual CPU to be added or removed and must be a number from 0 to the maximum number of CPUs configured with the `--cpus` option. CPU 0 can never be removed.

- `--cpuexecutioncap <1-100>`: Controls how much CPU time a virtual CPU can use. A value of 50 implies a single virtual CPU can use up to 50% of a single host CPU.

- `--pae on|off`: Enables and disables PAE. See Section 3.5.2, "Processor Tab".

- `--longmode on|off`: Enables and disables long mode. See Section 3.5.2, "Processor Tab".

- `--spec-ctrl on|off`: Enables and disables the exposure of speculation control interfaces to the guest, provided they are available on the host. Depending on the host CPU and workload, enabling speculation control may significantly reduce performance.

- `--cpu-profile <host|intel 80[86|286|386]>`: Enables specification of a profile for guest CPU emulation. Specify either one based on the host system CPU (host), or one from a number of older Intel Micro-architectures: 8086, 80286, 80386.

- `--hpet on|off`: Enables and disables a High Precision Event Timer (HPET) which can replace the legacy system timers. This is turned off by default. Note that Windows supports a HPET only from Vista onwards.

- `--hwvirtex on|off`: Enables and disables the use of hardware virtualization extensions, such as Intel VT-x or AMD-V, in the processor of your host system. See Hardware vs. Software Virtualization.

- `--triplefaultreset on|off`: Enables resetting of the guest instead of triggering a Guru Meditation. Some guests raise a triple fault to reset the CPU so sometimes this is desired behavior. Works only for non-SMP guests.

- `--apic on|off`: Enables and disables I/O APIC. With I/O APIC, operating systems can use more than 16 interrupt requests (IRQs) thus avoiding IRQ sharing for improved reliability. This setting is enabled by default. See Section 3.5.1, "Motherboard Tab".

- `--x2apic on|off`: Enables and disables CPU x2APIC support. CPU x2APIC support helps operating systems run more efficiently on high core count configurations, and optimizes interrupt distribution in virtualized environments. This setting is enabled by default. Disable this setting when using host or guest operating systems that are incompatible with x2APIC support.

- `--paravirtprovider none|default|legacy|minimal|hyperv|kvm`: Specifies which paravirtualization interface to provide to the guest operating system. Specifying `none` explicitly turns off exposing any paravirtualization interface. The option `default` selects an appropriate interface when starting the VM, depending on the guest OS type. This is the default option chosen when creating new VMs. The `legacy` option is used for VMs which were created with older Oracle VM VirtualBox versions and will pick a paravirtualization interface when starting the VM with Oracle VM VirtualBox 5.0 and newer. The `minimal` provider is mandatory for Mac OS X guests. `kvm` and `hyperv` are recommended for Linux and Windows guests respectively. These options are explained in Paravirtualization Providers.

- `--paravirtdebug <keyword=value> [,<keyword=value> ...]`: Specifies debugging options specific to the paravirtualization provider configured for this VM. See the provider specific options in Paravirtualized Debugging for a list of supported keyword-value pairs for each provider.

- `--nestedpaging on|off`: If hardware virtualization is enabled, this additional setting enables or disables the use of the nested paging feature in the processor of your host system. See Hardware vs. Software Virtualization and CVE-2018-3646.

- `--largepages on|off`: If hardware virtualization *and* nested paging are enabled, for Intel VT-x only, an additional performance improvement of up to 5% can be obtained by enabling this setting. This causes the hypervisor to use large pages to reduce TLB use and overhead.

- `--vtxvpid on|off`: If hardware virtualization is enabled, for Intel VT-x only, this additional setting enables or disables the use of the tagged TLB (VPID) feature in the processor of your host system. See Hardware vs. Software Virtualization.

- `--vtxux on|off`: If hardware virtualization is enabled, for Intel VT-x only, this setting enables or disables the use of the unrestricted guest mode feature for executing your guest.

- `--accelerate3d on|off`: If the Guest Additions are installed, this setting enables or disables hardware 3D acceleration. See Section 4.5.1, "Hardware 3D Acceleration (OpenGL and Direct3D 8/9)".

- `--accelerate2dvideo on|off`: If the Guest Additions are installed, this setting enables or disables 2D video acceleration. See Section 4.5.2, "Hardware 2D Video Acceleration for Windows Guests".

- `--chipset piix3|ich9`: By default, Oracle VM VirtualBox emulates an Intel PIIX3 chipset. Usually there is no reason to change the default setting unless this is required to relax some of its constraints. See Section 3.5.1, "Motherboard Tab".

- You can influence the BIOS logo that is displayed when a virtual machine starts up with a number of settings. By default, an Oracle VM VirtualBox logo is displayed.

  With `--bioslogofadein on|off` and `--bioslogofadeout on|off`, you can determine whether the logo should fade in and out, respectively.

  With `--bioslogodisplaytime <msec>` you can set how long the logo should be visible, in milliseconds.

  With `--bioslogoimagepath <imagepath>` you can replace the image that is shown with your own logo. The image must be an uncompressed 256 color BMP file without color space information (Windows 3.0 format). The image must not be bigger than 640 x 480.

- `--biosbootmenu disabled|menuonly|messageandmenu`: Specifies whether the BIOS enables the user to select a temporary boot device. The `menuonly` option suppresses the message, but the user can still press F12 to select a temporary boot device.

- `--biosapic x2apic|apic|disabled`: Specifies the firmware APIC level to be used. Options are: x2apic, apic or disabled (no apic or x2apic) respectively.

  Note that if x2apic is specified and x2APIC is unsupported by the VCPU, biosapic downgrades to apic, if supported. Otherwise biosapic downgrades to disabled. Similarly, if apic is specified, and APIC is unsupported, a downgrade to disabled results.

- `--biossystemtimeoffset <ms>`: Specifies a fixed time offset, in milliseconds, of the guest relative to the host time. If the offset is positive, the guest time runs ahead of the host time.

- `--biospxedebug on|off`: Enables additional debugging output when using the Intel PXE boot ROM. The output is written to the release log file. See Collecting Debugging Information.

- `--boot<1-4> none|floppy|dvd|disk|net`: Specifies the boot order for the virtual machine. There are four *slots*, which the VM will try to access from 1 to 4, and for each of which you can set a device that the VM should attempt to boot from.

- `--rtcuseutc on|off`: Sets the real-time clock (RTC) to operate in UTC time. See Section 3.5.1, "Motherboard Tab".

- `--graphicscontroller none|vboxvga|vmsvga|vboxsvga`: Specifies the use of a graphics controller, with an option to choose a specific type. See Section 3.6.1, "Screen Tab".

- `--snapshotfolder default|<path>`: Specifies the folder where snapshots are kept for a virtual machine.

- `--firmware bios|efi|efi32|efi64`: Specifies the firmware to be used to boot the VM: Available options are: BIOS, or one of the EFI options: efi, efi32, or efi64. Use EFI options with care.

- `--guestmemoryballoon <size>` Sets the default size of the guest memory balloon. This is the memory allocated by the Oracle VM VirtualBox Guest Additions from the guest operating system and returned to the hypervisor for reuse by other virtual machines. `<size>` must be specified in megabytes. The default size is 0 megabytes. See Section 4.10.1, "Memory Ballooning".

- `--defaultfrontend default|<name>`: Specifies the default frontend to be used when starting this VM. See Section 7.12, "VBoxManage startvm".

- `--vm-process-priority default|flat|low|normal|high`: Specifies the priority scheme of the VM process to be used when starting this VM and during VM execution. See Section 7.12, "VBoxManage startvm".

7.8.2. Networking Settings

The following networking settings are available through `VBoxManage modifyvm`. With all these settings, the decimal number directly following the option name, 1-N in the list below, specifies the virtual network adapter whose settings should be changed.

- `--nic<1-N> none|null|nat|natnetwork|bridged|intnet|hostonly|generic`: Configures the type of networking for each of the VM's virtual network cards. Options are: not present (`none`), not connected to the host (`null`), use network address translation (`nat`), use the new network address translation engine (`natnetwork`), bridged networking (`bridged`), or use internal networking (`intnet`), host-only networking (`hostonly`), or access rarely used sub-modes (`generic`). These options correspond to the modes described in Section 6.2, "Introduction to Networking Modes".

- `--nictype<1-N> Am79C970A|Am79C973|82540EM|82543GC|82545EM|virtio`: Enables you to specify the networking hardware that Oracle VM VirtualBox presents to the guest for a specified VM virtual network card. See Section 6.1, "Virtual Networking Hardware".

- `--cableconnected<1-N> on|off`: Enables you to temporarily disconnect a virtual network interface, as if a network cable had been pulled from a real network card. This might be useful, for example for resetting certain software components in the VM.

- With the `nictrace` options, you can optionally trace network traffic by dumping it to a file, for debugging purposes.

  With `--nictrace<1-N> on|off`, you can enable network tracing for a particular virtual network card.

  If enabled, you must specify with `--nictracefile<1-N> <filename>` the absolute path of the file the trace should be logged to.

- `--nicproperty<1-N> <paramname>="paramvalue"`: This option, in combination with `nicgenericdrv` enables you to pass parameters to rarely-used network backends.

  These parameters are backend engine-specific, and are different between UDP Tunnel and the VDE backend drivers. For examples, see Section 6.8, "UDP Tunnel Networking".

- `--nicspeed<1-N> <kbps>`: Only has an effect if generic networking has been enabled for a particular virtual network card. See the `--nic` option. This mode enables access to rarely used networking sub-modes, such as VDE network or UDP Tunnel. This option specifies the throughput rate in KBps.

- `--nicbootprio<1-N> <priority>`: Specifies the order in which NICs are tried for booting over the network, using PXE. The priority is an integer in the 0 to 4 range. Priority 1 is the highest, priority 4 is low. Priority 0, which is the default unless otherwise specified, is the lowest.

  Note that this option only has an effect when the Intel PXE boot ROM is used.

- `--nicpromisc<1-N> deny|allow-vms|allow-all`: Enables you to specify how promiscuous mode is handled for the specified VM virtual network card. This setting is only relevant for bridged networking. `deny`, the default setting, hides any traffic not intended for the VM. `allow-vms` hides all host traffic from the VM, but allows the VM to see traffic to and from other VMs. `allow-all` removes this restriction completely.

- `--nicbandwidthgroup<1-N> none|<name>`: Adds and removes an assignment of a bandwidth group for the specified virtual network interface. Specifying `none` removes any current bandwidth group assignment from the specified virtual network interface. Specifying `<name>` adds an assignment of a bandwidth group to the specified virtual network interface.

  See Section 6.10, "Limiting Bandwidth for Network Input/Output".

- `--bridgeadapter<1-N> none|<devicename>`: Only has an effect if bridged networking has been enabled for a virtual network card. See the `--nic` option. Use this option to specify which host interface the given virtual network interface will use. See Section 6.5, "Bridged Networking".

- `--hostonlyadapter<1-N> none|<devicename>`: Only has an effect if host-only networking has been enabled for a virtual network card. See the `--nic` option. Use this option to specify which host-only networking interface the given virtual network interface will use. See Section 6.7, "Host-Only Networking".

- `--intnet<1-N> network`: Only has an effect if internal networking has been enabled for a virtual network card. See the `--nic` option. Use this option to specify the name of the internal network. See Section 6.6, "Internal Networking".

- `--nat-network<1-N> <network name>`: If the networking type is set to `natnetwork`, not `nat`, then this setting specifies the name of the NAT network this adapter is connected to. Optional.

- `--nicgenericdrv<1-N> <backend driver>`: Only has an effect if generic networking has been enabled for a virtual network card. See the `--nic` option. This mode enables you to access rarely used networking sub-modes, such as VDE network or UDP Tunnel.

- `--macaddress<1-N> auto|<mac>`: With this option you can set the MAC address of a particular network adapter on the VM. Normally, each network adapter is assigned a random address by Oracle VM VirtualBox at VM creation.

7.8.2.1. NAT Networking Settings

The following NAT networking settings are available through `VBoxManage modifyvm`. With all these settings, the decimal number directly following the option name, 1-N in the list below, specifies the virtual network adapter whose settings should be changed.

- `--natnet<1-N> <network>|default`: If the networking type is set to `nat`, not `natnetwork`, then this setting specifies the IP address range to be used for this network. See Fine Tuning the Oracle VM VirtualBox NAT Engine.

- `--natpf<1-N> [<name>],tcp|udp,[<hostip>],<hostport>,[<guestip>], <guestport>`: Defines a NAT port-forwarding rule. See Section 6.3.1, "Configuring Port Forwarding with NAT".

- `--natpf<1-N> delete <name>`: Deletes a NAT port-forwarding rule. See Section 6.3.1, "Configuring Port Forwarding with NAT".

- `--nattftpprefix<1-N> <prefix>`: Defines a prefix for the built-in TFTP server. For example, where the boot file is located. See Section 6.3.2, "PXE Booting with NAT" and Configuring the Boot Server (Next Server) of a NAT Network Interface.

- `--nattftpfile<1-N> <bootfile>`: Defines the TFT boot file. See Configuring the Boot Server (Next Server) of a NAT Network Interface.

- `--nattftpserver<1-N> <tftpserver>`: Defines the TFTP server address to boot from. See Configuring the Boot Server (Next Server) of a NAT Network Interface.

- `--nattbindip<1-N> <ip;>`: Oracle VM VirtualBox's NAT engine normally routes TCP/IP packets through the default interface assigned by the host's TCP/IP stack. Use this setting to instruct the NAT engine to bind to a specified IP address instead. See Tuning TCP/IP Buffers for NAT.

- `--natdnspassdomain<1-N> on|off`: Specifies whether the built-in DHCP server passes the domain name for network name resolution.

- `--natdnsproxy<1-N> on|off`: Makes the NAT engine proxy all guest DNS requests to the host's DNS servers. See Enabling DNS Proxy in NAT Mode.

- `--natdnshostresolver<1-N> on|off`: Makes the NAT engine use the host's resolver mechanisms to handle DNS requests. See Enabling DNS Proxy in NAT Mode.

- `--natsettings<1-N> [<mtu>],[<socksnd>],[<sockrcv>],[<tcpsnd>], [<tcprcv>]`: Controls several NAT settings. See Tuning TCP/IP Buffers for NAT.

- `--nataliasmode<1-N> default|[log],[proxyonly],[sameports]`: Defines behaviour of the NAT engine core: log - enables logging, proxyonly - switches off aliasing mode and makes NAT transparent, sameports - enforces the NAT engine to send packets through the same port as they originated on, default - disable all aliasing modes. See Configuring Aliasing of the NAT Engine.

7.8.3. Miscellaneous Settings

The following hardware settings, such as serial port, audio, clipboard, drag and drop, monitor, and USB settings are available through `VBoxManage modifyvm`:

- `--mouse <ps2|usb|usbtablet|usbmultitouch>`: Specifies the mode of the mouse to be used in the VM. Available options are: ps2, usb, usbtablet, usbmultitouch.

- `--keyboard <ps2|usb>`: Specifies the mode of the keyboard to be used in the VM. Available options are: ps2, usb.

- `--uart<1-N> off|<I/O base> <IRQ>`: Configures virtual serial ports for the VM. See Section 3.10, "Serial Ports".

- `--uartmode<1-N> <arg>`: Controls how Oracle VM VirtualBox connects a given virtual serial port, configured with the `--uartX` setting, to the host on which the virtual machine is running. As described in Section 3.10, "Serial Ports", for each such port, you can specify `<arg>` as one of the following options:

  - `disconnected`: Even though the serial port is shown to the guest, it has no "other end". This is like a real COM port without a cable.

  - `server <pipename>`: On a Windows host, this tells Oracle VM VirtualBox to create a named pipe on the host named `<pipename>` and connect the virtual serial device to it. Note that Windows requires that the name of a named pipe begins with `\\.\pipe\`.

    On a Linux host, instead of a named pipe, a local domain socket is used.

  - `client <pipename>`: Operates as for `server`, except that the pipe, or local domain socket, is not created by Oracle VM VirtualBox but is assumed to exist already.

  - `tcpserver <port>`: Configures Oracle VM VirtualBox to create a TCP socket on the host with TCP `<port>` and connect the virtual serial device to it. Note that UNIX-like systems require ports over 1024 for normal users.

  - `tcpclient <hostname:port>`: Operates as for `tcpserver`, except that the TCP socket is not created by Oracle VM VirtualBox, but is assumed to exist already.

  - `file <file>`: Redirects the serial port output to a raw file <file> specified by its absolute path on the host file system.

  - `<devicename>`: If, instead of the above options, the device name of a physical hardware serial port of the host is specified, the virtual serial port is connected to that hardware port. On a Windows host, the

device name will be a COM port such as `COM1`. On a Linux host, the device name will be `/dev/ttyS0` or similar. This enables you to wire up a real serial port to a virtual machine.

- `uarttype <1-N> 16450|16550A|16750`: Configures the UART type for a virtual serial port. The default UART type is 16550A.

- `--lptmode<1-N> <Device>`: Specifies the Device Name of the parallel port that the Parallel Port feature will be using. Use this *before* `--lpt`. This feature depends on the host operating system. For Windows hosts, use a device name such as lpt1. On Linux hosts, use a device name such as /dev/lp0.

- `--lpt<1-N> <I/O base> <IRQ>`: Specifies the I/O address of the parallel port and the IRQ number that the Parallel Port feature will be using. Optional. Use this *after* `--lptmod`. I/O base address and IRQ are the values that guest sees. For example, the values avalable under guest Device Manager.

- `--audio none|null|dsound|oss|alsa|pulse|coreaudio`: Specifies whether the VM should have audio support, and if so, which type. The list of supported audio types depends on the host and can be determined with `VBoxManage modifyvm`.

- `--audiocontroller ac97|hda|sb16`: Specifies the audio controller to be used with the VM.

- `--audiocodec stac9700|ad1980|stac9221|sb16`: Specifies the audio codec to be used with the VM.

- `--audioin on`: Specifies whether capturing audio from the host is enabled or disabled.

- `--audioout on`: Specifies whether audio playback from the guest is enabled or disabled.

- `--clipboard disabled|hosttoguest|guesttohost|bidirectional`: Configues how the guest or host operating system's clipboard should be shared with the host or guest. See Section 3.4, "General Settings". This setting requires that the Guest Additions be installed in the virtual machine.

- `--draganddrop disabled|hosttoguest|guesttohost|bidirectional`: Specifies the drag and drop mode to use between the host and the virtual machine. See Section 4.4, "Drag and Drop". This requires that the Guest Additions be installed in the virtual machine.

- `--monitorcount <count>`: Enables multi-monitor support. See Section 3.6, "Display Settings".

- `--usb on|off`: Enables and disables the VM's virtual USB controller. See Section 3.11.1, "USB Settings".

- `--usbehci on|off`: Enables and disables the VM's virtual USB 2.0 controller. See Section 3.11.1, "USB Settings".

- `--usbxhci on|off`: Enables and disables the VM's virtual USB 3.0 controller. See Section 3.11.1, "USB Settings".

- `--usbrename <oldname> <newname>`: Enables renaming of the VM's virtual USB controller from <oldname> to <newname>.

7.8.4. Recording Settings

The `VBoxManage modifyvm` command enables you to modify recording settings for video recording, audio recording, or both.

Use the following options to update the recording settings:

- `--recording on|off` enables or disables the recording of a VM session into a WebM/VP8 file. When this option value is `on`, recording begins when the VM session starts.

- `--recordingscreens all|screen-ID [screen-ID ...]` enables you to specify which VM screens to record. The recording for each screen that you specify is saved to its own file.

- `--recordingfile filename` specifies the file in which to save the recording.

- `--recordingmaxsize MB` specifies the maximum size of the recorded video file in megabytes. The recording stops when the file reaches the specified size. If this value is zero, the recording continues until you stop the recording.

- `--recordingmaxtime seconds` specifies the maximum amount time to record in seconds. The recording stops after the specified number of seconds elapses. If this value is zero, the recording continues until you stop the recording.

- `--recordingopts keyword=value[,keyword=value ...]` specifies additional video-recording options in a comma-separated keyword-value format. For example, `foo=bar,a=b`.

  Only use this option only if you are an advanced user. For information about keywords, see *Oracle VM VirtualBox Programming Guide and Reference*.

- `--recordingvideofps fps` specifies the maximum number of video frames per second (FPS) to record. Frames that have a higher frequency are skipped. Increasing this value reduces the number of skipped frames and increases the file size.

- `--recordingvideorate bit-rate` specifies the bit rate of the video in kilobits per second. Increasing this value improves the appearance of the video at the cost of an increased file size.

- `--recordingvideores widthxheight` specifies the video resolution of the recorded video in pixels.

7.8.5. Remote Machine Settings

The following settings that affect remote machine behavior are available through `VBoxManage modifyvm`:

- `--vrde on|off`: Enables and disables the VirtualBox Remote Desktop Extension (VRDE) server.

- `--vrdeproperty "TCP/Ports|Address=<value>"`: Sets the port numbers and IP address on the VM that the VRDE server can bind to.

  - For TCP/Ports, <value> should be a port or a range of ports that the VRDE server can bind to. `default` or `0` means port 3389, the standard port for RDP. See the description for the `--vrdeport` option in Section 7.8.5, "Remote Machine Settings".

  - For TCP/Address, <value> should be the IP address of the host network interface that the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface. See the description for the `--vrdeaddress` option in Section 7.8.5, "Remote Machine Settings".

- `--vrdeproperty "VideoChannel/Enabled|Quality|DownscaleProtection=<value>"`: Sets the VRDP video redirection properties.

  - For VideoChannel/Enabled, <value> can be set to "1", switching the VRDP video channel on. See VRDP Video Redirection.

  - For VideoChannel/Quality, <value> should be set between 10 and 100% inclusive, representing a JPEG compression level on the VRDE server video channel. Lower values mean lower quality but higher compression. See VRDP Video Redirection.

- For VideoChannel/DownscaleProtection, <value> can be set to "1" to enable the videochannel downscale protection feature. When enabled, if a video's size equals the shadow buffer size, then it is regarded as a full screen video, and is displayed. But if its size is between fullscreen and the downscale threshold then it is *not* displayed, as it could be an application window, which would be unreadable when downscaled. When the downscale protection feature is disabled, an attempt is always made to display videos.

- `--vrdeproperty "Client/DisableDisplay|DisableInput|DisableAudio|DisableUSB=1"`: Disables one of the VRDE server features: Display, Input, Audio or USB respectively. To reenable a feature, use "Client/DisableDisplay=" for example. See VRDP Customization.

- `--vrdeproperty "Client/DisableClipboard|DisableUpstreamAudio=1"`: Disables one of the VRDE server features: Clipboard or UpstreamAudio respectively. To reenable a feature, use "Client/DisableClipboard=" for example. See VRDP Customization.

- `--vrdeproperty "Client/DisableRDPDR=1"`: Disables the VRDE server feature: RDP device redirection for smart cards. To reenable this feature, use "Client/DisableRDPR=".

- `--vrdeproperty "H3DRedirect/Enabled=1"`: Enables the VRDE server feature: 3D redirection. To disable this feature, use "H3DRedirect/Enabled=".

- `--vrdeproperty "Security/Method|ServerCertificate|ServerPrivateKey|CACertificate=<value>"`: Sets the desired security method and path of server certificate, path of server private key, path of CA certificate, that are used for a connection.

  - `--vrdeproperty "Security/Method=<value>"` sets the desired security method, which is used for a connection. Valid values are:

    - `Negotiate`: Both Enhanced (TLS) and Standard RDP Security connections are allowed. The security method is negotiated with the client. This is the default setting.

    - `RDP`: Only Standard RDP Security is accepted.

    - `TLS`: Only Enhanced RDP Security is accepted. The client must support TLS.

    See RDP Encryption.

  - `--vrdeproperty "Security/ServerCertificate=<value>"` where <value> is the absolute path of the server certificate. See RDP Encryption.

  - `--vrdeproperty "Security/ServerPrivateKey=<value>"` where <value> is the absolute path of the server private key. See RDP Encryption.

  - `--vrdeproperty "Security/CACertificate=<value>"` where <value> is the absolute path of the CA self signed certificate. See RDP Encryption.

- `--vrdeproperty "Audio/RateCorrectionMode|LogPath=<value>"` sets the audio connection mode, or path of the audio logfile.

  - `--vrdeproperty "Audio/RateCorrectionMode=<value>"` where <value> is the desired rate correction mode. Allowed values are:

    - `VRDP_AUDIO_MODE_VOID`: No mode specified, use to unset any Audio mode already set.

    - `VRDP_AUDIO_MODE_RC`: Rate correction mode.

    - `VRDP_AUDIO_MODE_LPF`: Low pass filter mode.

- `VRDP_AUDIO_MODE_CS`: Client sync mode to prevent underflow or overflow of the client queue.

- `--vrdeproperty "Audio/LogPath=<value>"` where <value> is the absolute path of the Audio log file.

- `--vrdeextpack default|<name>`: Specifies the library to use for accessing the VM remotely. The default is to use the RDP code which is part of the Oracle VM VirtualBox Extension Pack.

- `--vrdeport default|<ports>`: A port or a range of ports the VRDE server can bind to. `default` or `0` means port 3389, the standard port for RDP. You can specify a comma-separated list of ports or ranges of ports. Use a dash between two port numbers to specify a range. The VRDE server will bind to *one* of the available ports from the specified list. Only one machine can use a given port at a time. For example, the option `--vrdeport 5000,5010-5012` will tell the server to bind to one of following ports: 5000, 5010, 5011, or 5012.

- `--vrdeaddress <IP address>`: The IP address of the host network interface the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface.

  The setting can be used to specify whether the VRDP server should accept either IPv4, IPv6, or both connections:

  - Only IPv4: `--vrdeaddress "0.0.0.0"`

  - Only IPv6: `--vrdeaddress "::"`

  - Both IPv6 and IPv4: `--vrdeaddress ""`

    This is the default setting.

- `--vrdeauthtype null|external|guest`: Enables you to indicate use of authorization, and specify how authorization will be performed. See RDP Authentication.

- `--vrdeauthlibrary default|<name>`: Specifies the library used for RDP authentication. See RDP Authentication.

- `--vrdemulticon on|off`: Enables multiple connections to be made to the same VRDE server, if the server supports this feature. See Multiple Connections to the VRDP Server.

- `--vrdereusecon on|off`: This specifies the VRDE server behavior when multiple connections are disabled. When this option is enabled, the server will allow a new client to connect and will drop the existing connection. When this option is disabled, the default setting, a new connection will not be accepted if there is already a client connected to the server.

- `--vrdevideochannel on|off`: Enables video redirection, if it is supported by the VRDE server. See VRDP Video Redirection.

- `--vrdevideochannelquality <percent>`: Specifies the image quality for video redirection. See VRDP Video Redirection.

7.8.6. Teleporting Settings

With the following commands for `VBoxManage modifyvm` you can configure a machine to be a target for teleporting. See Teleporting.

- `--teleporter on|off`: Enables and disables the teleporter feature whereby when the machine is started, it waits to receive a teleporting request from the network instead of booting normally. Teleporting requests are received on the port and address specified using the following parameters.

- `--teleporterport <port>`, `--teleporteraddress <address>`: These settings must be used with `--teleporter`. They specify the port and address the virtual machine should listen to in order to receive a teleporting request sent from another virtual machine. `<port>` can be any free TCP/IP port number, such as 6000. `<address>` can be any IP address or hostname and specifies the TCP/IP socket to bind to. The default is 0.0.0.0, which means any address.

- `--teleporterpassword <password>`: If this optional setting is used, then the teleporting request will only succeed if the source machine specifies the same password as the one given with this command.

- `--teleporterpasswordfile <password>`: If this optional setting is used, then the teleporting request will only succeed if the source machine specifies the same password as the one specified in the file give with this command. Use `stdin` to read the password from stdin.

- `--cpuid <leaf> <eax> <ebx> <ecx> <edx>`: Advanced users can use this setting before a teleporting operation, to restrict the virtual CPU capabilities that Oracle VM VirtualBox presents to the guest operating system. This must be run on both the source and the target machines involved in the teleporting and will then modify what the guest sees when it executes the `CPUID` machine instruction. This might help with misbehaving applications that wrongly assume that certain CPU capabilities are present. The meaning of the parameters is hardware dependent, refer to the AMD or Intel processor documentation.

7.8.7. Debugging Settings

The following settings are only relevant for low-level VM debugging. Regular users will never need these settings.

- `--tracing-enabled on|off`: Enables the tracebuffer. This consumes some memory for the tracebuffer and adds extra overhead.

- `--tracing-config <config-string>`: Enables tracing configuration. In particular, this defines which group of tracepoints are enabled.

- `--tracing-allow-vm-access on|off`: Enables and disables VM access to the tracebuffer. By default, this setting is disabled.

7.8.8. USB Card Reader Settings

The following setting defines access to a USB Card Reader by the guest environment. USB card readers are typically used for accessing data on memory cards such as CompactFlash (CF), Secure Digital (SD), or MultiMediaCard (MMC).

- `--usbcardreader on|off`: Enables and disables the USB card reader interface.

7.8.9. Autostarting VMs During Host System Boot

These settings configure the VM autostart feature, which automatically starts the VM at host system boot-up. Note that there are prerequisites that need to be addressed before using this feature. See Starting Virtual Machines During System Boot.

- `--autostart-enabled on|off`: Enables and disables VM autostart at host system boot-up, using the specified user name.

- `--autostart-delay <seconds>`: Specifies a delay, in seconds, following host system boot-up, before the VM autostarts.

7.9. VBoxManage movevm

This command moves a virtual machine to a new location on the host.

Associated files of the virtual machine, such as settings files and disk image files, are moved to the new location. The Oracle VM VirtualBox configuration is updated automatically.

The `movevm` subcommand requires the name of the virtual machine which should be moved.

Also required is the type of move operation, specified by `--type basic`. Other types of move operation may be supported in future releases.

The `--folder` setting configures the new location on the host file system. Enter a relative pathname or a full pathname.

7.10. VBoxManage import

This command imports one or more virtual machines into Oracle VM VirtualBox. You can import from either of the following:

• A virtual appliance in OVF format.

• A cloud service, such as Oracle Cloud Infrastructure. Only a single cloud instance can be imported.

See for more details on importing VMs into Oracle VM VirtualBox.

7.10.1. Import from OVF

The `import` subcommand takes at least the path name of an OVF file as input and expects the disk images, if needed, to be in the same directory as the OVF file. Many additional command-line options are supported. These enable you to control in detail what is being imported and to modify the import parameters, depending on the content of the OVF file.

It is therefore recommended to first run the `import` subcommand with the `--dry-run` or `-n` option. This will then print a description of the appliance's contents to the screen how it would be imported into Oracle VM VirtualBox, together with the optional command-line options to influence the import behavior.

Use of the `--options keepallmacs|keepnatmacs|keepdisknames` option enables additional fine tuning of the import operation. The first two options enable you to specify how the MAC addresses of every virtual network card should be handled. They can either be reinitialized, which is the default setting, left unchanged (`keepallmacs`) or left unchanged when the network type is NAT (`keepnatmacs`). If you add `keepdisknames` all new disk images are assigned the same names as the originals, otherwise they are renamed.

As an example, the following is a screen output for a sample appliance containing a Windows XP guest:

```
VBoxManage import WindowsXp.ovf --dry-run
     Interpreting WindowsXp.ovf...
     OK.
     Virtual system 0:
      0: Suggested OS type: "WindowsXP"
         (change with "--vsys 0 --ostype <type>"; use "list ostypes" to list all)
      1: Suggested VM name "Windows XP Professional_1"
         (change with "--vsys 0 --vmname <name>")
      2: Suggested VM group "/"
         (change with "--vsys 0 --group <group>")
      3: Suggested VM settings file name "/home/klaus/VirtualBox VMs/dummy2 2/dummy2 2.vbox"
         (change with "--vsys 0 --settingsfile <filename>")
```

```
    4: Suggested VM base folder "/home/klaus/VirtualBox VMs"
       (change with "--vsys 0 --basefolder <path>")
    5: End-user license agreement
       (display with "--vsys 0 --eula show";
       accept with "--vsys 0 --eula accept")
    6: Number of CPUs: 1
       (change with "--vsys 0 --cpus <n>")
    7: Guest memory: 956 MB (change with "--vsys 0 --memory <MB>")
    8: Sound card (appliance expects "ensoniq1371", can change on import)
       (disable with "--vsys 0 --unit 5 --ignore")
    9: USB controller
       (disable with "--vsys 0 --unit 6 --ignore")
   10: Network adapter: orig bridged, config 2, extra type=bridged
   11: Floppy
       (disable with "--vsys 0 --unit 8 --ignore")
   12: SCSI controller, type BusLogic
       (change with "--vsys 0 --unit 9 --scsitype {BusLogic|LsiLogic}";
       disable with "--vsys 0 --unit 9 --ignore")
   13: IDE controller, type PIIX4
       (disable with "--vsys 0 --unit 10 --ignore")
   14: Hard disk image: source image=WindowsXp.vmdk,
         target path=/home/user/disks/WindowsXp.vmdk, controller=9;channel=0
       (change controller with "--vsys 0 --unit 11 --controller <id>";
       disable with "--vsys 0 --unit 11 --ignore")
```

The individual configuration items are numbered, and depending on their type support different command-line options. The import subcommand can be directed to ignore many such items with a `--vsys X --unit Y --ignore` option, where X is the number of the virtual system and Y the item number, as printed on the screen. X is zero, unless there are several virtual system descriptions in the appliance.

In the above example, Item #1 specifies the name of the target machine in Oracle VM VirtualBox. Items #12 and #13 specify hard disk controllers, respectively. Item #14 describes a hard disk image. In this case, the additional `--controller` option indicates which item the disk image should be connected to, with the default coming from the OVF file.

You can combine several items for the same virtual system using the `--vsys` option. For example, to import a machine as described in the OVF, but without the sound card and without the USB controller, and with the disk image connected to the IDE controller instead of the SCSI controller, use the following command:

```
VBoxManage import WindowsXp.ovf
  --vsys 0 --unit 8 --ignore --unit 9 --ignore --unit 14 --controller 13
```

7.10.2. Import from Oracle Cloud Infrastructure

As the result of this operation, a file with the suffix `.oci` is downloaded to the local host. This file is a TAR archive which contains a bootable instance image in QCOW2 format and a JSON file with some metadata related to the imported instance.

The downloaded file is deleted after a successful import. If import fails, the downloaded file may not be deleted and the VBoxSVC log file may indicate the location where the file was stored.

During import the bootable image is extracted from the archive and converted into VMDK format. The JSON file is also extracted and stored in the VM machine folder.

The command syntax for importing an Oracle Cloud Infrastructure instance begins with `VBoxManage import OCI:// --cloud`.

You can list the available Oracle Cloud Infrastructure VM instances and their IDs by using the following command:

```
VBoxManage cloud --provider=OCI --profile=cloud-profile-name list instances
```

To import a VM from a cloud service such as Oracle Cloud Infrastructure, use the `--cloud` option to specify the import from the Cloud. Some of the following options are settings for the VM, for others you must enter an Oracle Cloud Identifier (OCID) for a resource. Use the Oracle Cloud Infrastructure Console to view OCIDs.

The following parameters can be specified:

- `--vmname`: Specifies a new name for the imported VM. This name is used as the VM name by Oracle VM VirtualBox.

- `--cloudinstanceid`: The ID of an existing instance in the Cloud.

- `--cloudprofile`: Specifies the cloud profile that is used to connect to the cloud service provider. The cloud profile contains your Oracle Cloud Infrastructure account details, such as your user OCID and the fingerprint for your public key. To use a cloud profile, you must have the required permissions on Oracle Cloud Infrastructure.

- `--cloudbucket`: Specifies the bucket name in which to store the object created from an instance bootable volume. In Oracle Cloud Infrastructure, a bucket is a logical container for storing objects.

The following import options have the same meaning as for OVF import:

- `--ostype`: An OS type supported by Oracle VM VirtualBox. Use the `VBoxManage list ostypes` command to see the whole list of supported OSes. If the type was not set, the `Unknown` type is used.

- `--basefolder`: The folder where the new VM is stored.

- `--description`: A string describing the VM.

- `--memory`: The amount of RAM memory assigned for the VM, in MB. If this option is not set either the default memory size for the OS type is used, or the value is taken from the Oracle Cloud Infrastructure instance.

- `--cpus`: the number of virtual CPUs assigned for the VM. If this option is not set, either the default virtual CPUs setting for the OS type is used, or the value is taken from the Oracle Cloud Infrastructure instance.

The import options `--disk`, `--controller`, `--scsitype`, `--unit`, `--settingsfile` are not valid for cloud import.

The following example shows a typical command line for importing an instance from Oracle Cloud Infrastructure:

```
# VBoxManage import OCI:// --cloud --vmname import_from_oci --memory 4000
  --cpus 3 --ostype FreeBSD_64 --cloudprofile "standard user"
  --cloudinstanceid ocid1.instance.oc1.iad.abuwc... --cloudbucket myBucket
```

7.11. VBoxManage export

This command exports one or more virtual machines from Oracle VM VirtualBox. You can export to either of the following:

- A virtual appliance in OVF format, including copying their virtual disk images to compressed VMDK.

- A cloud service, such as Oracle Cloud Infrastructure. A single VM can be exported in VMDK format.

See Section 1.14, "Importing and Exporting Virtual Machines" for more details on exporting VMs from Oracle VM VirtualBox.

7.11.1. Export to OVF

List the machine, or the machines, that you would like to export to the same OVF file and specify the target OVF file after an additional `--output` or `-o` option. Note that the directory of the target OVF file will also receive the exported disk images in the compressed VMDK format, regardless of the original format, and should have enough disk space left for them.

Beside a simple export of a given virtual machine, you can append several product information to the appliance file. Use `--product`, `--producturl`, `--vendor`, `--vendorurl`, `--version` and `--description` to specify this additional information. For legal reasons you may add a license text or the content of a license file by using the `--eula` and `--eulafile` option respectively.

As with OVF import, you use the `--vsys X` option to apply these options to the correct virtual machine.

For virtualization products which are not fully compatible with the OVF standard 1.0 you can enable an OVF 0.9 legacy mode with the `--legacy09` option. Other options are `--ovf09`, `--ovf10`, `--ovf20`.

To specify options controlling the exact content of the appliance file, you can use `--options` to request the creation of a manifest file, which enables detection of corrupted appliances on import, the additional export of DVD images, and the exclusion of MAC addresses. You can specify a list of options, such as `--options manifest,nomacs`. For details, check the help output of `VBoxManage export`.

7.11.2. Export to Oracle Cloud Infrastructure

By default, an exported disk image is converted into stream VMDK format. This ensures compatibility with Oracle Cloud Infrastructure.

List the machine that you want to export to Oracle Cloud Infrastructure and specify the target cloud service provider by using the `--output` or `-o` option.

To export a VM to a cloud service such as Oracle Cloud Infrastructure, use the `--cloud` option to specify the VM to export. This option works in the same way as the `--vsys` option for OVF export.

Some of the following options are settings for the VM instance. As a result, you must enter an Oracle Cloud Identifier (OCID) for a resource. Use the Oracle Cloud Infrastructure Console to view OCIDs.

- `--output/-o`: Specifies the short name of the cloud service provider to which you export. For Oracle Cloud Infrastructure, enter `OCI://`.

- `--cloud` *number-of-virtual-system*: Specifies a number that identifies the VM that you are exporting. Numbering starts at `0` for the first VM.

- `--vmname` *name*: Specifies the name of the exported VM. This name is used as the VM instance name in Oracle Cloud Infrastructure.

- `--cloudprofile` *cloud-profile-name*: Specifies the cloud profile that is used to connect to the cloud service provider. The cloud profile contains your Oracle Cloud Infrastructure account details, such as your user OCID and the fingerprint for your public key. See Section 1.14.4, "Exporting an Appliance to Oracle Cloud Infrastructure".

  To use a cloud profile, you must have the required permissions on Oracle Cloud Infrastructure.

- `--cloudshape` *shape*: Specifies the shape used for the VM instance. The shape defines the number of CPUs and the amount of memory allocated to the VM instance. The shape must be compatible with the exported image.

- `--clouddomain` *domain*: Specifies the availability domain to use for the VM instance. Enter the full name of the availability domain.

- `--clouddisksize` *`disk-size-in-GB`*: Specifies the disk size used for the exported disk image in gigabytes. The minimum value is 50 GB and the maximum value is 300 GB.

- `--cloudbucket` *`bucket-name`*: Specifies the bucket in which to store the uploaded files. In Oracle Cloud Infrastructure, a bucket is a logical container for storing objects.

- `--cloudocivcn` *`OCI-vcn-ID`*: Specifies the virtual cloud network (VCN) to use for the VM instance. Enter the OCID for the VCN.

- `--cloudocisubnet` *`OCI-subnet-ID`*: Specifies the subnet of the VCN to use for the VM instance. Enter the OCID for the subnet.

- `--cloudkeepobject true | false`: Specifies whether to store the exported disk image in Oracle Object Storage.

- `--cloudlaunchinstance true | false`: Specifies whether to start the VM instance after the export to Oracle Cloud Infrastructure completes.

- `--cloudpublicip true | false`: Specifies whether to enable a public IP address for the VM instance.

The following example shows a typical command line for exporting a VM to Oracle Cloud Infrastructure.

```
# VBoxManage export myVM --output OCI:// --cloud 0 --vmname myVM_Cloud \
--cloudprofile "standard user" --cloudbucket myBucket \
--cloudshape VM.Standard2.1 --clouddomain US-ASHBURN-AD-1 --clouddisksize 50  \
--cloudocivcn ocid1.vcn.oc1.iad.aaaa... --cloudocisubnet ocid1.subnet.oc1.iad.aaaa... \
--cloudkeepobject true --cloudlaunchinstance true --cloudpublicip true
```

7.12. VBoxManage startvm

This command starts a virtual machine that is currently in the Powered Off or Saved states.

The optional `--type` specifier determines whether the machine will be started in a window or whether the output should go through `VBoxHeadless`, with VRDE enabled or not. See VBoxHeadless, the Remote Desktop Server. The list of types is subject to change, and it is not guaranteed that all types are accepted by any product variant.

The global or per-VM default value for the VM frontend type will be taken if the type is not explicitly specified. If none of these are set, the GUI variant will be started.

The following values are allowed:

`gui`	Starts a VM showing a GUI window. This is the default.
`headless`	Starts a VM without a window for remote display only.
`separate`	Starts a VM with a detachable UI. Technically, it is a headless VM with user interface in a separate process. This is an experimental feature as it lacks certain functionality, such as 3D acceleration.

Note

If you experience problems with starting virtual machines with particular frontends and there is no conclusive error information, consider starting virtual machines directly by running the respective front-end, as this can give additional error information.

7.13. VBoxManage controlvm

The `controlvm` subcommand enables you to change the state of a virtual machine that is currently running. The following can be specified:

- `VBoxManage controlvm <vm> pause`: Temporarily puts a virtual machine on hold, without permanently changing its state. The VM window is gray, to indicate that the VM is currently paused. This is equivalent to selecting the **Pause** item in the **Machine** menu of the GUI.

- Use `VBoxManage controlvm <vm> resume`: Undoes a previous `pause` command. This is equivalent to selecting the **Resume** item in the **Machine** menu of the GUI.

- `VBoxManage controlvm <vm> reset`: Has the same effect on a virtual machine as pressing the Reset button on a real computer. A cold reboot of the virtual machine is done, which immediately restarts and reboots the guest operating system. The state of the VM is not saved beforehand, and data may be lost. This is equivalent to selecting the **Reset** item in the **Machine** menu of the GUI.

- `VBoxManage controlvm <vm> poweroff`: Has the same effect on a virtual machine as pulling the power cable on a real computer. The state of the VM is not saved beforehand, and data may be lost. This is equivalent to selecting the **Close** item in the **Machine** menu of the GUI, or clicking the VM window's close button, and then selecting **Power Off the Machine** in the displayed dialog.

  After this, the VM's state will be Powered Off. From that state, it can be started again. See Section 7.12, "VBoxManage startvm".

- `VBoxManage controlvm <vm> savestate`: Saves the current state of the VM to disk and then stops the VM. This is equivalent to selecting the **Close** item in the **Machine** menu of the GUI or clicking the VM window's close button, and then selecting **Save the Machine State** in the displayed dialog.

  After this, the VM's state will be Saved. From this state, it can be started again. See Section 7.12, "VBoxManage startvm".

- `VBoxManage controlvm <vm> acpipowerbutton`: Sends an ACPI shutdown signal to the VM, as if the power button on a real computer had been pressed. So long as the VM is running a fairly modern guest operating system providing ACPI support, this should trigger a proper shutdown mechanism from within the VM.

- `VBoxManage controlvm <vm> keyboardputscancode <hex> [<hex>...]`: Sends commands using keycodes to the VM. Keycodes are documented in the public domain. For example: http://www.win.tue.nl/~aeb/linux/kbd/scancodes-1.html.

- `VBoxManage controlvm "VM name" teleport --hostname <name> --port <port> [--passwordfile <file> | --password <password>]`: Makes the machine the source of a teleporting operation and initiates a teleport to the given target. See Teleporting. If the optional password is specified, it must match the password that was given to the `modifyvm` command for the target machine. See Section 7.8.6, "Teleporting Settings".

The following extra options are available with `controlvm` that do not directly affect the VM's running state:

- `setlinkstate<1-N> on|off`: Connects or disconnects virtual network cables from their network interfaces.

- `nic<1-N> null|nat|bridged|intnet|hostonly|generic|natnetwork[<devicename>]`: Specifies the type of networking that should be made available on the specified VM virtual network card. They available types are: not connected to the host (`null`), use network address translation (`nat`), bridged networking (`bridged`), communicate with other virtual machines using internal networking

(`intnet`), host-only networking (`hostonly`), natnetwork networking (`natnetwork`), or access to rarely used submodes (`generic`). These options correspond to the modes which are described in detail in Section 6.2, "Introduction to Networking Modes".

- With the `nictrace` options, you can optionally trace network traffic by dumping it to a file, for debugging purposes.

  `nictrace<1-N> on|off`: Enables network tracing for a particular virtual network card.

  Before enabling you should specify a file name to which the trace should be logged. This can be done with the `nictracefile<1-N> <filename>` option to `VBoxManage controlvm` at runtime or with the `<filename>` option to `VBoxManage modifyvm` otherwise.

- `nicpromisc<1-N> deny|allow-vms|allow-all`: Specifies how the promiscious mode is handled for the specified VM virtual network card. This setting is only relevant for bridged networking. The default setting of `deny` hides any traffic not intended for this VM. `allow-vms` hides all host traffic from this VM but enables the VM to see traffic to and from other VMs. `allow-all` removes this restriction completely.

- `nicproperty<1-N> <paramname>="paramvalue"`: This option, in combination with `nicgenericdrv` enables you to pass parameters to rarely-used network backends.

  Those parameters are backend engine-specific, and are different between UDP Tunnel and the VDE backend drivers. See Section 6.8, "UDP Tunnel Networking".

- `natpf<1-N> [<name>],tcp|udp,[<hostip>],<hostport>,[<guestip>], <guestport>`: Specifies a NAT port-forwarding rule. See Section 6.3.1, "Configuring Port Forwarding with NAT".

- `natpf<1-N> delete <name>`: Deletes a NAT port-forwarding rule. See Section 6.3.1, "Configuring Port Forwarding with NAT".

- The `guestmemoryballoon<balloon size in MB>`: Changes the size of the guest memory balloon. This is the memory allocated by the Oracle VM VirtualBox Guest Additions from the guest operating system and returned to the hypervisor for reuse by other virtual machines. This must be specified in megabytes. See Section 4.10.1, "Memory Ballooning".

- `usbattach<uuid|address> [--capturefile <filename>]`

  and `usbdetach <uuid|address> [--capturefile <filename>]`: Makes host USB devices visible or invisible to the virtual machine on the fly, without the need for creating filters first. The USB devices can be specified by UUID (unique identifier) or by address on the host system. Use the `--capturefile` option to specify the absolute path of a file for writing activity logging data.

  You can use `VBoxManage list usbhost` to locate this information.

- `audioin on`: Selects whether capturing audio from the host is enabled or disabled.

- `audioout on`: Selects whether audio playback from the guest is enabled or disabled.

- `clipboard disabled|hosttoguest|guesttohost|bidirectional`: Selects how the guest or host operating system's clipboard should be shared with the host or guest. See Section 3.4, "General Settings". This requires that the Guest Additions be installed in the virtual machine.

- `draganddrop disabled|hosttoguest|guesttohost|bidirectional`: Selects the current drag and drop mode being used between the host and the virtual machine. See Section 4.4, "Drag and Drop". This requires that the Guest Additions be installed in the virtual machine.

- `vrde on|off`: Enables and disables the VRDE server, if it is installed.

- `vrdeport default|<ports>`: Changes the port or a range of ports that the VRDE server can bind to. `default` or `0` means port 3389, the standard port for RDP. See the description for the `--vrdeport` option in Section 7.8.5, "Remote Machine Settings".

- `vrdeproperty "TCP/Ports|Address=<value>"`: Sets the port numbers and IP address on the VM to which the VRDE server can bind.

  - For TCP/Ports, <value> should be a port or a range of ports to which the VRDE server can bind. `default` or `0` means port 3389, the standard port for RDP. See the description for the `--vrdeport` option in Section 7.8.5, "Remote Machine Settings".

  - For TCP/Address, <value>: The IP address of the host network interface that the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface. See the description for the `--vrdeaddress` option in Section 7.8.5, "Remote Machine Settings".

- `vrdeproperty "VideoChannel/Enabled|Quality|DownscaleProtection=<value>"`: Sets the VRDP video redirection properties.

  - For VideoChannel/Enabled, <value> can be set to "1" switching the VRDP video channel on. See VRDP Video Redirection.

  - For VideoChannel/Quality, <value> should be set between 10 and 100% inclusive, representing a JPEG compression level on the VRDE server video channel. Lower values mean lower quality but higher compression. See VRDP Video Redirection.

  - For VideoChannel/DownscaleProtection, <value> can be set to "1" to enable the videochannel downscale protection feature. When enabled, if a video's size equals the shadow buffer size, then it is regarded as a full screen video, and is displayed. If its size is between fullscreen and the downscale threshold it is not displayed, as it could be an application window, which would be unreadable when downscaled. When the downscale protection feature is disabled, an attempt is always made to display videos.

- `vrdeproperty "Client/DisableDisplay|DisableInput|DisableAudio|DisableUSB=1"`: Disables one of the VRDE server features: Display, Input, Audio, or USB. To reenable a feature, use "Client/DisableDisplay=" for example. See VRDP Customization.

- `vrdeproperty "Client/DisableClipboard|DisableUpstreamAudio=1"`. Disables one of the VRDE server features: Clipboard or UpstreamAudio. To reenable a feature, use "Client/DisableClipboard=" for example. See VRDP Customization.

- `vrdeproperty "Client/DisableRDPDR=1"`: Disables the VRDE server feature: RDP device redirection for smart cards. To reenable this feature, use "Client/DisableRDPR=".

- `vrdeproperty "H3DRedirect/Enabled=1"`: Enables the VRDE server feature: 3D redirection. To disable this feature, use "H3DRedirect/Enabled=".

- `vrdeproperty "Security/Method|ServerCertificate|ServerPrivateKey|CACertificate=<value>"`: Sets the desired security method, path of the server certificate, path of the server private key, and path of CA certificate, used for a connection.

  - `vrdeproperty "Security/Method=<value>"`: Sets the desired security method, which is used for a connection. Valid values are as follows:

    - `Negotiate`: Both Enhanced (TLS) and Standard RDP Security connections are allowed. The security method is negotiated with the client. This is the default setting.

    - `RDP`: Only Standard RDP Security is accepted.

- `TLS`: Only Enhanced RDP Security is accepted. The client must support TLS.

  See RDP Encryption.

- `vrdeproperty "Security/ServerCertificate=<value>"` where <value> is the absolute path of the server certificate. See RDP Encryption.

- `vrdeproperty "Security/ServerPrivateKey=<value>"` where <value> is the absolute path of the server private key. See RDP Encryption.

- `vrdeproperty "Security/CACertificate=<value>"` where <value> is the absolute path of the CA self signed certificate. See RDP Encryption.

- `vrdeproperty "Audio/RateCorrectionMode|LogPath=<value>"`: Sets the audio connection mode, or path of the audio logfile.

  - `vrdeproperty "Audio/RateCorrectionMode=<value>"` where <value> is the desired rate correction mode, allowed values are:

    - `VRDP_AUDIO_MODE_VOID`: No mode specified, use to unset any Audio mode already set.

    - `VRDP_AUDIO_MODE_RC`: Rate correction mode.

    - `VRDP_AUDIO_MODE_LPF`: Low pass filter mode.

    - `VRDP_AUDIO_MODE_CS`: Client sync mode to prevent underflow or overflow of the client queue.

  - `vrdeproperty "Audio/LogPath=<value>"` where <value> is the absolute path of the audio log file.

- `vrdevideochannelquality <percent>`: Sets the image quality for video redirection. See VRDP Video Redirection.

- `setvideomodehint`: Requests that the guest system change to a particular video mode. This requires that the Guest Additions be installed, and will not work for all guest systems.

- `screenshotpng`: Takes a screenshot of the guest display and saves it in PNG format.

- `recording on|off` enables or disables the recording of a VM session into a WebM/VP8 file. When this option value is `on`, recording begins when the VM session starts.

- `recordingscreens all|screen-ID [screen-ID ...]` enables you to specify which VM screens to record. The recording for each screen that you specify is saved to its own file. You cannot modify this setting while recording is enabled.

- `recordingfile filename` specifies the file in which to save the recording. You cannot modify this setting while recording is enabled.

- `recordingvideores widthxheight` specifies the resolution of the recorded video in pixels. You cannot modify this setting while recording is enabled.

- `recordingvideorate bit-rate` specifies the bit rate of the video in kilobits per second. Increasing this value improves the appearance of the video at the cost of an increased file size. You cannot modify this setting while recording is enabled.

- `recordingvideofps fps` specifies the maximum number of video frames per second (FPS) to record. Frames that have a higher frequency are skipped. Increasing this value reduces the number of skipped frames and increases the file size. You cannot modify this setting while recording is enabled.

- `recordingmaxtime` *seconds* specifies the maximum amount time to record in seconds. The recording stops after the specified number of seconds elapses. If this value is zero, the recording continues until you stop the recording.

- `recordingmaxsize` *MB* specifies the maximum size of the recorded video file in megabytes. The recording stops when the file reaches the specified size. If this value is zero, the recording continues until you stop the recording. You cannot modify this setting while recording is enabled.

- `recordingopts` *keyword=value[,keyword=value ...]* specifies additional recording options in a comma-separated keyword-value format. For example, `foo=bar,a=b`. You cannot modify this setting while recording is enabled.

  Only use this option only if you are an advanced user. For information about keywords, see *Oracle VM VirtualBox Programming Guide and Reference*.

- `setcredentials`: Used for remote logins on Windows guests. See Automated Guest Logins.

- `teleport --host <name> --port <port>`: Configures a VM as a target for teleporting. <name> specifies the virtual machine name. <port> specifies the port on the virtual machine which should listen for teleporting requests from other virtual machines. It can be any free TCP/IP port number, such as 6000. See Teleporting.

  - `--maxdowntime <msec>`: Specifies the maximum downtime, in milliseconds, for the teleporting target VM. Optional.

  - `--password <password>`: The teleporting request will only succeed if the source machine specifies the same password as the one given with this command. Optional.

  - `--passwordfile <password file>`: The teleporting request will only succeed if the source machine specifies the same password as the one specified in the password file with the path specified with this command. Use `stdin` to read the password from stdin. Optional.

- `plugcpu|unplugcpu <id>`: If CPU hot-plugging is enabled, this setting adds and removes a virtual CPU to the virtual machine. `<id>` specifies the index of the virtual CPU to be added or removed and must be a number from 0 to the maximum number of CPUs configured. CPU 0 can never be removed.

- The `cpuexecutioncap <1-100>`: Controls how much CPU time a virtual CPU can use. A value of 50 implies a single virtual CPU can use up to 50% of a single host CPU.

- `vm-process-priority default|flat|low|normal|high`: Changes the priority scheme of the VM process. See Section 7.12, "VBoxManage startvm".

- `webcam attach <path|alias> [<keyword=value>[;<keyword=value>...]]`: Attaches a webcam to a running VM. Specify the absolute path of the webcam on the host operating system, or use its alias, obtained by using the command: `VBoxManage list webcams`.

  Note that alias '.0' means the default video input device on the host operating system, '.1', '.2', etc. mean first, second, etc. video input device. The device order is host-specific.

  The optional settings parameter is a `;` delimited list of name-value pairs, enabling configuration of the emulated webcam device.

  The following settings are supported:

  MaxFramerate: Specifies the highest rate in frames per second, at which video frames are sent to the guest. Higher frame rates increase CPU load, so this setting can be useful when there is a need to reduce CPU load. The default setting is `no maximum limit`, thus enabling the guest to use all frame rates supported by the host webcam.

MaxPayloadTransferSize: Specifies the maximum number of bytes the emulated webcam can send to the guest in one buffer. The default setting is 3060 bytes, which is used by some webcams. Higher values can slightly reduce CPU load, if the guest is able to use larger buffers. Note that higher MaxPayloadTransferSize values may be not supported by some guest operating systems.

- `webcam detach <path|alias>`: Detaches a webcam from a running VM. Specify the absolute path of the webcam on the host, or use its alias obtained from the `webcam list` command.

  Please note the following points, relating to specific host operating systems:

  - Windows hosts: When the webcam device is detached from the host, the emulated webcam device is automatically detached from the guest.

  - Mac OS X hosts: OS X version 10.7 or newer is required.

    When the webcam device is detached from the host, the emulated webcam device remains attached to the guest and must be manually detached using the `VBoxManage controlvm webcam detach` command.

  - Linux hosts: When the webcam is detached from the host, the emulated webcam device is automatically detached from the guest only if the webcam is streaming video. If the emulated webcam is inactive, it should be manually detached using the `VBoxManage controlvm webcam detach` command.

- `webcam list`: Lists webcams attached to the running VM. The output is a list of absolute paths or aliases that were used for attaching the webcams to the VM using the `webcam attach` command.

- `addencpassword <id> <password file>|- [--removeonsuspend <yes|no>]`: Supplies an encrypted VM specified by <id> with the encryption password to enable a headless start. Either specify the absolute path of a password file on the host file system: <password file>, or use `-` to instruct `VBoxManage` to prompt the user for the encryption password.

  `--removeonsuspend <yes|no>`: Specifies whether to remove the passsword or keep the password in VM memory when the VM is suspended. If the VM has been suspended and the password has been removed, the user needs to resupply the password before the VM can be resumed. This feature is useful in cases where the user does not want the password to be stored in VM memory, and the VM is suspended by a host suspend event.

  Note

  On Oracle VM VirtualBox versions 5.0 and later, data stored on hard disk images can be transparently encrypted for the guest. Oracle VM VirtualBox uses the AES algorithm in XTS mode and supports 128 or 256 bit data encryption keys (DEK). The DEK is stored encrypted in the medium properties, and is decrypted during VM startup by supplying the encryption password.

  The `VBoxManage encryptmedium` command is used to create a DEK encrypted medium. See Encrypting Disk Images. When starting an encrypted VM from the Oracle VM VirtualBox GUI, the user will be prompted for the encryption password.

  For a headless encrypted VM start, use the following command:

  ```
  VBoxManage startvm "vmname" --type headless
  ```

Then supply the required encryption password as follows:

```
VBoxManage "vmname" controlvm "vmname" addencpassword ...
```

- `removeencpassword <id>`: Removes encryption password authorization for password <id> for all encrypted media attached to the VM.

- `removeallencpasswords`: Removes encryption password authorization for all passwords for all encrypted media attached to the VM.

- `changeuartmode <1-N>`: Changes the connection mode for a given virtual serial port.

7.14. VBoxManage discardstate

This command discards the saved state of a virtual machine which is not currently running. This will cause the VM's operating system to restart next time you start it. This is the equivalent of pulling out the power cable on a physical machine, and should be avoided if possible.

7.15. VBoxManage adoptstate

If you have a Saved state file (`.sav`) that is separate from the VM configuration, you can use this command to *adopt* the file. This will change the VM to saved state and when you start it, Oracle VM VirtualBox will attempt to restore it from the saved state file you indicated. This command should only be used in special setups.

7.16. VBoxManage closemedium

This command removes a hard disk, DVD, or floppy image from a Oracle VM VirtualBox media registry.

```
VBoxManage closemedium     [disk|dvd|floppy] <uuid|filename>
                           [--delete]
```

Optionally, you can request that the image be deleted. You will get appropriate diagnostics that the deletion failed, however the image will become unregistered in any case.

7.17. VBoxManage storageattach

This command attaches, modifies, and removes a storage medium connected to a storage controller that was previously added with the `storagectl` command. The syntax is as follows:

```
VBoxManage storageattach   <uuid|vmname>
                           --storagectl <name>
                           [--port <number>]
                           [--device <number>]
                           [--type dvddrive|hdd|fdd]
                           [--medium none|emptydrive|additions|
                                   <uuid>|<filename>|host:<drive>|iscsi]
                           [--mtype normal|writethrough|immutable|shareable
                                   readonly|multiattach]
                           [--comment <text>]
                           [--setuuid <uuid>]
                           [--setparentuuid <uuid>]
                           [--passthrough on|off]
                           [--tempeject on|off]
                           [--nonrotational on|off]
                           [--discard on|off]
```

```
                              [--hotpluggable on|off]
                              [--bandwidthgroup name|none]
                              [--forceunmount]
                              [--server <name>|<ip>]
                              [--target <target>]
                              [--tport <port>]
                              [--lun <lun>]
                              [--encodedlun <lun>]
                              [--username <username>]
                              [--password <password>]
                              [--passwordfile <file>]
                              [--initiator <initiator>]
                              [--intnet]
```

A number of parameters are commonly required. Some parameters are required only for iSCSI targets.

The common parameters are as follows:

`uuid|vmname`	The VM UUID or VM Name. Mandatory.
`--storagectl`	Name of the storage controller. Mandatory. The list of the storage controllers currently attached to a VM can be obtained with `VBoxManage showvminfo`. See Section 7.5, "VBoxManage showvminfo".
`--port`	The number of the storage controller's port which is to be modified. Mandatory, unless the storage controller has only a single port.
`--device`	The number of the port's device which is to be modified. Mandatory, unless the storage controller has only a single device per port.
`--type`	Define the type of the drive to which the medium is being attached, detached, or modified. This argument can only be omitted if the type of medium can be determined from either the medium given with the `--medium` argument or from a previous medium attachment.
`--medium`	Specifies what is to be attached. The following values are supported:

- `none`: Any existing device should be removed from the given slot.

- `emptydrive`: For a virtual DVD or floppy drive only, this makes the device slot behave like a removeable drive into which no media has been inserted.

- `additions`: For a virtual DVD drive only, this attaches the *VirtualBox Guest Additions* image to the given device slot.

- If a UUID is specified, it must be the UUID of a storage medium that is already known to Oracle VM VirtualBox. For example, because it has been attached to another virtual machine. See Section 7.4, "VBoxManage list" for details of how to list known media. This medium is then attached to the given device slot.

- If a filename is specified, it must be the full path of an existing disk image in ISO, RAW, VDI, VMDK, or other format. The disk image is then attached to the given device slot.

- `host:<drive>`: For a virtual DVD or floppy drive only, this connects the given device slot to the specified DVD or floppy drive on the host computer.

- `iscsi`: For virtual hard disks only, this is used for specifying an iSCSI target. In this case, additional parameters must be given. These are described below.

  Some of the above changes, in particular for removeable media such as floppies and CDs/DVDs, can be effected while a VM is running. Others, such as device changes or changes in hard disk device slots, require the VM to be powered off.

`--mtype`

Defines how this medium behaves with respect to snapshots and write operations. See Section 5.4, "Special Image Write Modes".

`--comment`

An optional description that you want to have stored with this medium. For example, for an iSCSI target, "Big storage server downstairs". This is purely descriptive and not needed for the medium to function correctly.

`--setuuid, --setparentuuid`

Modifies the UUID or parent UUID of a medium before attaching it to a VM. This is an expert option. Inappropriate use can make the medium unusable or lead to broken VM configurations if any other VM is referring to the same media already. The most frequently used variant is `--setuuid ""`, which assigns a new random UUID to an image. This option is useful for resolving duplicate UUID errors if you duplicated an image using a file copy utility.

`--passthrough`

For a virtual DVD drive only, you can enable DVD writing support. This feature is currently experimental, see Section 5.9, "CD/DVD Support".

`--tempeject`

For a virtual DVD drive only, you can configure the behavior for guest-triggered medium eject. If this is set to on, the eject has only a temporary effect. If the VM is powered off and restarted the originally configured medium will be still in the drive.

`--nonrotational`

Enables you to enable the non-rotational flag for virtual hard disks. Some guests, such as Windows 7 or later, treat such disks like SSDs and do not perform disk fragmentation on such media.

`--discard`

Enables the auto-discard feature for a virtual hard disks. This specifies that a VDI image will be shrunk in response to the trim command from the guest OS. The following requirements must be met:

- The disk format must be VDI.

- The size of the cleared area must be at least 1 MB.

- Oracle VM VirtualBox will only trim whole 1 MB blocks. The VDIs themselves are organized into 1 MB blocks, so this will only work if the space being trimmed is at least a 1 MB contiguous block at a 1 MB boundary. On Windows, occasional defragmentation with `defrag.exe /D`, or on Linux running `btrfs filesystem defrag` as a background cron job may be beneficial.

  Note

  The Guest OS must be configured to issue the `trim` command, and typically this means that

> the guest OS is made to see the disk as an SSD. Ext4 supports the -o discard mount flag. Mac OS X probably requires additional settings. Windows should automatically detect and support SSDs, at least in versions 7, 8, and 10. The Linux exFAT driver from Samsung supports the `trim` command.

It is unclear whether Microsoft's implementation of exFAT supports this feature, even though that file system was originally designed for flash.

Alternatively, there are other methods to issue trim. For example, the Linux `fstrim` command, part of the util-linux package. Earlier solutions required a user to zero out unused areas, using zerofree or similar, and to compact the disk. This is only possible when the VM is offline.

`--bandwidthgroup`	Sets the bandwidth group to use for the given device. See Section 5.8, "Limiting Bandwidth for Disk Images".
`--forceunmount`	For a virtual DVD or floppy drive only, this forcibly unmounts the DVD/CD/Floppy or mounts a new DVD/CD/Floppy even if the previous one is locked down by the guest for reading. See Section 5.9, "CD/DVD Support".

When `iscsi` is used with the `--medium` parameter for iSCSI support, additional parameters must or can be used. See also Section 5.10, "iSCSI Servers".

`--server`	The host name or IP address of the iSCSI target. Required.
`--target`	Target name string. This is determined by the iSCSI target and used to identify the storage resource. Required.
`--tport`	TCP/IP port number of the iSCSI service on the target. Optional.
`--lun`	Logical Unit Number of the target resource. Optional. Often, this value is zero.
`--encodedlun`	Hex-encoded Logical Unit Number of the target resource. Optional. Often, this value is zero.
`--username, --password, --passwordfile`	Username and password, called the initiator secret, for target authentication, if required. Optional.

Note

Username and password are stored without encryption, in clear text, in the XML machine configuration file if no settings password is provided. When a settings password is specified for the first time, the password is stored in encrypted form. As an alternative to providing the password on the command line, a reference to a file containing the text can be provided using the `passwordfile` option.

`--initiator`	iSCSI Initiator. Optional.

Microsoft iSCSI Initiator is a system, such as a server that attaches to an IP network and initiates requests and receives responses from an iSCSI target. The SAN components in Microsoft iSCSI Initiator are largely analogous to Fibre Channel SAN components, and they include the following:

- To transport blocks of iSCSI commands over the IP network, an iSCSI driver must be installed on the iSCSI host. An iSCSI driver is included with Microsoft iSCSI Initiator.

- A gigabit Ethernet adapter that transmits 1000 megabits per second (Mbps) is recommended for the connection to an iSCSI target. Like standard 10/100 adapters, most gigabit adapters use a preexisting Category 5 or Category 6E cable. Each port on the adapter is identified by a unique IP address.

- An iSCSI target is any device that receives iSCSI commands. The device can be an end node, such as a storage device, or it can be an intermediate device, such as a network bridge between IP and Fibre Channel devices. Each port on the storage array controller or network bridge is identified by one or more IP addresses

`--intnet` If specified, connect to the iSCSI target using Internal Networking. This needs further configuration, see Access iSCSI Targets Using Internal Networking.

7.18. VBoxManage storagectl

This command attaches, modifies, and removes a storage controller. After this, virtual media can be attached to the controller with the `storageattach` command.

The syntax for this command is as follows:

```
VBoxManage storagectl       <uuid|vmname>
                            --name <name>
                            [--add ide|sata|scsi|floppy|sas|usb|pcie]
                            [--controller LSILogic|LSILogicSAS|BusLogic|
                                          IntelAhci|PIIX3|PIIX4|ICH6|I82078|
                                          USB|NVMe]
                            [--portcount <1-30>]
                            [--hostiocache on|off]
                            [--bootable on|off]
                            [--rename <name>]
                            [--remove]
```

The parameters are as follows:

`uuid|vmname` The VM UUID or VM Name. Mandatory.

`--name` Specifies the name of the storage controller. Mandatory.

`--add` Specifies the type of the system bus to which the storage controller must be connected.

`--controller` Enables a choice of chipset type being emulated for the given storage controller.

`--portcount` This specifies the number of ports the storage controller should support.

`--hostiocache`	Configures the use of the host I/O cache for all disk images attached to this storage controller. See Section 5.7, "Host Input/Output Caching".
`--bootable`	Specifies whether this controller is bootable.
`--rename`	Specifies a new name for the storage controller.
`--remove`	Removes the storage controller from the VM configuration.

7.19. VBoxManage bandwidthctl

This command creates, deletes, modifies, and shows bandwidth groups of the given virtual machine.

```
VBoxManage bandwidthctl    <uuid|vmname>
                           add <name> --type disk|network --limit <MBps>[k|m|g|K|M|G] |
                           set <name> --limit <MBps>[k|m|g|K|M|G] |
                           remove <name> |
                           list [--machinereadable]
```

The following subcommands are available:

- `add`: Creates a new bandwidth group of a given type.

- `set`: Modifies the limit for an existing bandwidth group.

- `remove`: Deletes a bandwidth group.

- `list`: Shows all bandwidth groups defined for the given VM. Use the `--machinereadable` option to produce the same output, but in machine readable format. This is of the form: name="value" on a line by line basis.

The parameters are as follows:

`uuid|vmname`	The VM UUID or VM Name. Mandatory.
`--name`	Name of the bandwidth group. Mandatory.
`--type`	Type of the bandwidth group. Mandatory. Two types are supported: `disk` and `network`. See Section 5.8, "Limiting Bandwidth for Disk Images" or Section 6.10, "Limiting Bandwidth for Network Input/Output" for the description of a particular type.
`--limit`	Specifies the limit for the given bandwidth group. This can be changed while the VM is running. The default unit is megabytes per second. The unit can be changed by specifying one of the following suffixes: `k` for kilobits per second, `m` for megabits per second, `g` for gigabits per second, `K` for kilobytes per second, `M` for megabytes per second, `G` for gigabytes per second.

> **Note**
>
> The network bandwidth limits apply only to the traffic being sent by virtual machines. The traffic being received by VMs is unlimited.

> **Note**
>
> To remove a bandwidth group it must not be referenced by any disks or adapters in the running VM.

7.20. VBoxManage showmediuminfo

This command shows information about a medium, notably its size, its size on disk, its type, and the virtual machines which use it.

Note

For compatibility with earlier versions of Oracle VM VirtualBox, the `showvdiinfo` command is also supported and mapped internally to the `showmediuminfo` command.

```
VBoxManage showmediuminfo      [disk|dvd|floppy] <uuid|filename>
```

The medium must be specified either by its UUID, if the medium is registered, or by its filename. Registered images can be listed using `VBoxManage list hdds`, `VBoxManage list dvds`, or `VBoxManage list floppies`, as appropriate. See Section 7.4, "VBoxManage list".

7.21. VBoxManage createmedium

This command creates a new medium. The syntax is as follows:

```
VBoxManage createmedium       [disk|dvd|floppy]    --filename <filename>
                              [--size <megabytes>|--sizebyte <bytes>]
                              [--diffparent <uuid>|<filename>
                              [--format VDI|VMDK|VHD] (default: VDI)
                              [--variant Standard,Fixed,Split2G,Stream,ESX]
```

The parameters are as follows:

`--filename <filename>` Specifies a file name <filename> as an absolute path on the host file system. Mandatory.

`--size <megabytes>` Specifies the image capacity, in 1 MB units. Optional.

`--diffparent <uuid>|`
`<filename>` Specifies the differencing image parent, either as a UUID or by the absolute pathname of the file on the host file system. Useful for sharing a base box disk image among several VMs.

`--format VDI|VMDK|VHD` Specifies the file format for the output file. Available options are VDI, VMDK, VHD. The default format is VDI. Optional.

`--variant` Specifies any required file format variants for the output file. This is a comma-separated list of variant flags. Options are Standard,Fixed,Split2G,Stream,ESX. Not all combinations are supported, and specifying mutually incompatible flags results in an error message. Optional.

Note

For compatibility with earlier versions of Oracle VM VirtualBox, the `createvdi` and `createhd` commands are also supported and mapped internally to the `createmedium` command.

7.22. VBoxManage modifymedium

With the `modifymedium` command, you can change the characteristics of a disk image after it has been created.

```
VBoxManage modifymedium  [disk|dvd|floppy]    <uuid|filename>
                         [--type normal|writethrough|immutable|shareable|
                                readonly|multiattach]
                         [--autoreset on|off]
                         [--property <name=[value]>]
                         [--compact]
                         [--resize <megabytes>|--resizebyte <bytes>]
                         [--move <path>]
                         [--setlocation <path>]
```

Note

For compatibility with earlier versions of Oracle VM VirtualBox, the `modifyvdi` and `modifyhd` commands are also supported and mapped internally to the `modifymedium` command.

The disk image to modify must be specified either by its UUID, if the medium is registered, or by its filename. Registered images can be listed using `VBoxManage list hdds`, see Section 7.4, "VBoxManage list". A filename must be specified as a valid path, either as an absolute path or as a relative path starting from the current directory.

The following options are available:

- With the `--type` argument, you can change the type of an existing image between the normal, immutable, write-through and other modes. See Section 5.4, "Special Image Write Modes".

- For immutable hard disks only, the `--autoreset on|off` option determines whether the disk is automatically reset on every VM startup. See Section 5.4, "Special Image Write Modes". By default, autoreset is on.

- The `--compact` option can be used to compact disk images. Compacting removes blocks that only contains zeroes. Using this option will shrink a dynamically allocated image. It will reduce the *physical* size of the image without affecting the logical size of the virtual disk. Compaction works both for base images and for differencing images created as part of a snapshot.

  For this operation to be effective, it is required that free space in the guest system first be zeroed out using a suitable software tool. For Windows guests, you can use the `sdelete` tool provided by Microsoft. Run `sdelete -z` in the guest to zero the free disk space, before compressing the virtual disk image. For Linux, use the `zerofree` utility which supports ext2/ext3 filesystems. For Mac OS X guests, use the `diskutil secureErase freespace 0 /` command from an elevated Terminal.

  Please note that compacting is currently only available for VDI images. A similar effect can be achieved by zeroing out free blocks and then cloning the disk to any other dynamically allocated format. You can use this workaround until compacting is also supported for disk formats other than VDI.

- The `--resize x` option, where x is the desired new total space in megabytes enables you to change the capacity of an existing image. This adjusts the *logical* size of a virtual disk without affecting the physical size much.

  This option currently works only for VDI and VHD formats, and only for the dynamically allocated variants. It can only be used to expand, but not shrink, the capacity. For example, if you originally created a 10 GB disk which is now full, you can use the `--resize 15360` command to change the capacity to 15 GB (15,360 MB) without having to create a new image and copy all data from within a virtual machine. Note however that this only changes the drive capacity. You will typically next need to use a partition management tool inside the guest to adjust the main partition to fill the drive.

  The `--resizebyte x` option does almost the same thing, except that x is expressed in bytes instead of megabytes.

- The `--move <path>` option can be used to relocate a medium to a different location <path> on the host file system. The path can be either relative to the current directory or absolute.

- The `--setlocation <path>` option can be used to set the new location <path> of the medium on the host file system if the medium has been moved for any reasons. The path can be either relative to the current directory or absolute.

> **Note**
>
> The new location is used as is, without any sanity checks. The user is responsible for setting the correct path.

7.23. VBoxManage clonemedium

This command duplicates a virtual disk, DVD, or floppy medium to a new medium, usually an image file, with a new unique identifier (UUID). The new image can be transferred to another host system or reimported into Oracle VM VirtualBox using the Virtual Media Manager. See Section 5.3, "The Virtual Media Manager" and Section 5.6, "Cloning Disk Images". The syntax is as follows:

```
VBoxManage clonemedium       [disk|dvd|floppy] <uuid|inputfile> <uuid|outputfile>

                             [--format VDI|VMDK|VHD|RAW|<other>]
                             [--variant Standard,Fixed,Split2G,Stream,ESX]
                             [--existing]
```

The medium to clone as well as the target image must be described either by its UUIDs, if the mediums are registered, or by its filename. Registered images can be listed by `VBoxManage list hdds`. See Section 7.4, "VBoxManage list". A filename must be specified as valid path, either as an absolute path or as a relative path starting from the current directory.

The following options are available:

`--format`	Set a file format for the output file different from the file format of the input file.
`--variant`	Set a file format variant for the output file. This is a comma-separated list of variant flags. Not all combinations are supported, and specifying inconsistent flags will result in an error message.
`--existing`	Perform the clone operation to an already existing destination medium. Only the portion of the source medium which fits into the destination medium is copied. This means if the destination medium is smaller than the source only a part of it is copied, and if the destination medium is larger than the source the remaining part of the destination medium is unchanged.

> **Note**
>
> For compatibility with earlier versions of Oracle VM VirtualBox, the `clonevdi` and `clonehd` commands are still supported and mapped internally to the `clonemedium` command.

7.24. VBoxManage mediumproperty

This command sets, gets, or deletes a medium property. The syntax is as follows:

```
VBoxManage mediumproperty [disk|dvd|floppy] set <uuid|filename>
                                               <property> <value>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd, or floppy.

- Use `<uuid|filename>` to supply either the UUID or absolute path of the medium or image.

- Use `<property>` to supply the name of the property.

- Use `<value>` to supply the property value.

```
VBoxManage mediumproperty [disk|dvd|floppy] get <uuid|filename>
                                                <property>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd, or floppy.

- Use `<uuid|filename>` to supply either the UUID or absolute path of the medium or image.

- Use `<property>` to supply the name of the property.

```
VBoxManage mediumproperty [disk|dvd|floppy] delete <uuid|filename>
                                                   <property>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd, or floppy.

- Use `<uuid|filename>` to supply either the UUID or absolute path of the medium or image.

- Use `<property>` to supply the name of the property.

7.25. VBoxManage encryptmedium

This command is used to create a DEK encrypted medium or image. See Encrypting Disk Images.

The syntax is as follows:

```
VBoxManage encryptmedium <uuid|filename>
                         [--newpassword <file|->]
                         [--oldpassword <file|->]
                         [--cipher <cipher id>]
                         [--newpasswordid <password id>]
```

- Use `<uuid|filename>` to supply the UUID or absolute path of the medium or image to be encrypted.

- Use `--newpassword <file|->` to supply a new encryption password. Either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the password on the command line. Always use the `--newpasswordid` option with this option.

- Use `--oldpassword <file|->` to supply any old encryption password. Either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the old password on the command line.

  Use this option to gain access to an encrypted medium or image to either change its password using `--newpassword` or change its encryption using `--cipher`.

- Use `--cipher <cipher>` to specify the cipher to use for encryption. This can be either `AES-XTS128-PLAIN64` or `AES-XTS256-PLAIN64`.

  Use this option to change any existing encryption on the medium or image, or to set up new encryption on it for the first time.

- Use `--newpasswordid <password id>` to supply the new password identifier. This can be chosen by the user, and is used for correct identification when supplying multiple passwords during VM startup.

  If the user uses the same password when encrypting multiple images and also the same password identifier, the user needs to supply the password only once during VM startup.

7.26. VBoxManage checkmediumpwd

This command is used to check the current encryption password on a DEK encrypted medium or image. See Encrypting Disk Images.

The syntax is as follows:

```
VBoxManage checkmediumpwd <uuid|filename>
                                    <pwd file|->
```

- Use `<uuid|filename>` to supply the UUID or absolute path of the medium or image to be checked.

- Use `<pwd file|->` to supply the password identifier to be checked. Either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the password on the command line.

7.27. VBoxManage convertfromraw

This command converts a raw disk image to an Oracle VM VirtualBox Disk Image (VDI) file. The syntax is as follows:

```
VBoxManage convertfromraw    <filename> <outputfile>
                             [--format VDI|VMDK|VHD]
                             [--variant Standard,Fixed,Split2G,Stream,ESX]
                             [--uuid <uuid>]
VBoxManage convertfromraw    stdin <outputfile> <bytes>
                             [--format VDI|VMDK|VHD]
                             [--variant Standard,Fixed,Split2G,Stream,ESX]
                             [--uuid <uuid>]
```

The parameters are as follows:

`--bytes`   The size of the image file, in bytes, provided through stdin.

`--format`   Select the disk image format to create. The default format is VDI. Other options are VMDK and VHD.

`--variant`   Choose a file format variant for the output file. This is a comma-separated list of variant flags. Not all combinations are supported, and specifying inconsistent flags will result in an error message.

`--uuid`   Specify the UUID of the output file.

The `stdin` form of the command forces `VBoxManage` to read the content of the disk image from standard input. This useful when using the command in a pipe.

Note	
	For compatibility with earlier versions of Oracle VM VirtualBox, the `convertdd` command is also supported and mapped internally to the `convertfromraw` command.

7.28. VBoxManage getextradata/setextradata

These commands enable you to attach and retrieve string data for a virtual machine or for an Oracle VM VirtualBox configuration, by specifying `global` instead of a virtual machine name. You must specify a keyword as a text string to associate the data with, which you can later use to retrieve it. For example:

```
VBoxManage setextradata Fedora5 installdate 2006.01.01
VBoxManage setextradata SUSE10 installdate 2006.02.02
```

This example would associate the string "2006.01.01" with the keyword installdate for the virtual machine Fedora5, and "2006.02.02" on the machine SUSE10. You could then retrieve the information as follows:

```
VBoxManage getextradata Fedora5 installdate
```

This would return the following:

```
VirtualBox Command Line Management Interface Version version-number
(C) 2005-2018 Oracle Corporation
All rights reserved.

Value: 2006.01.01
```

You could retrieve the information for all keywords as follows:

```
VBoxManage getextradata Fedora5 enumerate
```

To remove a keyword, the `setextradata` command must be run without specifying data, only the keyword. For example:

```
VBoxManage setextradata Fedora5 installdate
```

7.29. VBoxManage setproperty

This command is used to change global settings which affect the entire Oracle VM VirtualBox installation. Some of these correspond to the settings in the **Global Settings** dialog in the graphical user interface. The following properties are available:

`machinefolder`	Specifies the default folder in which virtual machine definitions are kept. See Where Oracle VM VirtualBox Stores its Files.
`hwvirtexclusive`	Specifies whether Oracle VM VirtualBox will make exclusive use of the hardware virtualization extensions (Intel VT-x or AMD-V) of the host system's processor. See Hardware vs. Software Virtualization. If you wish to share these extensions with other hypervisors running at the same time, you must disable this setting. Doing so has negative performance implications.
`vrdeauthlibrary`	Specifies which library to use when external authentication has been selected for a particular virtual machine. See RDP Authentication.
`websrvauthlibrary`	Specifies which library the web service uses to authenticate users. For details about the Oracle VM VirtualBox web service, see the Oracle VM VirtualBox SDK reference, Oracle VM VirtualBox Programming Interfaces.
`vrdeextpack`	Specifies which library implements the VirtualBox Remote Desktop Extension.
`loghistorycount`	Selects how many rotated VM logs are retained.
`autostartdbpath`	Selects the path to the autostart database. See Starting Virtual Machines During System Boot.
`defaultfrontend`	Selects the global default VM frontend setting. See Section 7.12, "VBoxManage startvm".
`logginglevel`	Configures the VBoxSVC release logging details. See http://www.virtualbox.org/wiki/VBoxLogging.
`proxymode`	Configures the mode for an HTTP proxy server.

proxyurl   Configures the URL for an HTTP proxy server. Used when a manual
                                  proxy is configured using the manual setting of the proxymode
                                  property.

7.30. VBoxManage usbfilter add/modify/remove

```
VBoxManage usbfilter        add <index,0-N>
                             --target <uuid|vmname>global
                             --name <string>
                             --action ignore|hold (global filters only)
                            [--active yes|no (yes)]
                            [--vendorid <XXXX> (null)]
                            [--productid <XXXX> (null)]
                            [--revision <IIFF> (null)]
                            [--manufacturer <string> (null)]
                            [--product <string> (null)]
                            [--remote yes|no (null, VM filters only)]
                            [--serialnumber <string> (null)]
                            [--maskedinterfaces <XXXXXXXX>]
```

```
VBoxManage usbfilter        modify <index,0-N>
                             --target <uuid|vmname>global
                            [--name <string>]
                            [--action ignore|hold (global filters only)]
                            [--active yes|no]
                            [--vendorid <XXXX>]
                            [--productid <XXXX>]
                            [--revision <IIFF>]
                            [--manufacturer <string>]
                            [--product <string>]
                            [--remote yes|no (null, VM filters only)]
                            [--serialnumber <string>]
                            [--maskedinterfaces <XXXXXXXX>]
```

```
VBoxManage usbfilter        remove <index,0-N>
                             --target <uuid|vmname>global
```

The usbfilter commands are used for working with USB filters in virtual machines, or global filters
which affect the whole Oracle VM VirtualBox setup. Global filters are applied before machine-specific
filters, and may be used to prevent devices from being captured by any virtual machine. Global filters are
always applied in a particular order, and only the first filter which fits a device is applied. For example, if the
first global filter says to hold, or make available, a particular Kingston memory stick device and the second
filter says to ignore all Kingston devices. That particular Kingston memory stick will be available to any
machine with the appropriate filter, but no other Kingston device will.

When creating a USB filter using usbfilter add, you must supply three or four mandatory parameters.
The index specifies the position in the list at which the filter should be placed. If there is already a filter
at that position, then it and the following ones will be shifted back one place. Otherwise, the new filter
will be added onto the end of the list. The target parameter selects the virtual machine that the filter
should be attached to or use global to apply it to all virtual machines. name is a name for the new filter.
For global filters, action says whether to allow VMs access to devices that fit the filter description (hold)
or not to give them access (ignore). In addition, you should specify parameters to filter by. You can find
the parameters for devices attached to your system using VBoxManage list usbhost. Finally, you
can specify whether the filter should be active. For local filters, whether they are for local devices, remote
devices over an RDP connection, or either.

When you modify a USB filter using usbfilter modify, you must specify the filter by index and by
target, which is either a virtual machine or global. See the output of VBoxManage list usbfilters

to find global filter indexes and `VBoxManage showvminfo` to find indexes for individual machines. The properties which can be changed are the same as for `usbfilter add`. To remove a filter, use `usbfilter remove` and specify the index and the target.

The following is a list of the additional `usbfilter add` and `usbfilter modify` options, with details of how to use them.

- `--action ignore|hold`: Specifies whether devices that fit the filter description are allowed access by machines (hold), or have access denied (ignore). Applies to global filters only.

- `--active yes|no`: Specifies whether the USB Filter is active or temporarily disabled. For `usbfilter create` the default is active.

- `--vendorid <XXXX>|""`: Specifies a vendor ID filter. The string representation for an exact match has the form XXXX, where X is the hexadecimal digit, including leading zeroes.

- `--productid <XXXX>|""`: Specifies a product ID filter. The string representation for an exact match has the form XXXX, where X is the hexadecimal digit, including leading zeroes.

- `--revision <IIFF>|""`: Specifies a revision ID filter. The string representation for an exact match has the form IIFF, where I is the decimal digit of the integer part of the revision, and F is the decimal digit of its fractional part, including leading and trailing zeros. Note that for interval filters, it is best to use the hexadecimal form, because the revision is stored as a 16-bit packed BCD value. Therefore, the expression int:0x0100-0x0199 will match any revision from 1.0 to 1.99 inclusive.

- `--manufacturer <string>|""`: Specifies a manufacturer ID filter, as a string.

- `--product <string>|""`: Specifies a product ID filter, as a string.

- `--remote yes|no""`: Specifies a remote filter, indicating whether the device is physically connected to a remote VRDE client or to a local host machine. Applies to VM filters only.

- `--serialnumber <string>|""`: Specifies a serial number filter, as a string.

- `--maskedinterfaces <XXXXXXXX>`: Specifies a masked interface filter, for hiding one or more USB interfaces from the guest. The value is a bit mask where the set bits correspond to the USB interfaces that should be hidden, or masked off. This feature only works on Linux hosts.

7.31. VBoxManage sharedfolder add/remove

```
VBoxManage sharedfolder     add <uuid|vmname>
                                --name <name> --hostpath <hostpath>
                                [--transient] [--readonly] [--automount]
```

This command enables you to share folders on the host computer with guest operating systems. For this, the guest systems must have a version of the Oracle VM VirtualBox Guest Additions installed which supports this functionality.

Parameters are as follows:

- `<uuid|vmname>`: Specifies the UUID or name of the VM whose guest operating system will be sharing folders with the host computer. Mandatory.

- `--name <name>`: Specifies the name of the share. Each share has a unique name within the namespace of the host operating system. Mandatory.

- `-hostpath <hostpath>`: Specifies the absolute path on the host operating system of the directory to be shared with the guest operating system. Mandatory.

- `-transient`: Specifies that the share is transient, meaning that it can be added and removed at runtime and does not persist after the VM has stopped. Optional.

- `-readonly`: Specifies that the share has only read-only access to files at the host path.

  By default, shared folders have read/write access to the files on the host path. On Linux distributions, shared folders are mounted with 770 file permissions with root user and vboxsf as the group. Using this option the file permissions change to 700. Optional.

- `-automount`: Specifies that the share will be automatically mounted. On Linux distributions, this will be to either `/media/USER/sf_share-name` or `/media/sf_share-name`, where `share-name` is the share named. The actual location depends on the guest OS. Optional.

```
VBoxManage sharedfolder      remove <uuid|vmname>
                             --name <name> [--transient]
```

This command enables you to delete shared folders on the host computer shares with the guest operating systems. For this, the guest systems must have a version of the Oracle VM VirtualBox Guest Additions installed which supports this functionality.

Parameters are as follows:

- `<uuid|vmname>`: Specifies the UUID or name of the VM whose guest operating system is sharing folders with the host computer. Mandatory.

- `--name <name>`: Specifies the name of the share to be removed. Each share has a unique name within the namespace of the host operating system. Mandatory.

- `-transient`: Specifies that the share is transient, meaning that it can be added and removed at runtime and does not persist after the VM has stopped. Optional.

Shared folders are described in Section 4.3, "Shared Folders".

7.32. VBoxManage guestproperty

The `guestproperty` commands enable you to get or set properties of a running virtual machine. See Section 4.7, "Guest Properties". Guest properties are arbitrary keyword-value string pairs which can be written to and read from by either the guest or the host, so they can be used as a low-volume communication channel for strings, provided that a guest is running and has the Guest Additions installed. In addition, a number of values whose keywords begin with `/VirtualBox/` are automatically set and maintained by the Guest Additions.

The following subcommands are available, where `<vm>` can either be a VM name or a VM UUID, as with the other `VBoxManage` commands:

- `enumerate <vm> [--patterns <pattern>]`: Lists all the guest properties that are available for the given VM, including the value. This list will be very limited if the guest's service process cannot be contacted, for example because the VM is not running or the Guest Additions are not installed.

  If `--patterns <pattern>` is specified, it acts as a filter to only list properties that match the given pattern. The pattern can contain the following wildcard characters:

  - `*` (asterisk): Represents any number of characters. For example, "`/VirtualBox*`" would match all properties beginning with "/VirtualBox".

  - `?` (question mark): Represents a single arbitrary character. For example, "`fo?`" would match both "foo" and "for".

- | (pipe symbol): Can be used to specify multiple alternative patterns. For example, "`s*|t*`" would match anything starting with either "s" or "t".

- `get <vm> <property>`: Retrieves the value of a single property only. If the property cannot be found, for example because the guest is not running, the following message is shown:

```
No value set!
```

- `set <vm> <property> [<value> [--flags <flags>]]`: Enables you to set a guest property by specifying the keyword and value. If `<value>` is omitted, the property is deleted. With `--flags`, you can specify additional behavior. You can combine several flags by separating them with commas.

  - `TRANSIENT`: The value will not be stored with the VM data when the VM exits.

  - `TRANSRESET`: The value will be deleted as soon as the VM restarts or exits.

  - `RDONLYGUEST`: The value can only be changed by the host, but the guest can only read it.

  - `RDONLYHOST`: The value can only be changed by the guest, but the host can only read it.

  - `READONLY`: The value cannot be changed at all.

- `wait <vm> <pattern> --timeout <timeout>`: Waits for a particular value described by the pattern string to change or to be deleted or created. The pattern rules are the same as for the `enumerate` subcommand.

- `delete <vm> <property>`: Deletes a guest property which has been set previously.

7.33. VBoxManage guestcontrol

The `guestcontrol` commands enable control of the guest from the host. See Section 4.9, "Guest Control of Applications" for an introduction.

The `guestcontrol` command has two sets of subcommands. The first set requires guest credentials to be specified, the second does not.

The first set of subcommands is of the following form:

```
VBoxManage guestcontrol <uuid|vmname> <sub-command>
          [--username <name> ]
          [--passwordfile <file> | --password <password>]
          [--domain <domain> ]
          [-v|--verbose] [-q|quiet] ...
```

The common options are as follows:

```
          [--username <name> ]
          [--passwordfile <file> | --password <password>]
          [--domain <domain> ]
          [-v|--verbose] [-q|quiet]
```

The common options for the first set of subcommands are explained in the following list.

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

`--username <name>`   Specifies the user name on guest OS under which the process should run. This user name must already exist on the guest OS. If unspecified, the host user name is used. Optional

`--passwordfile <file>|--password`	Specifies the absolute path on guest file system of password file containing the password for the specified user account or password for the specified user account. Optional. If both are omitted, empty password is assumed.
`--domain <domain>`	User domain for Windows guests. Optional.
`-v|--verbose`	Makes the subcommand execution more verbose. Optional
`-q|--quiet`	Makes the subcommand execution quieter. Optional.

The first set of subcommands are as follows:

- `run`: Executes a guest program, forwarding stdout, stderr, and stdin to and from the host until it completes.

```
VBoxManage guestcontrol <uuid|vmname> run [common-options]
          --exe <path to executable> [--timeout <msec>]
        [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
        [--ignore-operhaned-processes] [--profile]
        [--no-wait-stdout|--wait-stdout]
        [--no-wait-stderr|--wait-stderr]
        [--dos2unix] [--unix2dos]
         -- <program/arg0> [argument1] ... [argumentN]]
```

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--exe <path to executable>`	Specifies the absolute path of the executable on the guest OS file system. Mandatory. For example: `C:\Windows\System32\calc.exe`.
`--timeout <msec>`	Specifies the maximum time, in microseconds, that the executable can run, during which `VBoxManage` receives its output. Optional. If unspecified, `VBoxManage` waits indefinitely for the process to end, or an error occurs.
`-E|--putenv <NAME>=<VALUE>`	Sets, modifies, and unsets environment variables in the environment in which the program will run. Optional. The guest process is created with the standard default guest OS environment. Use this option to modify that default environment. To set or modify a variable use: `<NAME>=<VALUE>`. To unset a variable use: `<NAME>=` Any spaces in names and values should be enclosed by quotes. To set, modify, and unset multiple variables, use multiple instances of the `--E|--putenv` option.
`--unquoted-args`	Disables escaped double quoting, such as \"fred\", on arguments passed to the executed program. Optional.
`--ignore-operhaned-processes`	Ignore orphaned processes. Not yet implemented. Optional.
`--profile`	Use Profile. Not yet implemented. Optional.

`--no-wait-stdout|--wait-stdout`	Does not wait or waits until the guest process ends and receives its exit code and reason/flags. In the case of `--wait-stdout`, `VBoxManage` receives its stdout while the process runs. Optional.
`--no-wait-stderr|--wait-stderr`	Does not wait or waits until the guest process ends and receives its exit code, error messages, and flags. In the case of `--wait-stderr`, `VBoxManage` receives its stderr while the process runs. Optional.
`--dos2unix`	Converts output from DOS/Windows guests to UNIX/Linux-compatible line endings, CR + LF to LF. Not yet implemented. Optional.
`--unix2dos`	Converts output from a UNIX/Linux guests to DOS/Windows-compatible line endings, LF to CR + LF. Not yet implemented. Optional.
`[-- <program/arg0> [<argument1>] ... [<argumentN>]]`	Specifies the program name, followed by one or more arguments to pass to the program. Optional. Any spaces in arguments should be enclosed by quotes.

Note

On Windows there are certain limitations for graphical applications. See Known Limitations.

Examples of using the `guestcontrol run` command are as follows:

```
VBoxManage --nologo guestcontrol "My VM" run --exe "/bin/ls"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout -- -l /usr
```

```
VBoxManage --nologo guestcontrol "My VM" run --exe "c:\\windows\\system32\\ipconfig.exe"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout
```

Note that the double backslashes in the second example are only required on UNIX hosts.

Note

For certain commands a user name of an existing user account on the guest must be specified. Anonymous executions are not supported for security reasons. A user account password, however, is optional and depends on the guest's OS security policy or rules. If no password is specified for a given user name, an empty password will be used. On certain OSes like Windows the security policy may needs to be adjusted in order to allow user accounts with an empty password set. Also, global domain rules might apply and therefore cannot be changed.

Starting at Oracle VM VirtualBox 4.1.2 guest process execution by default is limited to serve up to five guest processes at a time. If a new guest process gets started which would exceed this limit, the oldest not running guest process will be discarded in order to be able to run that new process. Also, retrieving output from this old guest process will not be possible anymore then. If all five guest processes are still active and running, starting a new guest process will result in an appropriate error message.

To raise or lower the guest process execution limit, either use the guest property `/VirtualBox/GuestAdd/VBoxService/--control-procs-max-kept` or `VBoxService` command line by specifying `--control-procs-max-kept` needs to be modified. A restart of the guest OS is

required afterwards. To serve unlimited guest processes, a value of `0` needs to be set, but this is not recommended.

- `start`: Executes a guest program until it completes.

```
VBoxManage guestcontrol <uuid|vmname> start [common-options]
        [--exe <path to executable>] [--timeout <msec>]
        [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
        [--ignore-operhaned-processes] [--profile]
         -- <program/arg0> [argument1] ... [argumentN]]
```

Where the options are as follows:

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--exe <path to executable>`	Specifies the absolute path of the executable on the guest OS file system. Mandatory. For example: `C:\Windows \System32\calc.exe`
`--timeout <msec>`	Specifies the maximum time, in microseconds, that the executable can run. Optional. If unspecified, `VBoxManage` waits indefinitely for the process to end, or an error occurs.
`-E|--putenv <NAME>=<VALUE>`	Sets, modifies, and unsets environment variables in the environment in which the program will run. Optional. The guest process is created with the standard default guest OS environment. Use this option to modify that default environment. To set or modify a variable use: `<NAME>=<VALUE>`. To unset a variable use: `<NAME>=` Any spaces in names and values should be enclosed by quotes. To set, modify, or unset multiple variables, use multiple instances of the `--E|--putenv` option.
`--unquoted-args`	Disables escaped double quoting, such as \"fred\", on arguments passed to the executed program. Optional.
`--ignore-operhaned-processes`	Ignores orphaned processes. Not yet implemented. Optional.
`--profile`	Use a profile. Not yet implemented. Optional.
`[-- <program/arg0> [<argument1>] ... [<argumentN>]]`	Specifies the program name, followed by one or more arguments to pass to the program. Optional. Any spaces in arguments should be enclosed by quotes.

Note

On Windows there are certain limitations for graphical applications. See Known Limitations.

Examples of using the `guestcontrol start` command are as follows:

```
VBoxManage --nologo guestcontrol "My VM" start --exe "/bin/ls"
```

```
            --username foo --passwordfile bar.txt --wait-exit --wait-stdout -- -l /usr
```

```
VBoxManage --nologo guestcontrol "My VM" start --exe "c:\\windows\\system32\\ipconfig.exe"
            --username foo --passwordfile bar.txt --wait-exit --wait-stdout
```

Note that the double backslashes in the second example are only required on UNIX hosts.

Note

For certain commands a user name of an existing user account on the guest must be specified. Anonymous executions are not supported for security reasons. A user account password, however, is optional and depends on the guest's OS security policy or rules. If no password is specified for a given user name, an empty password will be used. On certain OSes like Windows the security policy may needs to be adjusted in order to allow user accounts with an empty password set. Also, global domain rules might apply and therefore cannot be changed.

Starting at Oracle VM VirtualBox 4.1.2 guest process execution by default is limited to serve up to five guest processes at a time. If a new guest process gets started which would exceed this limit, the oldest not running guest process will be discarded in order to be able to run that new process. Also, retrieving output from this old guest process will not be possible anymore then. If all five guest processes are still active and running, starting a new guest process will result in an appropriate error message.

To raise or lower the guest process execution limit, either use the guest property `/VirtualBox/GuestAdd/VBoxService/--control-procs-max-kept` or `VBoxService` command line by specifying `--control-procs-max-kept` needs to be modified. A restart of the guest OS is required afterwards. To serve unlimited guest processes, a value of `0` needs to be set, but this is not recommended.

- `copyfrom`: Copies files from the guest to the host file system. Only available with Guest Additions 4.0 or later installed.

```
VBoxManage guestcontrol <uuid|vmname> copyfrom [common-options]
            [--follow] [--R|recursive]
             --target-directory <host-dst-dir>
             <guest-src0> [<guest-src1> [...]]
```

Where the parameters are as follows:

`<uid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--follow`	Enables symlink following on the guest file system. Optional.
`-R|--recursive`	Enables recursive copying of files and directories from the specified guest file system directory. Optional.
`--target-directory <host-dst-dir>`	Specifies the absolute path of the host file system destination directory. Mandatory. For example: `C:\Temp`.
`<guest-src0> [<guest-src1> [...]]`	Specifies the absolute paths of guest file system files to be copied. Mandatory. For example: `C:\Windows\System32\calc.exe`. Wildcards can be used in the expressions. For example: `C:\Windows\System*\*.dll`.

- `copyto`: Copies files from the host to the guest file system. Only available with Guest Additions 4.0 or later installed.

```
VBoxManage guestcontrol <uuid|vmname> copyto [common-options]
         [--follow] [--R|recursive]
           --target-directory <guest-dst>
            <host-src0> [<host-src1> [...]]
```

Where the parameters are as follows:

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--follow`	Enables symlink following on the host file system. Optional.
`-R|--recursive`	Enables recursive copying of files and directories from the specified host file system directory. Optional.
`--target-directory <guest-dst>`	Specifies the absolute path of the guest file system destination directory. Mandatory. For example: `C:\Temp`.
`<host-src0> [<host-src1> [...]]`	Specifies the absolute paths of host file system files to be copied. Mandatory. For example: `C:\Windows\System32\calc.exe`. Wildcards can be used in the expressions. For example: `C:\Windows\System*\*.dll`.

- `md|mkdir|createdir|createdirectory`: Creates one or more directories on the guest file system. Only available with Guest Additions 4.0 or later installed.

```
VBoxManage guestcontrol <uuid|vmname>  md|mkdir|createdir|createdirectory [common-options]
         [--parents] [--mode <mode>]
           <guest-dir0> [<guest-dir1> [...]]
```

Where the parameters are as follows:

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--parents`	Creates any absent parent directories of the specified directory. Optional.
	For example: If specified directory is `D:\Foo\Bar` and `D:\Foo` is absent, it will be created. In such a case, had the `--parents` option not been used, this command would have failed.
`--mode <mode>`	Specifies the permission mode on the specified directories, and any parents, if the `--parents` option is used. Currently octal modes only, such as. `0755`, are supported.
`<guest-dir0> [<guest-dir1> [...]]`	Specifies a list of absolute paths of directories to be created on guest file system. Mandatory. For example: `D:\Foo\Bar`.
	All parent directories must already exist unless the switch `--parents` is used. For example, in the above example `D:\Foo`. The specified user must have sufficient rights to create the specified directories, and any parents that need to be created.

- `rmdir|removedir|removedirectory`: Deletes specified guest file system directories. Only available with installed Guest Additions 4.3.2 and later.

```
VBoxManage guestcontrol <uuid|vmname> rmdir|removedir|removedirectory [common-options]
```

```
            [--recursive|-R]
             <guest-dir0> [<guest-dir1> [...]]
```

Where the parameters are as follows:

<uuid|vmname>   Specifies the VM UUID or VM name. Mandatory.

--recursive   Recursively removes directories and contents. Optional.

<guest-dir0> [<guest-   Specifies a list of the absolute paths of directories to be deleted on
dir1> [...]]   guest file system. Mandatory. Wildcards are allowed. For example: `D:\Foo\*Bar`. The specified user must have sufficient rights to delete
                                 the specified directories.

- `rm|removefile`: Deletes specified files on the guest file system. Only available with installed Guest Additions 4.3.2 and later.

```
VBoxManage guestcontrol <uuid|vmname> rm|removefile [common-options]
            [-f|--force]
             <guest-file0> [<guest-file1> [...]]
```

Where the parameters are as follows:

<uuid|vmname>   Specifies the VM UUID or VM name. Mandatory.

-f|--force   Enforce operation and override any requests for confirmations.
                                 Optional.

<guest-file0> [<guest-   Specifies a list of absolute paths of files to be deleted on guest file
file1> [...]]   system. Mandatory. Wildcards are allowed. For example: `D:\Foo\Bar\text*.txt`. The specified user should have sufficient rights to
                                 delete the specified files.

- `mv|move|ren|rename`: Renames files and/or directories on the guest file system. Only available with installed Guest Additions 4.3.2 and later.

```
VBoxManage guestcontrol <uuid|vmname> mv|move|ren|rename [common-options]
            <guest-source0> [<guest-source1> [...]] <guest-dest>
```

Where the parameters are as follows:

<uuid|vmname>   Specifies the VM UUID or VM name. Mandatory.

<guest-source0> [<guest-   Specifies absolute paths of files or a single directory to be moved and
source1> [...]]   renamed on guest file system. Mandatory. Wildcards are allowed in
                                 file names. The specified user should have sufficient rights to access
                                 the specified files.

<dest>   Specifies the absolute path of the destination file or directory to
                                 which the files are to be moved. Mandatory. If only one file to be
                                 moved, <dest> can be file or directory, else it must be a directory. The
                                 specified user must have sufficient rights to access the destination file
                                 or directory.

- `mktemp|createtemp|createtemporary`: Creates a temporary file or directory on the guest file system, to assist subsequent copying of files from the host to the guest file systems. By default, the

file or directory is created in the guest's platform specific temp directory. Not currently supported. Only available with installed Guest Additions 4.2 and later.

```
VBoxManage guestcontrol <uuid|vmname> mktemp|createtemp|createtemporary [common-options]
          [--directory] [--secure] [--mode <mode>] [--tmpdir <directory>]
           <template>
```

The parameters are as follows:

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`--directory`	Creates a temporary directory instead of a file, specified by the <template> parameter. Optional.
`--secure`	Enforces secure file and directory creation. Optional. The permission mode is set to `0755`. Operation fails if it cannot be performed securely.
`--mode <mode>`	Specifies the permission mode of the specified directory. Optional. Currently only octal modes, such as `0755`, are supported.
`--tmpdir <directory>`	Specifies the absolute path of the directory on the guest file system where the file or directory specified will be created. Optional. If unspecified, the platform-specific temp directory is used.
`<template>`	Specifies a file name without a directory path, containing at least one sequence of three consecutive X characters, or ending in X. Mandatory.

- `stat`: Displays file or file system statuses on the guest.

```
VBoxManage guestcontrol <uuid|vmname> stat [common-options]
          <file0> [<file1> [...]]
```

Where the parameters are as follows:

`<uuid|vmname>`	Specifies the VM UUID or VM name. Mandatory.
`<file0> [<file1> [...]]`	Specifies absolute paths of files or file systems on the guest file system. Mandatory. For example: `/home/foo/a.out`. The specified user should have sufficient rights to access the specified files or file systems.

The second set of subcommands is of the form:

```
VBoxManage guestcontrol <uuid|vmname> <sub-command>
          [-v|--verbose] [-q|quiet] ...
```

The common options are as follows:

```
          [-v|--verbose] [-q|--quiet]
```

Details of the common options for the second set of subcommands are as follows:

`-v|--verbose`	Makes the subcommand execution more verbose. Optional.
`-q|--quiet`	Makes the subcommand execution quieter. Optional.

The second set of subcommands are as follows:

- `list`: Lists guest control configuration and status data. For example: open guest sessions, guest processes, and files.

```
VBoxManage guestcontrol <uuid|vmname> list [common-opts]
          <all|sessions|processes|files>
```

Where the parameters are as follows:

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

`all|sessions|processes|`   Indicates whether to list all available data or guest sessions,
`files`   processes or files. Mandatory.

- `closeprocess`: Terminates guest processes specified by PIDs running in a guest session, specified by the session ID or name.

```
VBoxManage guestcontrol <uuid|vmname> closeprocess [common-options]
          --session-id <ID> | --session-name <name or pattern>
          <PID0> [<PID1> [...]]
```

Where the parameters are as follows:

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

`--session-id <ID>`   Specifies the guest session by its ID. Optional.

`--session-name <name or`   Specifies the guest session by its name, or multiple sessions using a
`pattern>`   pattern containing wildcards. Optional.

`<PID0> [<PID1> [...]]`   Specifies a list of process identifiers (PIDs) of guest processes to be
                                   terminated. Mandatory.

- `closesession`: Closes specified guest sessions, specified either by session ID or name.

```
VBoxManage guestcontrol <uuid|vmname> closesession [common-options]
          --session-id <ID> | --session-name <name or pattern> | --all
```

Where the parameters are as follows:

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

`--session-id <ID>`   Specifies the guest session to be closed by ID. Optional.

`--session-name <name or`   Specifies the guest session to be closed by name. Optional. Multiple
`pattern>`   sessions can be specified by using a pattern containing wildcards.

`--all`   Close all guest sessions. Optional.

- `updatega|updateadditions|updateguestadditions`: Ugrades Guest Additions already installed on the guest. Only available for already installed Guest Additions 4.0 and later.

```
VBoxManage guestcontrol <uuid|vmname> updatega|updateadditions|updateguestadditions
          [common-options]
          [--source <New .ISO path>]
          [--wait-start]
```

```
            [-- <argument0> [<argument1> [...]]]
```

Where the parameters are as follows:

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

`--source <New .ISO path>`   Specifies the absolute path on the guest file system of the .ISO file for the Guest Additions update. Mandatory.

`--wait-start`   Indicates that `VBoxManage` starts the usual updating process on the guest and then waits until the actual Guest Additions updating begins, at which point `VBoxManage` self-terminates. Optional.

Default behavior is that `VBoxManage` waits for completion of the Guest Additions update before terminating. Use of this option is sometimes necessary, as a running `VBoxManage` can affect the interaction between the installer and the guest OS.

`[-- <argument0>`   Specifies optional command line arguments to be supplied to the
`[<argument1> [...]]]`   Guest Additions updater. Useful for retrofitting features which are not currently installed.

Arguments containing spaces should be enclosed by quotes.

- `watch`: Prints current guest control activity.

```
VBoxManage guestcontrol <uuid|vmname> watch [common-options]
```

Where the parameters are as follows:

`<uuid|vmname>`   Specifies the VM UUID or VM name. Mandatory.

7.34. VBoxManage metrics

This command supports monitoring the usage of system resources. Resources are represented by various metrics associated with the host system or a particular VM. For example, the host system has a `CPU/Load/User` metric that shows the percentage of time CPUs spend executing in user mode over a specific sampling period.

Metric data is collected and retained internally. It may be retrieved at any time with the `VBoxManage metrics query` subcommand. The data is available as long as the background `VBoxSVC` process is alive. That process terminates shortly after all VMs and frontends have been closed.

By default no metrics are collected at all. Metrics collection does not start until `VBoxManage metrics setup` is invoked with a proper sampling interval and the number of metrics to be retained. The interval is measured in seconds. For example, to enable collecting the host processor and memory usage metrics every second and keeping the five most current samples, the following command can be used:

```
VBoxManage metrics setup --period 1 --samples 5 host CPU/Load,RAM/Usage
```

Metric collection can only be enabled for started VMs. Collected data and collection settings for a particular VM will disappear as soon as it shuts down. Use the `VBoxManage metrics list` subcommand to see which metrics are currently available. You can also use the `--list` option with any subcommand that modifies metric settings to find out which metrics were affected.

Note that the `VBoxManage metrics setup` subcommand discards all samples that may have been previously collected for the specified set of objects and metrics.

To enable or disable metrics collection without discarding the data, `VBoxManage metrics enable` and `VBoxManage metrics disable` subcommands can be used. Note that these subcommands expect metrics as parameters, not submetrics such as `CPU/Load` or `RAM/Usage`. In other words enabling `CPU/Load/User` while disabling `CPU/Load/Kernel` is not supported.

The host and VMs have different sets of associated metrics. Available metrics can be listed with `VBoxManage metrics list` subcommand.

A complete metric name may include an aggregate function. The name has the following form: `Category/Metric[/SubMetric][:aggregate]`. For example, `RAM/Usage/Free:min` stands for the minimum amount of available memory over all retained data if applied to the host object.

Subcommands may apply to all objects and metrics or can be limited to one object and a list of metrics. If no objects or metrics are given in the parameters, the subcommands will apply to all available metrics of all objects. You may use an asterisk "`*`" to explicitly specify that the command should be applied to all objects or metrics. Use `host` as the object name to limit the scope of the command to host-related metrics. To limit the scope to a subset of metrics, use a metric list with names separated by commas.

For example, to query metric data on the CPU time spent in user and kernel modes by the virtual machine named `test`, use the following command:

```
VBoxManage metrics query test CPU/Load/User,CPU/Load/Kernel
```

The following list summarizes the available subcommands:

list
: Shows the parameters of the currently existing metrics. Note that VM-specific metrics are only available when a particular VM is running.

setup
: Sets the interval between taking two samples of metric data and the number of samples retained internally. The retained data is available for displaying with the `query` subcommand. The `--list` option shows which metrics have been modified as the result of the command execution.

enable
: Resumes data collection after it has been stopped with the `disable` subcommand. Note that specifying submetrics as parameters will not enable underlying metrics. Use `--list` to find out if the command worked as expected.

disable
: Suspends data collection without affecting collection parameters or collected data. Note that specifying submetrics as parameters will not disable underlying metrics. Use `--list` to find out if the command worked as expected.

query
: Retrieves and displays the currently retained metric data.

> **Note**
>
> The `query` subcommand does not remove or flush retained data. If you query often enough you will see how old samples are gradually being phased out by new samples.

collect
: Sets the interval between taking two samples of metric data and the number of samples retained internally. The collected data is displayed periodically until Ctrl+C is pressed, unless the `--detach` option is

specified. With the `--detach` option, this subcommand operates the same way as `setup` does. The `--list` option shows which metrics match the specified filter.

7.35. VBoxManage natnetwork

NAT networks use the Network Address Translation (NAT) service, which works in a similar way to a home router. It groups systems using it into a network and prevents outside systems from directly accessing those inside, while letting systems inside communicate with each other and outside systems using TCP and UDP over IPv4 and IPv6.

A NAT service is attached to an internal network. Virtual machines to make use of one should be attached to it. The name of an internal network is chosen when the NAT service is created, and the internal network will be created if it does not already exist. The following is an example command to create a NAT network:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable
```

Here, `natnet1` is the name of the internal network to be used and `192.168.15.0/24` is the network address and mask of the NAT service interface. By default, in this static configuration the gateway will be assigned the address 192.168.15.1, the address after the interface address, though this is subject to change.

To add a DHCP server to the NAT network after creation, run the following command:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp on
```

The subcommands for `VBoxManage natnetwork` are as follows:

```
VBoxManage natnetwork add --netname <name>
                         [--network <network>]
                         [--enable|--disable]
                         [--dhcp on|off]
                         [--port-forward-4 <rule>]
                         [--loopback-4 <rule>]
                         [--ipv6 on|off]
                         [--port-forward-6 <rule>]
                         [--loopback-6 <rule>]
```

`VBoxManage natnetwork add`: Creates a new internal network interface, and adds a NAT network service. This command is a prerequisite for enabling attachment of VMs to the NAT network. Parameters are as follows:

`--netname <name>` — Where <name> is the name of the new internal network interface on the host OS.

`--network <network>` — Where <network> specifies the static or DHCP network address and mask of the NAT service interface. The default is a static network address.

`--enable|--disable` — Enables and disables the NAT network service.

`--dhcp on|off` — Enables and disables a DHCP server specified by `--netname`. Use of this option also indicates that it is a DHCP server.

`--port-forward-4 <rule>` — Enables IPv4 port forwarding, with a rule specified by <rule>.

`--loopback-4 <rule>` — Enables the IPv4 loopback interface, with a rule specified by <rule>.

`--ipv6 on|off`   Enables and disables IPv6. The default setting is IPv4, disabling IPv6 enables IPv4.

`--port-forward-6 <rule>`   Enables IPv6 port forwarding, with a rule specified by <rule>.

`--loopback-6 <rule>`   Enables the IPv6 loopback interface, with a rule specified by <rule>.

```
VBoxManage natnetwork remove --netname <name>
```

`VBoxManage natnetwork remove`: Removes a NAT network service. Parameters are as follows:

`--netname <name>`   Where <name> specifies an existing NAT network service. Does not remove any DHCP server enabled on the network.

```
VBoxManage natnetwork modify --netname <name>
                             [--network <network>]
                             [--enable|--disable]
                             [--dhcp on|off]
                             [--port-forward-4 <rule>]
                             [--loopback-4 <rule>]
                             [--ipv6 on|off]
                             [--port-forward-6 <rule>]
                             [--loopback-6 <rule>]
```

`VBoxManage natnetwork modify`: Modifies an existing NAT network service. Parameters are as follows:

`--netname <name>`   Where <name> specifies an existing NAT network service.

`--network <network>`   Where <network> specifies the new static or DHCP network address and mask of the NAT service interface. The default is a static network address.

`--enable|--disable`   Enables and disables the NAT network service.

`--dhcp on|off`   Enables and disables a DHCP server. If a DHCP server is not present, using enable adds a new DHCP server.

`--port-forward-4 <rule>`   Enables IPv4 port forwarding, with a rule specified by <rule>.

`--loopback-4 <rule>`   Enables the IPv4 loopback interface, with a rule specified by <rule>.

`--ipv6 on|off`   Enables and disables IPv6. The default setting is IPv4, disabling IPv6 enables IPv4.

`--port-forward-6 <rule>`   Enables IPv6 port forwarding, with a rule specified by <rule>.

`--loopback-6 <rule>`   Enables IPv6 loopback interface, with a rule specified by <rule>.

```
VBoxManage natnetwork start --netname <name>
```

`VBoxManage natnetwork start`: Starts the specified NAT network service and any associated DHCP server. Parameters are as follows:

`--netname <name>`   Where <name> specifies an existing NAT network service.

```
VBoxManage natnetwork stop --netname <name>
```

`VBoxManage natnetwork stop`: Stops the specified NAT network service and any DHCP server. Parameters are as follows:

`--netname <name>`   Where <name> specifies an existing NAT network service.

```
VBoxManage natnetwork list [<pattern>]
```

`VBoxManage natnetwork list`: Lists all NAT network services, with optional filtering. Parameters are as follows:

`[<pattern>]`   Where <pattern> is an optional filtering pattern.

7.36. VBoxManage hostonlyif

The `hostonlyif` command enables you to change the IP configuration of a host-only network interface. For a description of host-only networking, see Section 6.7, "Host-Only Networking". Each host-only interface is identified by a name and can either use the internal DHCP server or a manual IP configuration, both IP4 and IP6.

The following list summarizes the available subcommands:

`ipconfig "<name>"`   Configures a host-only interface.

`create`   Creates a new vboxnet<N> interface on the host OS. This command is essential before you can attach VMs to a host-only network.

`remove vboxnet<N>`   Removes a vboxnet<N> interface from the host OS.

7.37. VBoxManage usbdevsource

The `usbdevsource` commands enable you to add and remove USB devices globally.

The following command adds a USB device.

```
VBoxManage usbdevsource add <source name>
                           --backend <backend>
                           --address <address>
```

Where the command line options are as follows:

- `<source name>`: Specifies the ID of the source USB device to be added. Mandatory.

- `--backend <backend>`: Specifies the USB proxy service backend to use. Mandatory.

- `--address <address>`: Specifies the backend specific address. Mandatory.

The following command removes a USB device.

```
VBoxManage usbdevsource remove <source name>
```

Where the command line options are as follows:

- `<source name>`: Specifies the ID of the source USB device to be removed. Mandatory.

7.38. VBoxManage unattended

Unattended guest OS installation.

7.38.1. Synopsis

`VBoxManage unattended detect` {--iso=*install-iso*} [--machine-readable]

`VBoxManage unattended install` { *uuid*/*vmname* } {--iso=*install-iso*} [--user=*login*] [--password=*password*] [--password-file=*file*] [--full-user-name=*name*] [--key=*product-key*] [--install-additions] [--no-install-additions] [--additions-iso=*add-iso*] [--install-txs] [--no-install-txs] [--validation-kit-iso=*testing-iso*] [--locale=*ll_CC*] [--country=*CC*] [--time-zone=*tz*] [--hostname=*fqdn*] [--package-selection-adjustment=*keyword*] [--dry-run] [--auxiliary-base-path=*path*] [--image-index=*number*] [--script-template=*file*] [--post-install-template=*file*] [--post-install-command=*command*] [--extra-install-kernel-parameters=*params*] [--language=*lang*] [--start-vm=*session-type*]

7.38.2. Description

7.38.2.1. unattended detect

`VBoxManage unattended detect` {--iso=*install-iso*} [--machine-readable]

Detects the guest operating system (OS) on the specified installation ISO and displays the result. This can be used as input when creating a VM for the ISO to be installed in.

`--iso=`*install-iso*	The installation ISO to run the detection on.
`--machine-readable`	Produce output that is simpler to parse from a script.

7.38.2.2. unattended install

`VBoxManage unattended install` { *uuid*/*vmname* } {--iso=*install-iso*} [--user=*login*] [--password=*password*] [--password-file=*file*] [--full-user-name=*name*] [--key=*product-key*] [--install-additions] [--no-install-additions] [--additions-iso=*add-iso*] [--install-txs] [--no-install-txs] [--validation-kit-iso=*testing-iso*] [--locale=*ll_CC*] [--country=*CC*] [--time-zone=*tz*] [--hostname=*fqdn*] [--package-selection-adjustment=*keyword*] [--dry-run] [--auxiliary-base-path=*path*] [--image-index=*number*] [--script-template=*file*] [--post-install-template=*file*] [--post-install-command=*command*] [--extra-install-kernel-parameters=*params*] [--language=*lang*] [--start-vm=*session-type*]

Reconfigures the specified VM for installation and optionally starts it up.

uuid/*vmname*	Either the UUID or the name (case sensitive) of a VM.
`--iso=`*install-iso*	The installation ISO to run the detection on.
`--user=`*login*	The login name. (default: vboxuser)
`--password=`*password*	The login password. This is used for the user given by `--user` as well as the root/administrator user. (default: changeme)
`--password-file=`*file*	Alternative to `--password` for providing the password. Special filename `stdin` can be used to read the password from standard input.
`--full-user-name=`*name*	The full user name. (default: --user)

`--key=`*`product-key`*	The guest OS product key. Not all guest OSes requires this.
`--install-additions`, `--no-install-additions`	Whether to install the VirtualBox guest additions. (default: --no-install-addations)
`--additions-iso=`*`add-iso`*	Path to the VirtualBox guest additions ISO. (default: installed/downloaded GAs)
`--install-txs`, `--no-install-txs`	Whether to install the test execution service (TXS) from the VirtualBox ValidationKit. This is useful when preparing VMs for testing or similar. (default: --no-install-txs)
`--validation-kit-iso=`*`testing-iso`*	Path to the VirtualBox ValidationKit ISO. This is required if `--install-txs` is specified.
`--locale=`*`ll_CC`*	The base locale specification for the guest, like en_US, de_CH, or nn_NO. (default: host or en_US)
`--country=`*`CC`*	The two letter country code if it differs from the specified by `--location`.
`--time-zone=`*`tz`*	The time zone to set up the guest OS with. (default: host time zone or UTC)
`--hostname=`*`fqdn`*	The fully qualified domain name of the guest machine. (default: *`vmname`*.myguest.virtualbox.org)
`--package-selection-adjustment=`*`keyword`*	Adjustments to the guest OS packages/components selection. This can be specfied more than once. Currently the only recognized keyword is `minimal` which triggers a minimal installation for some of the guest OSes.
`--dry-run`	Do not create any files or make any changes to the VM configuration.
`--start-vm=`*`session-type`*	Start the VM using the front end given by *`session-type`*. This is the same as the `--type` option for the `startvm` command, but we have add `none` for indicating that the VM should not be started. (default: `none`)

Advanced options:

`--auxiliary-base-path=`*`path`*	The path prefix to the media related files generated for the installation. (default: *`vm-config-dir`*/Unattended-*`vm-uuid`*-)
`--image-index=`*`number`*	Windows installation image index. (default: 1)
`--script-template=`*`file`*	The unattended installation script template. (default: IMachine::OSTypeId dependent)
`--post-install-template=`*`file`*	The post installation script template. (default: IMachine::OSTypeId dependent)
`--post-install-command=`*`command`*	A single command to run after the installation is completed. The exact format and exactly when this is run is guest OS installer dependent.
`--extra-install-kernel-parameters=`*`params`*	List of extra linux kernel parameters to use during the installation. (default: IMachine::OSTypeId dependent)

`--language=`*`lang`*   Specifies the UI language for a Windows installation. The *lang* is generally on the form {ll}-{CC}. See detectedOSLanguages results from `VBoxManage unattended detect`. (default: detectedOSLanguages[0])

7.39. VBoxManage snapshot

Manage Oracle VM VirtualBox virtual machine snapshots.

7.39.1. Synopsis

`VBoxManage snapshot` { *uuid*/*vmname* }

`VBoxManage snapshot` { *uuid*/*vmname* } take { *snapshot-name* } [--description=*description*] [--live] [--uniquename Number,Timestamp,Space,Force]

`VBoxManage snapshot` { *uuid*/*vmname* } delete { *snapshot-name* }

`VBoxManage snapshot` { *uuid*/*vmname* } restore { *snapshot-name* }

`VBoxManage snapshot` { *uuid*/*vmname* } restorecurrent

`VBoxManage snapshot` { *uuid*/*vmname* } edit { *snapshot-name* | --current } [--description=*description*] [--name=*new-name*]

`VBoxManage snapshot` { *uuid*/*vmname* } list [[--details] | [--machinereadable]]

`VBoxManage snapshot` { *uuid*/*vmname* } showvminfo { *snapshot-name* }

7.39.2. Description

The `VBoxManage snapshot` command manages snapshots.

Oracle VM VirtualBox uses the snapshot to capture the state of a virtual machine (VM). You can later use the snapshot to revert to the state described by the snapshot.

A snapshot is a complete copy of a VM's settings. If you take the snapshot while the VM is running, the snapshot also includes the VM's state file.

After you take a snapshot, Oracle VM VirtualBox creates a *differencing hard disk* for each normal hard disk that is associated with the host machine. When you restore a snapshot, Oracle VM VirtualBox uses these differencing files to quickly reset the contents of the VM's virtual hard disks.

For each `VBoxManage snapshot` command, you must specify the name or the universal unique identifier (UUID) of the VM for which you want to take a snapshot.

7.39.2.1. General Command Operand

uuid/*vmname*   Specifies the UUID or name of the VM.

7.39.2.2. Take a Snapshot of a Virtual Machine

`VBoxManage snapshot` { *uuid*/*vmname* } take { *snapshot-name* } [--description=*description*] [--live] [--uniquename Number,Timestamp,Space,Force]

The `VBoxManage snapshot take` command takes a snapshot of the current state of the VM. You must supply a name for the snapshot and can optionally supply a description. The new snapshot is inserted into the snapshots tree as a child of the current snapshot and then becomes the new current snapshot.

`--` `description=description`	Specifies a description of the snapshot.
`--live`	Specifies that the VM is not stopped while you create the snapshot. This operation is know as live snapshotting.
`--uniquename` `Number,Timestamp,Space,Force`	TBD.
`snapshot-name`	Specifies the name of the snapshot to create.

7.39.2.3. Delete a Snapshot

`VBoxManage snapshot` { `uuid/vmname` } delete { `snapshot-name` }

The `VBoxManage snapshot delete` command removes the specified snapshot.

The delete operation may take some time to finish. This is because the differencing images that are associated with the snapshot may need to be merged with their child differencing images.

`snapshot-name`	Specifies the UUID or name of the snapshot.

7.39.2.4. Restore a Snapshot

`VBoxManage snapshot` { `uuid/vmname` } restore { `snapshot-name` }

The `VBoxManage snapshot restore` command restores the specified snapshot. This operation resets the VM's settings and current state to that of the snapshot. The state of the VM on which you restore a snapshot is lost. When restored, the specified snapshot becomes the new current snapshot and subsequent snapshots are children of that snapshot.

`snapshot-name`	Specifies the UUID or name of the snapshot.

7.39.2.5. Restore the Current Snapshot

`VBoxManage snapshot` { `uuid/vmname` } restorecurrent

The `VBoxManage snapshot restorecurrent` command restores the current snapshot. The current snapshot is the one from which the current state is derived. This command is equivalent to using the `VBoxManage snapshot restore` command and specifying the name or UUID of the current snapshot.

7.39.2.6. Change the Name or Description of an Existing Snapshot

`VBoxManage snapshot` { `uuid/vmname` } edit { `snapshot-name` | --current } [--description=`description`] [--name=`new-name`]

The `VBoxManage snapshot edit` command enables you to change the name or the description of a specified snapshot.

`snapshot-name`	Specifies the UUID or name of the snapshot to edit.
	This option is mutually exclusive with the `--current` option.
`--current`	Specifies that you update the current version of the snapshot.

This option is mutually exclusive with a specific snapshot name or its UUID.

`--description=description` Specifies a new description for the snapshot.

`--name=new-name` Specifies a new name for the snapshot.

7.39.2.7. List the Snapshots

`VBoxManage snapshot` { `uuid`/`vmname` } list [[--details] | [--machinereadable]]

The `VBoxManage snapshot list` command lists all the snapshots for a VM.

`--details` Specifies that the output shows detailed information about the snapshot.

This option is mutually exclusive with the `--machinereadable` option.

`--machinereadable` Specifies that the output is shown in a machine-readable format.

This option is mutually exclusive with the `--details` option.

7.39.2.8. Show Information About a Snapshot's Settings

`VBoxManage snapshot` { `uuid`/`vmname` } showvminfo { `snapshot-name` }

The `VBoxManage snapshot showvminfo` command enables you to view the VM settings that are part of an existing snapshot.

`snapshot-name` Specifies the UUID or name of the snapshot.

7.39.3. Examples

The following command creates a snapshot of the `ol7u4` VM. The snapshot is called `ol7u4-snap-001`. The command uses the `--description` option to provide a description of the snapshot contents.

```
$ VBoxManage snapshot ol7u4 take ol7u4-snap-001 \
--description="Oracle Linux 7.4"
```

The following command lists the snapshots for the `ol7u4` VM.

```
$ VBoxManage snapshot ol7u4 list
```

The following command changes the description for the `ol7u4-snap-001` snapshot of the `ol7u4` VM.

```
$ VBoxManage snapshot ol7u4 edit ol7u4-snap-001 \
--description="Oracle Linux 7.4 with UEK4 kernel"
```

The following command shows VM settings for the `ol7u1-snap-001` snapshot of the `ol7u4` VM.

```
$ VBoxManage snapshot ol7u4 showvminfo ol7u4-snap-001
Name:          ol7u4
Groups:        /
Guest OS:      Oracle (64-bit)
UUID:          43349d78-2ab3-4cb8-978f-0e755cd98090
Config file:   C:\Users\user1\VirtualBox VMs\ol7u4\ol7u4.vbox
...
Snapshots:
```

```
    Name: ol7u4-snap-001 (UUID: 1cffc37d-5c37-4b86-b9c5-a0f157a55f43)
    Description: Oracle Linux 7.4 with UEK4 kernel
```

7.40. VBoxManage clonevm

Create a clone of an existing Oracle VM VirtualBox virtual machine.

7.40.1. Synopsis

VBoxManage clonevm { *vmname* | *uuid* } [--basefolder=*basefolder*] [--group=*group*,...] [--mode=machine | --mode=machinechildren | --mode=all] [--name=*name*] [--options=*option*,...] [--register] [--snapshot=*snapshot-name*] [--uuid=*uuid*]

7.40.2. Description

The VBoxManage clonevm command creates a clone of an existing virtual machine (VM). The clone can be a full copy of the VM or a linked copy of a VM.

You must specify the name or the universal unique identifier (UUID) of the VM you want to clone.

7.40.3. Command Operand and Options

The following list describes the operand and the options that you can use with the VBoxManage clonevm command:

vmname	*uuid*	Specifies the name or UUID of the VM to clone.
--basefolder=*basefolder*	Specifies the name of the folder in which to save the configuration for the new VM.
--groups=*group*,...	Assigns the clone to the specified group or groups. If you specify more than one group, separate each group name with a comma.
	Note that each group is identified by a group ID that starts with a slash character (/) so that groups can be nested. By default, a clone is always assigned membership to the / group.
--mode=machine | machineandchildren|all	Specifies which of the following cloning modes to use:

- machine mode clones the current state of the existing VM without any snapshots. This is the default mode.

- machineandchildren mode clones the snapshot specified by by the --snapshot option and all child snapshots.

- all mode clones all snapshots and the current state of the existing VM.

--name=*name*	Specifies a new name for the new VM. The default value is *original-name* Clone where *original-name* is the original name of the VM.
--options=*option*,...	Specifies how to create the new clone.
	The --options argument can be used multiple times to enable multiple options, or the options can be given as a comma separated list. The options are case insensitive.

The following options (case-insensitive) are recognized:

`Link`	Creates a linked clone from a snapshot only.
`KeepAllMACs`	Specifies that the new clone reuses the MAC addresses of each virtual network card from the existing VM. If you do not specify this option or the `--options=keepnatmacs` option, the default behavior is to reinitialize the MAC addresses of each virtual network card.
`KeepNATMACs`	Specifies that the new clone reuses the MAC addresses of each virtual network card from the existing VM when the network type is NAT. If you do not specify this option or the `KeepAllMACs` option, the default behavior is to reinitialize the MAC addresses of each virtual network card.
`KeepDiskNames`	Specifies that the new clone reuses the disk image names from the existing VM. By default, disk images are renamed.
`KeepHwUUIDs`	Specifies that the new clone reuses the hardware IDs from the existing VM. By default, new UUIDs are used.

`--register`	Automatically registers the new clone in this Oracle VM VirtualBox installation. You can manually register the new VM later by using the `VBoxManage registervm` command. See Section 7.6, "VBoxManage registervm/unregistervm".
`--snapshot=snapshot-name`	Specifies the snapshot on which to base the new VM. By default, the clone is created from the current state of the specified VM.
`--uuid=uuid`	Specifies the UUID for the new VM. Ensure that this ID is unique for the Oracle VM VirtualBox instance if you decide to register this new VM. By default, Oracle VM VirtualBox provides a new UUID.

7.40.4. Examples

The following command creates and registers an exact clone of the `ol7` VM. The clone is called `ol7-dev-001`.

The new clone includes all of the source VM's snapshots. The new VM also reuses all network interface MAC addresses, disk names, and UUIDs from the source VM.

```
$ VBoxManage clonevm ol7 --name="ol7-dev-001" --register --mode=all \
```

```
    --options=keepallmacs --options=keepdisknames --options=keephwuuids
```

The following command creates and registers a clone of the `Snapshot 1` snapshot of the `ol7` VM. The clone is called `ol7-dev-002`.

```
$ VBoxManage clonevm ol7 --name="ol7-dev-002" --register --snapshot="Snapshot 1"
```

7.40.5. See Also

Section 7.6, "VBoxManage registervm/unregistervm"

7.41. VBoxManage extpack

Extension package management.

7.41.1. Synopsis

`VBoxManage extpack install` [--replace] { *tarball* }

`VBoxManage extpack uninstall` [--force] { *name* }

`VBoxManage extpack cleanup`

7.41.2. Description

7.41.2.1. extpack install

`VBoxManage extpack install` [--replace] { *tarball* }

Installs a new extension pack on the system. This command will fail if an older version of the same extension pack is already installed. The `--replace` option can be used to uninstall any old package before the new one is installed.

`--replace`   Uninstall existing extension pack version.

tarball   The file containing the extension pack to be installed.

7.41.2.2. extpack uninstall

`VBoxManage extpack uninstall` [--force] { *name* }

Uninstalls an extension pack from the system. The subcommand will also succeed in the case where the specified extension pack is not present on the system. You can use `VBoxManage list extpacks` to show the names of the extension packs which are currently installed.

`--force`   Overrides most refusals to uninstall an extension pack

name   The name of the extension pack to be uninstalled.

7.41.2.3. extpack cleanup

`VBoxManage extpack cleanup`

Used to remove temporary files and directories that may have been left behind if a previous install or uninstall command failed.

7.41.3. Examples

How to list extension packs:

```
$ VBoxManage list extpacks
Extension Packs: 1
Pack no. 0:    Oracle VM VirtualBox Extension Pack
Version:       4.1.12
Revision:      77218
Edition:
Description:   USB 2.0 Host Controller, VirtualBox RDP, PXE ROM with E1000 support.
VRDE Module:   VBoxVRDP
Usable:        true
Why unusable:
```

How to remove an extension pack:

```
$ VBoxManage extpack uninstall "Oracle VM VirtualBox Extension Pack"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully uninstalled "Oracle VM VirtualBox Extension Pack".
```

7.42. VBoxManage dhcpserver

DHCP server management.

7.42.1. Synopsis

VBoxManage dhcpserver add { --network=*netname* | --interface=*ifname* } {--server-ip=*address*} {--netmask=*mask*} {--lower-ip=*address*} {--upper-ip=*address*} { --enable | --disable }
[[--global] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--group=*name*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--incl-mac=*address*...] [--excl-mac=*address*...] [--incl-mac-wild=*pattern*...] [--excl-mac-wild=*pattern*...] [--incl-vendor=*string*...] [--excl-vendor=*string*...] [--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--incl-user=*string*...] [--excl-user=*string*...] [--incl-user-wild=*pattern*...] [--excl-user-wild=*pattern*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--vm=*name|uuid*} [--nic=*1-N*] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] ...]
[{--mac-address=*address*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] ...]

VBoxManage dhcpserver modify { --network=*netname* | --interface=*ifname* } [--server-ip=*address*] [--lower-ip=*address*] [--upper-ip=*address*] [--netmask=*mask*] [--enable | --disable]
[[--global] [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--remove-config] ...]
[{--group=*name*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--del-mac=*address*...] [--incl-mac=*address*...] [--excl-mac=*address*...]

[--del-mac-wild=*pattern*...] [--incl-mac-wild=*pattern*...] [--excl-mac-wild=*pattern*...] [--del-vendor=*string*...] [--incl-vendor=*string*...] [--excl-vendor=*string*...] [--del-vendor-wild=*pattern*...] [--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--del-user=*string*...] [--incl-user=*string*...] [--excl-user=*string*...] [--del-user-wild=*pattern*...] [--incl-user-wild=*pattern*...] [--excl-user-wild=*pattern*...] [--zap-conditions] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--remove-config] ...]
[{--vm=*name*|*uuid*} [--nic=*1-N*] [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...]
[{--mac-address=*address*} [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...]

VBoxManage dhcpserver remove { --network=*netname* | --interface=*ifname* }

VBoxManage dhcpserver restart { --network=*netname* | --interface=*ifname* }

VBoxManage dhcpserver findlease { --network=*netname* | --interface=*ifname* } {--mac-address=*mac*}

7.42.2. Description

The `dhcpserver` commands enable you to control the DHCP server that is built into VirtualBox. You may find this useful when using internal or host-only networking. Theoretically, you can also enable it for a bridged network, but that may cause conflicts with other DHCP servers in your physical network.

7.42.2.1. Common options

The subcommands of `dhcpserver` all operate on an internal network that can be identified via its name or in the host-only case via the host-only interface name:

--network=*netname*   The internal network name. This is the same as you would use as value to the `VBoxManage modifyvm --intnet` option when configuring a VM for internal networking. Or you see as VBoxNetworkName in the output from `VBoxManage list intnets`, `VBoxManage list natnets`, or `VBoxManage list hostonlyifs`.

--interface=*ifname*   The host only interface name. This would be same value as you would use for the `VBoxManage modifyvm --hostonlyadapter` option when configuring a VM to use a host-only network. The value can also be found in the Name row in `VBoxManage list hostonlyifs`.

7.42.2.2. dhcpserver add

VBoxManage dhcpserver add { --network=*netname* | --interface=*ifname* } {--server-ip=*address*} {--netmask=*mask*} {--lower-ip=*address*} {--upper-ip=*address*} { --enable | --disable }
[[--global] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--group=*name*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--incl-mac=*address*...] [--excl-mac=*address*...] [--incl-mac-wild=*pattern*...] [--excl-mac-wild=*pattern*...] [--incl-vendor=*string*...] [--excl-vendor=*string*...] [--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--incl-

user=*string*...] [--excl-user=*string*...] [--incl-user-wild=*pattern*...] [--excl-user-wild=*pattern*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] ...]
[{--vm=*name* | *uuid*} [--nic=*1-N*] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] ...]
[{--mac-address=*address*} [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] ...]

Adds a new DHCP server to a network or host-only interface.

Options configuring the DHCP server core:

--server-ip=*address*	The IP address the DHCP server should use.
--lower-ip=*address*, --upper-ip=*address*	The IP address range for the DHCP server to manage. This should not include the address of the DHCP server itself, but it must be in the same network as it. The boundraries are inclusive, so both the lower and upper addresses will be handed out to clients.
--netmask=*mask*	The network mask. Typically 255.255.255.0.
--enable, --disable	Whether to enable the DHCP server or disable it. If not specified, the server will be created in disabled state and no IP addresses handed out.

Options selecting the scope:

--global	Set the configuration scope to global. Any subsequent --set-opt options will be apply to all the DHCP clients.
--vm=*vmname*	*uuid*	Set the configuration scope to the first NIC of the specified VM. Any subsequent --set-opt options will apply just to that interface, nothing else.
--nic=*1-N*	Set the configuration scope to a NIC other than first of the VM specified the in --vm.
--mac-address=*address*	Set the configuration scope to the specified MAC address.
--group=*name*	Set the configuration scope to the specified group.

Options configuring the currently selected scope:

--set-opt=*dhcp-opt-no value*	Adds the specified DHCP option number (0-255) and value. The value format is option specific (typically human readable) and will be validated by the API and the DHCP server.
--set-opt-hex=*dhcp-opt-no hexstring*	Adds the specified DHCP option number (0-255) and value. The option value is specified as a raw series of hex bytes, optionally separated by colons. No validation is performed on these by the API or the DHCP server, they will be pass as specified to the client.
--force-opt=*dhcp-opt-no*	Forces the specified DHCP option number (0-255) onto to be sent to the client whether it requested it or not (provided the option is configured with a value at some level).

`--suppress-opt=dhcp-opt-no`	Prevents the specified DHCP option number (0-255) from being sent to the client when present in this or a high configuration scope.
`--min-lease-time=seconds`	Sets the minimum lease time for the current scope in seconds. Zero means taking the value from a higher option level or use default.
`--default-lease-time=seconds`	Sets the default lease time for the current scope in seconds. Zero means taking the value from a higher option level or use default.
`--max-lease-time=seconds`	Sets the maximum lease time for the current scope in seconds. Zero means taking the value from a higher option level or use default.
`--fixed-address=address`	Fixed address assignment for a `--vm` or `--mac-address` configuration scope. Any empty `address` turns it back to dynamic address assignment.

Options configuring group membership conditions (excludes overrides includes):

`--incl-mac=address`	Include the specific MAC address in the group.
`--excl-mac=address`	Exclude the specific MAC address from the group.
`--incl-mac-wild=pattern`	Include the specific MAC address pattern in the group.
`--excl-mac-wild=pattern`	Exclude the specific MAC address pattern from the group.
`--incl-vendor=string`	Include the specific vendor class ID in the group.
`--excl-vendor=string`	Exclude the specific vendor class ID from the group.
`--incl-vendor-wild=pattern`	Include the specific vendor class ID pattern in the group.
`--excl-vendor-wild=pattern`	Exclude the specific vendor class ID pattern from the group.
`--incl-user=string`	Include the specific user class ID in the group.
`--excl-user=string`	Exclude the specific user class ID from the group.
`--incl-user-wild=pattern`	Include the specific user class ID pattern in the group.
`--excl-user-wild=pattern`	Exclude the specific user class ID pattern from the group.

7.42.2.3. dhcpserver modify

```
VBoxManage dhcpserver modify { --network=netname | --interface=ifname } [--server-ip=address]
[--lower-ip=address] [--upper-ip=address] [--netmask=mask] [ --enable | --disable ]
[ [--global] [--del-opt=dhcp-opt-no...] [--set-opt=dhcp-opt-no value...] [--set-opt-hex=dhcp-opt-no
hexstring...] [--force-opt=dhcp-opt-no...] [--unforce-opt=dhcp-opt-no...] [--supress-opt=dhcp-opt-
no...] [--unsupress-opt=dhcp-opt-no...] [--min-lease-time=seconds] [--default-lease-time=seconds] [--
max-lease-time=seconds] [--remove-config] ...]
[ {--group=name} [--set-opt=dhcp-opt-no value...] [--set-opt-hex=dhcp-opt-no hexstring...]
[--force-opt=dhcp-opt-no...] [--unforce-opt=dhcp-opt-no...] [--supress-opt=dhcp-opt-no...] [--
unsupress-opt=dhcp-opt-no...] [--del-mac=address...] [--incl-mac=address...] [--excl-mac=address...]
[--del-mac-wild=pattern...] [--incl-mac-wild=pattern...] [--excl-mac-wild=pattern...] [--del-
vendor=string...] [--incl-vendor=string...] [--excl-vendor=string...] [--del-vendor-wild=pattern...]
```

[--incl-vendor-wild=*pattern*...] [--excl-vendor-wild=*pattern*...] [--del-user=*string*...] [--incl-user=*string*...] [--excl-user=*string*...] [--del-user-wild=*pattern*...] [--incl-user-wild=*pattern*...] [--excl-user-wild=*pattern*...] [--zap-conditions] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--remove-config] ...]

[{--vm=*name | uuid*} --nic=*1-N* [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...]

[{--mac-address=*address*} [--del-opt=*dhcp-opt-no*...] [--set-opt=*dhcp-opt-no value*...] [--set-opt-hex=*dhcp-opt-no hexstring*...] [--force-opt=*dhcp-opt-no*...] [--unforce-opt=*dhcp-opt-no*...] [--supress-opt=*dhcp-opt-no*...] [--unsupress-opt=*dhcp-opt-no*...] [--min-lease-time=*seconds*] [--default-lease-time=*seconds*] [--max-lease-time=*seconds*] [--fixed-address=*address*] [--remove-config] ...]

This modifies an existing DHCP server configuration. It takes the same options as the `add` command with the addition of the following on scope configuration:

`--del-opt=`*dhcp-opt-no*	Counterpart to `--set-opt` that will cause the specified DHCP option number (0-255) to be deleted from the server settings. Like with `--set-opt` the scope of the deletion is governed by the `--global`, `--vm`, `--mac-address` and `--group` options.
`--unforce-opt=`*dhcp-opt-no*	Removes the specified DHCP option number (0-255) from the forced option list (i.e. the reverse of `--force-opt`). Like with `--set-opt` the scope of the deletion is governed by the `--global`, `--vm`, `--mac-address` and `--group` options.
`--unsuppress-opt=`*dhcp-opt-no*	Removes the specified DHCP option number (0-255) from the supressed option list (i.e. the reverse of `--suppress-opt`). Like with `--set-opt` the scope of the deletion is governed by the `--global`, `--vm`, `--mac-address` and `--group` options.
`--remove-config`	Removes the configuration currently being scoped. The `--global` scope is not removable. The configuration scope will change to `--global` after this option.

And the addition of these group membership condition options:

`--del-mac=`*address*	Delete the specific MAC address from the group conditions.
`--del-mac-wild=`*pattern*	Delete the specific MAC address pattern from the group conditions.
`--del-vendor=`*string*	Delete the specific vendor class ID from the group conditions.
`--del-vendor-wild=`*pattern*	Delete the specific vendor class ID pattern from the group conditions.
`--del-user=`*string*	Delete the specific user class ID pattern from the group conditions.
`--del-user-wild=`*pattern*	Delete the specific user class ID pattern from the group conditions.
`--zap-conditions`	Deletes all the group conditions.

7.42.2.4. dhcpserver remove

`VBoxManage dhcpserver remove { --network=`*netname* `| --interface=`*ifname* `}`

Removes the specified DHCP server.

7.42.2.5. dhcpserver restart

`VBoxManage dhcpserver restart` { --network=*netname* | --interface=*ifname* }

Restarts the specified DHCP server. The DHCP server must be running.

7.42.2.6. dhcpserver findlease

`VBoxManage dhcpserver findlease` { --network=*netname* | --interface=*ifname* } {--mac-address=*mac*}

Performs a lease database lookup. This is mainly for getting the IP address of a running VM.

`--mac-address=`*mac*	The MAC address to lookup in the lease database.

7.42.2.7. Common DHCP Options:

1 - SubnetMask	IPv4 netmask. Set to the value of the --netmask option by default.
2 - TimeOffset	UTC offset in seconds (32-bit decimal value).
3 - Routers	Space separated list of IPv4 router addresses.
4 - TimeServers	Space separated list of IPv4 time server (RFC 868) addresses.
5 - NameServers	Space separated list of IPv4 name server (IEN 116) addresses.
6 - DomainNameServers	Space separated list of IPv4 DNS addresses.
7 - LogServers	Space separated list of IPv4 log server addresses.
8 - CookieServers	Space separated list of IPv4 cookie server (RFC 865) addresses.
9 - LPRServers	Space separated list of IPv4 line printer server (RFC 1179) addresses.
10 - ImpressServers	Space separated list of IPv4 imagen impress server addresses.
11 - ResourseLocationServers	Space separated list of IPv4 resource location (RFC 887) addresses.
12 - HostName	The client name. See RFC 1035 for character limits.
13 - BootFileSize	Number of 512 byte blocks making up the boot file (16-bit decimal value).
14 - MeritDumpFile	Client core file.
15 - DomainName	Domain name for the client.
16 - SwapServer	IPv4 address of the swap server that the client should use.
17 - RootPath	The path to the root disk the client should use.
18 - ExtensionPath	Path to a file containing additional DHCP options (RFC2123).
19 - IPForwarding	Whether IP forwarding should be enabled by the client (boolean).
20 - OptNonLocalSourceRouting	Whether non-local datagrams should be forwarded by the client (boolean)

21 - PolicyFilter	List of IPv4 addresses and masks paris controlling non-local source routing.
22 - MaxDgramReassemblySize	The maximum datagram size the client should reassemble (16-bit decimal value).
23 - DefaultIPTTL	The default time-to-leave on outgoing (IP) datagrams (8-bit decimal value).
24 - PathMTUAgingTimeout	RFC1191 path MTU discovery timeout value in seconds (32-bit decimal value).
25 - PathMTUPlateauTable	RFC1191 path MTU discovery size table, sorted in ascending order (list of 16-bit decimal values).
26 - InterfaceMTU	The MTU size for the interface (16-bit decimal value).
27 - AllSubnetsAreLocal	Indicates whether the MTU size is the same for all subnets (boolean).
28 - BroadcastAddress	Broadcast address (RFC1122) for the client to use (IPv4 address).
29 - PerformMaskDiscovery	Whether to perform subnet mask discovery via ICMP (boolean).
30 - MaskSupplier	Whether to respond to subnet mask requests via ICMP (boolean).
31 - PerformRouterDiscovery	Whether to perform router discovery (RFC1256) (boolean).
32 - RouterSolicitationAddress	Where to send router solicitation requests (RFC1256) (IPv4 address).
33 - StaticRoute	List of network and router address pairs addresses.
34 - TrailerEncapsulation	Whether to negotiate the use of trailers for ARP (RTF893) (boolean).
35 - ARPCacheTimeout	The timeout in seconds for ARP cache entries (32-bit decimal value).
36 - EthernetEncapsulation	Whether to use IEEE 802.3 (RTF1042) rather than of v2 (RFC894) ethernet encapsulation (boolean).
37 - TCPDefaultTTL	Default time-to-live for TCP sends (non-zero 8-bit decimal value).
38 - TCPKeepaliveInterval	The interface in seconds between TCP keepalive messages (32-bit decimal value).
39 - TCPKeepaliveGarbage	Whether to include a byte of garbage in TCP keepalive messages for backward compatibility (boolean).
40 - NISDomain	The NIS (Sun Network Information Services) domain name (string).
41 - NISServers	Space separated list of IPv4 NIS server addresses.
42 - NTPServers	Space separated list of IPv4 NTP (RFC1035) server addresses.
43 - VendorSpecificInfo	Vendor specific information. Only accessible using --set-opt-hex.
44 - NetBIOSNameServers	Space separated list of IPv4 NetBIOS name server (NBNS) addresses (RFC1001,RFC1002).
45 - NetBIOSDatagramServers	Space separated list of IPv4 NetBIOS datagram distribution server (NBDD) addresses (RFC1001,RFC1002).

46 - NetBIOSNodeType	NetBIOS node type (RFC1001,RFC1002): 1=B-node, 2=P-node, 4=M-node, and 8=H-node (8-bit decimal value).
47 - NetBIOSScope	NetBIOS scope (RFC1001,RFC1002). Only accessible using --set-opt-hex.
48 - XWindowsFontServers	Space separated list of IPv4 X windows font server addresses.
49 - XWindowsDisplayManager	Space separated list of IPv4 X windows display manager addresses.
62 - NetWareIPDomainName	Netware IP domain name (RFC2242) (string).
63 - NetWareIPInformation	Netware IP information (RFC2242). Only accessible using --set-opt-hex.
64 - NISPlusDomain	The NIS+ domain name (string).
65 - NISPlusServers	Space separated list of IPv4 NIS+ server addresses.
66 - TFTPServerName	TFTP server name (string).
67 - BootfileName	Bootfile name (string).
68 - MobileIPHomeAgents	Space separated list of IPv4 mobile IP agent addresses.
69 - SMTPServers	Space separated list of IPv4 simple mail transport protocol (SMPT) server addresses.
70 - POP3Servers	Space separated list of IPv4 post office protocol 3 (POP3) server addresses.
71 - NNTPServers	Space separated list of IPv4 network news transport protocol (NTTP) server addresses.
72 - WWWServers	Space separated list of default IPv4 world wide web (WWW) server addresses.
73 - FingerServers	Space separated list of default IPv4 finger server addresses.
74 - IRCServers	Space separated list of default IPv4 internet relay chat (IRC) server addresses.
75 - StreetTalkServers	Space separated list of IPv4 StreetTalk server addresses.
76 - STDAServers	Space separated list of IPv4 StreetTalk directory assistance (STDA) server addresses.
78 - SLPDirectoryAgent	Addresses of one or more service location protocol (SLP) directory agent, and an indicator of whether their use is mandatory. Only accessible using --set-opt-hex.
79 - SLPServiceScope	List of service scopes for the service location protocol (SLP) and whether using the list is mandator. Only accessible using --set-opt-hex.
119 - DomainSearch	Domain search list, see RFC3397 and section 4.1.4 in RFC1035 for encoding. Only accessible using --set-opt-hex.

7.43. VBoxManage debugvm

Introspection and guest debugging.

7.43.1. Synopsis

`VBoxManage debugvm` { *uuid*/*vmname* } dumpvmcore [--filename=*name*]

`VBoxManage debugvm` { *uuid*/*vmname* } info { *item* } [*args* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } injectnmi

`VBoxManage debugvm` { *uuid*/*vmname* } log [[--release] | [--debug]] [*group-settings* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } logdest [[--release] | [--debug]] [*destinations* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } logflags [[--release] | [--debug]] [*flags* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } osdetect

`VBoxManage debugvm` { *uuid*/*vmname* } osinfo

`VBoxManage debugvm` { *uuid*/*vmname* } osdmesg [--lines=*lines*]

`VBoxManage debugvm` { *uuid*/*vmname* } getregisters [--cpu=*id*] [*reg-set.reg-name* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } setregisters [--cpu=*id*] [*reg-set.reg-name=value* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } show [[--human-readable] | [--sh-export] | [--sh-eval] | [--cmd-set]] [*settings-item* ...]

`VBoxManage debugvm` { *uuid*/*vmname* } stack [--cpu=*id*]

`VBoxManage debugvm` { *uuid*/*vmname* } statistics [--reset] [--descriptions] [--pattern=*pattern*]

7.43.2. Description

The "debugvm" commands are for experts who want to tinker with the exact details of virtual machine execution. Like the VM debugger described in The Built-In VM Debugger, these commands are only useful if you are very familiar with the details of the PC architecture and how to debug software.

7.43.2.1. Common options

The subcommands of `debugvm` all operate on a running virtual machine:

uuid/*vmname*   Either the UUID or the name (case sensitive) of a VM.

7.43.2.2. debugvm dumpvmcore

`VBoxManage debugvm` { *uuid*/*vmname* } dumpvmcore [--filename=*name*]

Creates a system dump file of the specified VM. This file will have the standard ELF core format (with custom sections); see VM Core Format.

This corresponds to the `writecore` command in the debugger.

`--filename=filename`   The name of the output file.

7.43.2.3. debugvm info

`VBoxManage debugvm` { *uuid*/*vmname* } info { *item* } [*args* ...]

Displays info items relating to the VMM, device emulations and associated drivers.

This corresponds to the `info` command in the debugger.

`info`	Name of the info item to display. The special name `help` will list all the available info items and hints about optional arguments.
`args`	Optional argument string for the info item handler. Most info items does not take any extra arguments. Arguments not recognized are generally ignored.

7.43.2.4. debugvm injectnmi

`VBoxManage debugvm` { `uuid`/`vmname` } injectnmi

Causes a non-maskable interrupt (NMI) to be injected into the guest. This might be useful for certain debugging scenarios. What happens exactly is dependent on the guest operating system, but an NMI can crash the whole guest operating system. Do not use unless you know what you're doing.

7.43.2.5. debugvm log

`VBoxManage debugvm` { `uuid`/`vmname` } log [[--release] | [--debug]] [`group-settings` ...]

Changes the group settings for either debug (`--debug`) or release (`--release`) logger of the VM process.

The `group-settings` are typically strings on the form `em.e.f.l`, `hm=~0` and `-em.f`. Basic wildcards are supported for group matching. The `all` group is an alias for all the groups.

Please do keep in mind that the group settings are applied as modifications to the current ones.

This corresponds to the `log` command in the debugger.

7.43.2.6. debugvm logdest

`VBoxManage debugvm` { `uuid`/`vmname` } logdest [[--release] | [--debug]] [`destinations` ...]

Changes the destination settings for either debug (`--debug`) or release (`--release`) logger of the VM process. For details on the destination format, the best source is src/VBox/Runtime/common/log/log.cpp.

The `destinations` is one or more mnemonics, optionally prefixed by "no" to disable them. Some of them take values after a ":" or "=" separator. Multiple mnemonics can be separated by space or given as separate arguments on the command line.

List of available destination:

`file[=file], nofile`	Specifies a log file. It no filname is given, one will be generated based on the current UTC time and VM process name and placed in the current directory of the VM process. Note that this will currently not have any effect if the log file has already been opened.
`dir=directory, nodir`	Specifies the output directory for log files. Note that this will currently not have any effect if the log file has already been opened.
`history=count, nohistory`	A non-zero value enables log historization, with the value specifying how many old log files to keep.
`histsize=bytes`	The max size of a log file before it is historized. Default is infinite.
`histtime=seconds`	The max age (in seconds) of a log file before it is historized. Default is infinite.

`ringbuffer, noringbuffer`	Only log to the log buffer until an explicit flush (e.g. via an assertion) occurs. This is fast and saves diskspace.
`stdout, nostdout`	Write the log content to standard output.
`stdout, nostdout`	Write the log content to standard error.
`debugger, nodebugger`	Write the log content to the debugger, if supported by the host OS.
`com, nocom`	Writes logging to the COM port. This is only applicable for raw-mode and ring-0 logging.
`user, nouser`	Custom destination which has no meaning to VM processes..

This corresponds to the `logdest` command in the debugger.

7.43.2.7. debugvm logflags

`VBoxManage debugvm {` *uuid/vmname* `}` logflags [[--release] | [--debug]] [*flags* ...]

Changes the flags on either debug (`--debug`) or release (`--release`) logger of the VM process. Please note that the modifications are applied onto the existing changes, they are not replacing them.

The *flags* are a list of flag mnemonics, optionally prefixed by a "no", "!", "~" or "-" to negate their meaning. The "+" prefix can be used to undo previous negation or use as a separator, though better use whitespace or separate arguments for that.

List of log flag mnemonics, with their counter form where applicable (asterisk indicates defaults):

`enabled*, disabled`	Enables or disables logging.
`buffered, unbuffered*`	Enabling buffering of log output before it hits the destinations.
`writethrough(/writethru)`	Whether to open the destination file with writethru buffering settings or not.
`flush`	Enables flushing of the output file (to disk) after each log statement.
`lockcnts`	Prefix each log line with lock counts for the current thread.
`cpuid`	Prefix each log line with the ID of the current CPU.
`pid`	Prefix each log line with the current process ID.
`flagno`	Prefix each log line with the numberic flags corresponding to the log statement.
`flag`	Prefix each log line with the flag mnemonics corresponding to the log statement.
`groupno`	Prefix each log line with the log group number for the log statement producing it.
`group`	Prefix each log line with the log group name for the log statement producing it.
`tid`	Prefix each log line with the current thread identifier.
`thread`	Prefix each log line with the current thread name.

time	Prefix each log line with the current UTC wall time.
timeprog	Prefix each log line with the current monotonic time since the start of the program.
msprog	Prefix each log line with the current monotonic timestamp value in milliseconds since the start of the program.
ts	Prefix each log line with the current monotonic timestamp value in nanoseconds.
tsc	Prefix each log line with the current CPU timestamp counter (TSC) value.
rel, abs*	Selects the whether ts and tsc prefixes should be displayed as relative to the previous log line or as absolute time.
hex*, dec	Selects the whether the ts and tsc prefixes should be formatted as hexadecimal or decimal.
custom	Custom log prefix, has by default no meaning for VM processes.
usecrlf, uself*	Output with DOS style (CRLF) or just UNIX style (LF) line endings.
overwrite*, append	Overwrite the destination file or append to it.

This corresponds to the `logflags` command in the debugger.

7.43.2.8. debugvm osdetect

`VBoxManage debugvm { uuid/vmname } osdetect`

Make the VMM's debugger facility (re)-detect the guest operating system (OS). This will first load all debugger plug-ins.

This corresponds to the `detect` command in the debugger.

7.43.2.9. debugvm osinfo

`VBoxManage debugvm { uuid/vmname } osinfo`

Displays information about the guest operating system (OS) previously detected by the VMM's debugger facility.

7.43.2.10. debugvm osdmesg

`VBoxManage debugvm { uuid/vmname } osdmesg [--lines=lines]`

Displays the guest OS kernel log, if detected and supported.

--lines=lines	Number of lines of the log to display, counting from the end. The default is infinite.

7.43.2.11. debugvm getregisters

`VBoxManage debugvm { uuid/vmname } getregisters [--cpu=id] [ reg-set.reg-name ...]`

Retrieves register values for guest CPUs and emulated devices.

`reg-set.reg-name`	One of more registers, each having one of the following forms:

1. register-set.register-name.sub-field

2. register-set.register-name

3. cpu-register-name.sub-field

4. cpu-register-name

5. all

The `all` form will cause all registers to be shown (no sub-fields). The registers names are case-insensitive.

`--cpu=id`	Selects the CPU register set when specifying just a CPU register (3rd and 4th form). The default is 0.

7.43.2.12. debugvm setregisters

`VBoxManage debugvm { uuid/vmname } setregisters [--cpu=id] [ reg-set.reg-name=value ...]`

Changes register values for guest CPUs and emulated devices.

`reg-set.reg-name=value`	One of more register assignment, each having one of the following forms:

1. register-set.register-name.sub-field=value

2. register-set.register-name=value

3. cpu-register-name.sub-field=value

4. cpu-register-name=value

The value format should be in the same style as what `getregisters` displays, with the exception that both octal and decimal can be used instead of hexadecimal.

`--cpu=id`	Selects the CPU register set when specifying just a CPU register (3rd and 4th form). The default is 0.

7.43.2.13. debugvm show

`VBoxManage debugvm { uuid/vmname } show [[--human-readable] | [--sh-export] | [--sh-eval] | [--cmd-set]] [ settings-item ...]`

Shows logging settings for the VM.

`--human-readable`	Selects human readable output.
`--sh-export`	Selects output format as bourne shell style `export` commands.
`--sh-eval`	Selects output format as bourne shell style `eval` command input.
`--cmd-set`	Selects output format as DOS style `SET` commands.

settings-item   What to display. One or more of the following:

- logdbg-settings - debug log settings.

- logrel-settings - release log settings.

- log-settings - alias for both debug and release log settings.

7.43.2.14. debugvm stack

`VBoxManage debugvm` { *uuid/vmname* } stack [--cpu=*id*]

Unwinds the guest CPU stacks to the best of our ability. It is recommended to first run the `osdetect` command, as this gives both symbols and perhaps unwind information.

`--cpu=id`   Selects a single guest CPU to display the stack for. The default is all CPUs.

7.43.2.15. debugvm statistics

`VBoxManage debugvm` { *uuid/vmname* } statistics [--reset] [--descriptions] [--pattern=*pattern*]

Displays or resets VMM statistics.

Retrieves register values for guest CPUs and emulated devices.

`--pattern=pattern`   DOS/NT-style wildcards patterns for selecting statistics. Multiple patterns can be specified by using the '|' (pipe) character as separator.

`--reset`   Select reset instead of display mode.

7.44. VBoxManage cloudprofile

Manage the cloud profiles.

7.44.1. Synopsis

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} add [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} update [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} delete

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} show

7.44.2. Description

7.44.2.1. Common options

The subcommands of `cloudprofile` implement the standard CRUD operations for a cloud profile. The next common options must be placed between the "cloud" and the following sub-commands:

--provider=*name*   Short cloud provider name.

--profile=*name*   Cloud profile name.

7.44.2.2. cloudprofile add

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} add [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]

Add new cloud profile for a specified cloud provider.

`--clouduser`   The name which fully identifies the user in the specified cloud provider.

`--fingerprint`   Fingerprint for the key pair being used.

`--keyfile`   Full path and filename of the private key.

`--passphrase`   Passphrase used for the key, if it is encrypted.

`--tenancy`   ID of your tenancy.

`--compartment`   ID of your compartment.

`--region`   Region name. Region is where you plan to deploy an application.

7.44.2.3. cloudprofile show

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} show

Display information about a cloud profile for a specified cloud provider.

7.44.2.4. cloudprofile update

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} update [--clouduser=*unique id*] [--fingerprint=*MD5 string*] [--keyfile=*path*] [--passphrase=*string*] [--tenancy=*unique id*] [--compartment=*unique id*] [--region=*string*]

Modify a cloud profile for the specified cloud provider.

`--clouduser`   The name which fully identifies the user in the specified cloud provider.

`--fingerprint`   Fingerprint for the key pair being used.

`--keyfile`   Full path and filename of the private key.

`--passphrase`   Passphrase used for the key, if it is encrypted.

`--tenancy`   ID of your tenancy.

`--compartment`   ID of your compartment.

`--region`   Region name. Region is where you plan to deploy an application.

7.44.2.5. cloudprofile delete

`VBoxManage cloudprofile` {--provider=*name*} {--profile=*name*} delete

Delete a cloud profile for a specified cloud provider.

7.45. VBoxManage cloud list

The cloud list command gives information about some standard entities in the every Cloud and which can be represented by the list like the list of instances/disk images/networks and etc.

7.45.1. Synopsis

VBoxManage cloud {--provider=*name*} {--profile=*name*} list instances [--state=*string*] [--compartment-id=*string*]

VBoxManage cloud {--provider=*name*} {--profile=*name*} list images [--state=*string*] [--compartment-id=*string*]

7.45.2. Description

7.45.2.1. Common options

The word "cloud" is an umbrella for all commands related to the interconnection with the Cloud. The following common options must be placed between the "cloud" and the following command, in our case "list":

--provider=*name*   Short cloud provider name.

--profile=*name*   Cloud profile name.

7.45.2.2. cloud list instances

VBoxManage cloud {--provider=*name*} {--profile=*name*} list instances [--state=*string*] [--compartment-id=*string*]

Displays the list of the instances for a specified compartment.

--state*"running/paused/ terminated"*	The state of cloud instance. The possible states are "running/paused/ terminated" at moment. If the state isn't provided the list of instances with all possible states is returned.
--compartment-id	A compartment is the logical container used to organize and isolate cloud resources. The different cloud providers can have the different names for this entity.

7.45.2.3. cloud list images

VBoxManage cloud {--provider=*name*} {--profile=*name*} list images [--state=*string*] [--compartment-id=*string*]

Displays the list of the images for a specified compartment.

--state*"available/ disabled/deleted"*	The state of cloud image. The possible states are "available/disabled/ deleted" at moment. If the state isn't provided the list of images with all possible states is returned.
--compartment-id	A compartment is the logical container used to organize and isolate cloud resources. The different cloud providers can have the different names for this entity.

7.46. VBoxManage cloud instance

Manage the cloud instances.

7.46.1. Synopsis

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance create {--domain-name=*name*} {--image-id=*unique id*} {--display-name=*name*} {--shape=*type*} {--subnet=*unique id*} {--boot-disk-size=*size in GB*} [--publicip=*true/false*] [--privateip=*IP address*] [--launch-mode=*NATIVE/EMULATED/ PARAVIRTUALIZED*]

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance info {--id=*unique id*}

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance terminate {--id=*unique id*}

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance start {--id=*unique id*}

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance pause {--id=*unique id*}

7.46.2. Description

7.46.2.1. Common options

The subcommands of `cloudinstance` implement the standard operations for a cloud instance like start/ pause/show/terminate. The next common options must be placed between the "cloud" and the following sub-commands:

--provider=*name*   Short cloud provider name.

--profile=*name*   Cloud profile name.

7.46.2.2. cloud instance create

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} instance create {--domain-name=*name*} {--image-id=*unique id*} {--display-name=*name*} {--shape=*type*} {--subnet=*unique id*} {--boot-disk-size=*size in GB*} [--publicip=*true/false*] [--privateip=*IP address*] [--launch-mode=*NATIVE/EMULATED/ PARAVIRTUALIZED*]

Creates new instance in the Cloud. There are two standard ways to create an instance in the Cloud: 1. Create an instance from an existing custom image. 2. Create an instance from an existing bootable volume. For the 1st approach next parameters are required: image-id and boot-disk-size. For the 2nd approach next parameters are required: boot-volume-id; The rest parameters are common for both cases: display-name, launch-mode, subnet-id, publicIP, privateIP, shape, domain.

--domain-name   Cloud domain where new instance is created.

--image-id   Unique identifier which fully identifies a custom image in the Cloud.

--display-name   Name for new instance in the Cloud.

--shape   The shape of instance, defines the number of CPUs and RAM memory.

--subnet   Unique identifier which fully identifies an existing subnet in the Cloud which will be used by the instance.

--boot-disk-size   The size of bootable image in GB. Default is 50GB.

--publicip   Whether the instance will have a public IP or not.

--privateip   Private IP address for the created instance.

--launch-mode   The most known values here may be EMULATED, NATIVE,
                              PARAVIRTUALIZED.

7.46.2.3. cloud instance info

Display information about a cloud instance with a specified id.

--id   Unique identifier which fully identify the instance in the Cloud.

7.46.2.4. cloud instance termination

Delete a cloud instance with a specified id.

--id   Unique identifier which fully identify the instance in the Cloud.

7.46.2.5. cloud instance start

Start a cloud instance with a specified id.

--id   Unique identifier which fully identify the instance in the Cloud.

7.46.2.6. cloud instance pause

Pause a cloud instance with a specified id.

--id   Unique identifier which fully identify the instance in the Cloud.

7.47. VBoxManage cloud image

Manage the cloud images.

7.47.1. Synopsis

VBoxManage cloud {--provider=*name*} {--profile=*name*} image create {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*] [--instance-id=*unique id*]

VBoxManage cloud {--provider=*name*} {--profile=*name*} image info {--id=*unique id*}

VBoxManage cloud {--provider=*name*} {--profile=*name*} image delete {--id=*unique id*}

VBoxManage cloud {--provider=*name*} {--profile=*name*} image import {--id=*unique id*} [--bucket-name=*name*] [--object-name=*name*]

VBoxManage cloud {--provider=*name*} {--profile=*name*} image export {--id=*unique id*} {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*]

7.47.2. Description

7.47.2.1. Common options

The subcommands of cloudimage implement the standard operations for a cloud image like create/delete/show/import/export. The next common options must be placed between the "cloud" and the following sub-commands:

--provider=*name*   Short cloud provider name.

--profile=*name*   Cloud profile name.

7.47.2.2. cloud image create

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image create {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*] [--instance-id=*unique id*]

Creates new image in the Cloud. There are two standard ways to create an image in the Cloud: 1. Create an image from an object in the Cloud Storage; 2. Create an image from an existing cloud instance. For the 1st approach next parameters are required: bucket-name - cloud bucket name where an object is located; object-name - name of object in the bucket; display-name - name for new image in the Cloud. For the 2d approach next parameters are required: instance-id - Id of instance in the Cloud; display-name - name for new image in the Cloud.

`--display-name`   Name for new image in the Cloud.

`--bucket-name`   Cloud bucket name where an object is located.

`--object-name`   Name of object in the bucket.

`--instance-id`   Unique identifier which fully identifies the instance in the Cloud.

7.47.2.3. cloud image info

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image info {--id=*unique id*}

Display information about a cloud image with a specified id.

`--id`   Unique identifier which fully identifies the image in the Cloud.

7.47.2.4. cloud image delete

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image delete {--id=*unique id*}

Delete an image with a specified id from the Cloud.

`--id`   Unique identifier which fully identifies the image in the Cloud.

7.47.2.5. cloud image import

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image import {--id=*unique id*} [--bucket-name=*name*] [--object-name=*name*]

Import an image with a specified id from the Cloud to a local host. The result is an object in the local "temp" folder on the local host. Possible approach may have two general steps: 1. Create an object from an image in the Cloud Storage; 2. Download the object to the local host. So the next parameters may be required: bucket-name - cloud bucket name where the object will be created; object-name - name of object in the bucket. if parameter "object-name" is absent a displayed image name is used. If the first step isn't needed only the parameter "id" is required.

`--id`   Unique identifier which fully identifies the image in the Cloud.

`--bucket-name`   Cloud bucket name where an object will be created.

`--object-name`   Name of created object in the bucket. The downloaded object will have this name.

7.47.2.6. cloud image export

`VBoxManage cloud` {--provider=*name*} {--profile=*name*} image export {--id=*unique id*} {--display-name=*name*} [--bucket-name=*name*] [--object-name=*name*]

Export an existing VBox image with a specified uuid from a local host to the Cloud. The result is new image in the Cloud. Possible approach may have two general steps: 1. Upload VBox image to the Cloud Storage; 2. Create an image from the uploaded object. So the next parameters may be required: bucket-name -cloud bucket name where the object will be uploaded; object-name - name of object in the bucket. If parameter "object-name" is absent the image id is used; display-name - name for new image in the Cloud. If the first step isn't needed the parameters "id" and "display-name" are required only.

`--id`   Unique identifier of the image in the VirtualBox.

`--display-name`   Name for new image in the Cloud.

`--bucket-name`   Cloud bucket name where the image (object) will be uploaded.

`--object-name`   Name of object in the bucket.

Glossary

A

ACPI
    Advanced Configuration and Power Interface, an industry specification for BIOS and hardware extensions to configure PC hardware and perform power management. Windows 2000 and later, as well as Linux 2.4 and later support ACPI. Windows can only enable or disable ACPI support at installation time.

AHCI
    Advanced Host Controller Interface, the interface that supports SATA devices such as hard disks. See Section 5.1, "Hard Disk Controllers".

AMD-V
    The hardware virtualization features built into modern AMD processors. See Hardware vs. Software Virtualization.

API
    Application Programming Interface.

APIC
    Advanced Programmable Interrupt Controller, a newer version of the original PC PIC (programmable interrupt controller). Most modern CPUs contain an on-chip APIC, called a local APIC. Many systems also contain an I/O APIC (input output APIC) as a separate chip which provides more than 16 IRQs. Windows 2000 and later use a different kernel if they detect an I/O APIC during installation. Therefore, an I/O APIC must not be removed after installation.

ATA
    Advanced Technology Attachment, an industry standard for hard disk interfaces which is synonymous with IDE. See Section 5.1, "Hard Disk Controllers".

B

BIOS
    Basic Input/Output System, the firmware built into most personal computers which is responsible of initializing the hardware after the computer has been turned on and then booting an operating system. Oracle VM VirtualBox ships with its own virtual BIOS that runs when a virtual machine is started.

C

COM
    Microsoft Component Object Model, a programming infrastructure for modular software. COM enables applications to provide application programming interfaces which can be accessed from various other programming languages and applications. Oracle VM VirtualBox makes use of COM both internally and externally to provide a comprehensive API to 3rd party developers.

D

DHCP
    Dynamic Host Configuration Protocol. This enables a networking device in a network to acquire its IP address and other networking details automatically, in order to avoid having to configure all devices in a network with fixed IP addresses. Oracle VM VirtualBox has a built-in DHCP server that delivers an IP addresses to a virtual machine when networking is configured to NAT. See Chapter 6, *Virtual Networking*.

E

EFI

Extensible Firmware Interface, a firmware built into computers which is designed to replace the aging BIOS. Originally designed by Intel, most modern operating systems can now boot on computers which have EFI instead of a BIOS built into them. See Section 3.14, "Alternative Firmware (EFI)".

EHCI

Enhanced Host Controller Interface, the interface that implements the USB 2.0 standard.

G

GUI

Graphical User Interface. Commonly used as an antonym to a "command line interface". In the context of Oracle VM VirtualBox, we sometimes refer to the main graphical `VirtualBox` program as the "GUI", to differentiate it from the `VBoxManage` interface.

GUID

See UUID.

I

I/O APIC

See APIC.

IDE

Integrated Drive Electronics, an industry standard for hard disk interfaces. See Section 5.1, "Hard Disk Controllers".

iSCSI

Internet SCSI. See Section 5.10, "iSCSI Servers".

M

MAC

Media Access Control, a part of an Ethernet network card. A MAC address is a 6-byte number which identifies a network card. It is typically written in hexadecimal notation where the bytes are separated by colons, such as `00:17:3A:5E:CB:08`.

MSI

Message Signaled Interrupts, as supported by modern chipsets such as the ICH9. See Section 3.5.1, "Motherboard Tab". As opposed to traditional pin-based interrupts, with MSI, a small amount of data can accompany the actual interrupt message. This reduces the amount of hardware pins required and allows for more interrupts and better performance.

N

NAT

Network Address Translation. A technique to share networking interfaces by which an interface modifies the source and/or target IP addresses of network packets according to specific rules. Commonly employed by routers and firewalls to shield an internal network from the Internet, Oracle VM VirtualBox can use NAT to easily share

a host's physical networking hardware with its virtual machines. See Section 6.3, "Network Address Translation (NAT)".

O

OVF

Open Virtualization Format, a cross-platform industry standard to exchange virtual appliances between virtualization products. See Section 1.14, "Importing and Exporting Virtual Machines".

P

PAE

Physical Address Extension. This enables access to more than 4 GB of RAM, even in 32-bit environments. See Section 3.4.2, "Advanced Tab".

PIC

See APIC.

PXE

Preboot Execution Environment, an industry standard for booting PC systems from remote network locations. It includes DHCP for IP configuration and TFTP for file transfer. Using UNDI, a hardware independent driver stack for accessing the network card from bootstrap code is available.

R

RDP

Remote Desktop Protocol, a protocol developed by Microsoft as an extension to the ITU T.128 and T.124 video conferencing protocol. With RDP, a PC system can be controlled from a remote location using a network connection over which data is transferred in both directions. Typically graphics updates and audio are sent from the remote machine and keyboard and mouse input events are sent from the client. An Oracle VM VirtualBox extension package by Oracle provides VRDP, an enhanced implementation of the relevant standards which is largely compatible with Microsoft's RDP implementation. See Remote Display (VRDP Support) for details.

S

SAS

Serial Attached SCSI, an industry standard for hard disk interfaces. See Section 5.1, "Hard Disk Controllers".

SATA

Serial ATA, an industry standard for hard disk interfaces. See Section 5.1, "Hard Disk Controllers".

SCSI

Small Computer System Interface. An industry standard for data transfer between devices, especially for storage. See Section 5.1, "Hard Disk Controllers".

SMP

Symmetrical Multiprocessing, meaning that the resources of a computer are shared between several processors. These can either be several processor chips or, as is more common with modern hardware, multiple CPU cores in one processor.

SSD

Solid-state drive, uses microchips for storing data in a computer system. Compared to classical hard-disks they are having no mechanical components like spinning disks.

T

TAR

A widely used file format for archiving. Originally, this stood for Tape ARchive and was already supported by very early UNIX versions for backing up data on tape. The file format is still widely used today. For example, with OVF archives using an `.ova` file extension. See Section 1.14, "Importing and Exporting Virtual Machines".

U

UUID

A Universally Unique Identifier, often also called GUID (Globally Unique Identifier). A UUID is a string of numbers and letters which can be computed dynamically and is guaranteed to be unique. Generally, it is used as a global handle to identify entities. Oracle VM VirtualBox makes use of UUIDs to identify VMs, Virtual Disk Images (VDI files), and other entities.

V

VM

Virtual Machine. A virtual computer that Oracle VM VirtualBox enables you to run on top of your actual hardware. See Section 1.2, "Some Terminology" for details.

VMM

Virtual Machine Manager. The component of Oracle VM VirtualBox that controls VM execution. See Oracle VM VirtualBox Executables and Components for a list of Oracle VM VirtualBox components.

VRDE

VirtualBox Remote Desktop Extension. This interface is built into Oracle VM VirtualBox to allow Oracle VM VirtualBox extension packages to supply remote access to virtual machines. An Oracle VM VirtualBox extension package by Oracle provides VRDP support. See Remote Display (VRDP Support).

VRDP

See RDP.

VT-x

The hardware virtualization features built into modern Intel processors. See Hardware vs. Software Virtualization.

X

xHCI

eXtended Host Controller Interface, the interface that implements the USB 3.0 standard.

XML

The eXtensible Markup Language, a metastandard for all kinds of textual information. XML only specifies how data in the document is organized generally and does not prescribe how to semantically organize content.

XPCOM

Mozilla Cross Platform Component Object Model, a programming infrastructure developed by the Mozilla browser project which is similar to Microsoft COM and enables applications to provide a modular programming interface. Oracle VM VirtualBox makes use of XPCOM on Linux both internally and externally to provide a comprehensive API to third-party developers.